# Multicores, Manycores, and Acclerators

**18-613**: Foundations of Computer Systems
7th Lecture, **March 21, 2019**

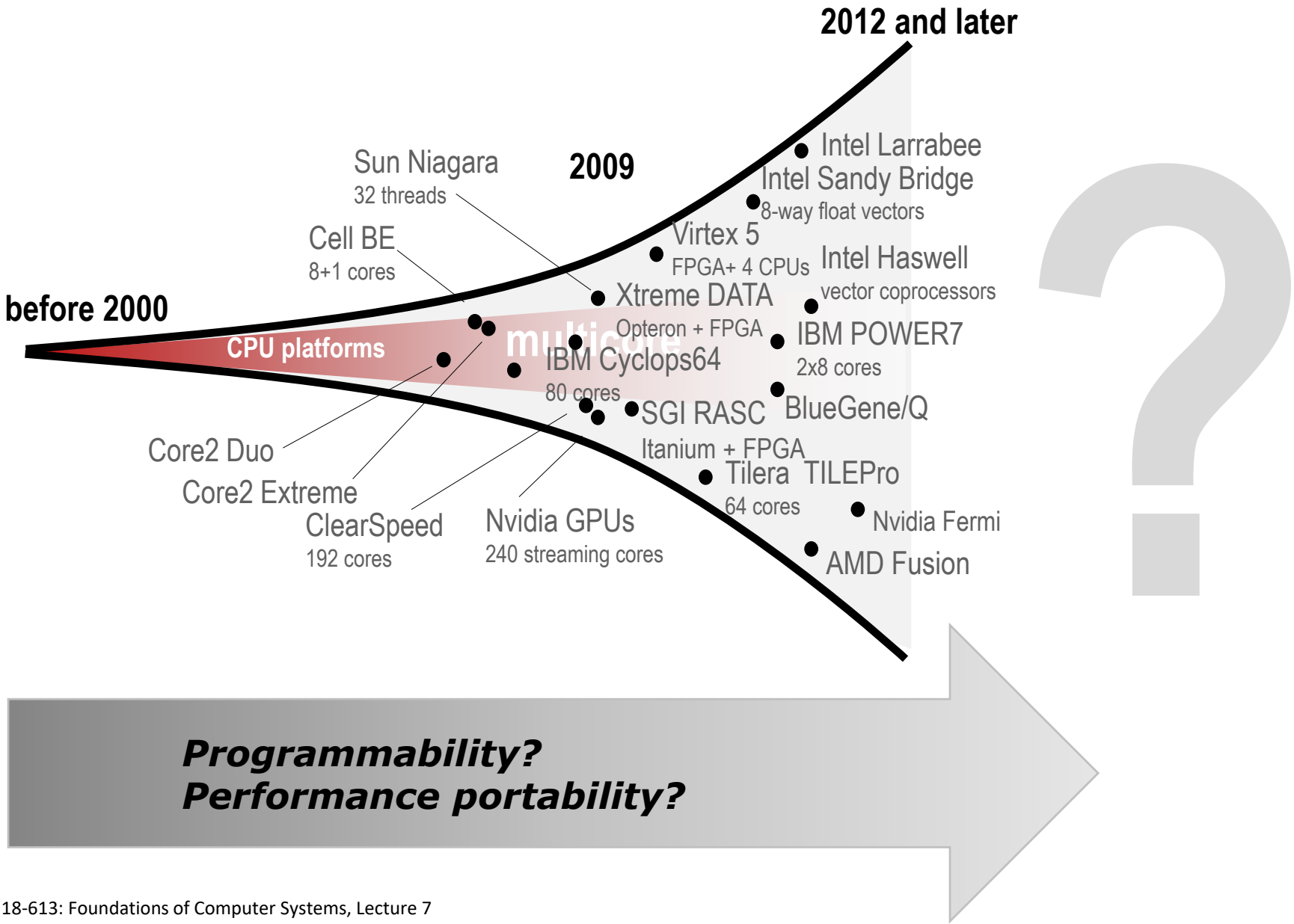
**Instructor:**

**Franz Franchetti**

# Outline

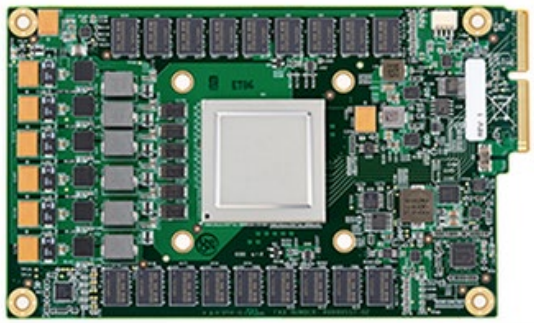
- **Multicores, Manycores, accelerators**
- **Experimental accelerators**
- **Latest Nvidia GPUs**
- **CUDA**
- **Summary**

# The Future is Parallel and Heterogeneous

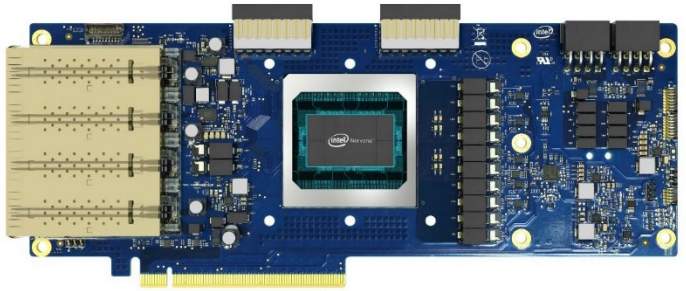


# The Future Is Here: Accelerators

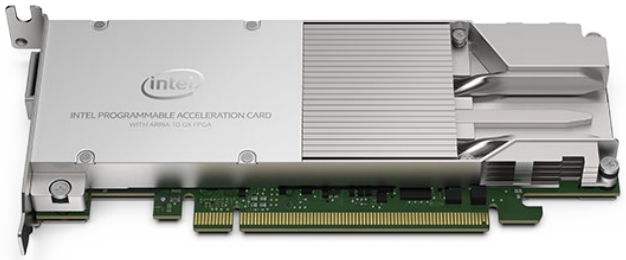
**Google TPU 2.0**
600 GB/s, 45 Tflop/s



**Intel Nervana Neural Network Processor**
aka Lake Crest, release date 2019



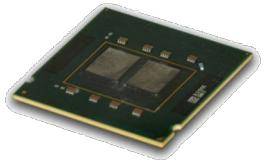

**Nvidia V100 GPU**
100 Tflop/s tensor operations
7.5 Tflop/s FP64 SIMT




**Intel Arria 10 GX FPGA**
15Gbps SERDES transceivers
1,150K logic elements, 53 Mb
8 GB DDR4 ECC

# Intel Multicores and Manycores

## Desktop and Server Multicores



**2005**
Pentium D
2 cores, 3.6 GHz

**2007**
Core 2 Quad
4 cores, 2.67 GHz


**2010**
Core i7 980X
6 cores, 3.3 GHz

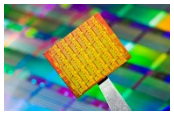
**2014**
Xeon E5-4624Lv2
10 cores, 2.5 GHz

**2015**
Xeon E7-8890 v3
18 cores, 2.5 GHz

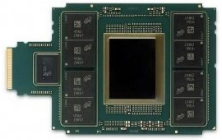

**2019**
Xeon Platinum 8176F
28 cores, 3.8 GHz

## Experimental Chips and Server Manycores

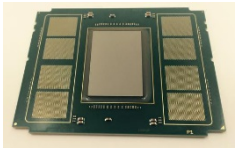

**2008**
Larrabee GPU
32 cores, 2 GHz



**2009**
Intel SCC
48 cores, 1GHz

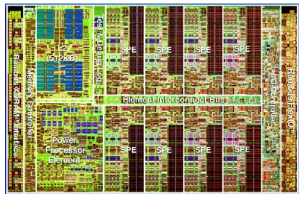


**2013**
Xeon Phi 5120D
60 cores, 1.05 GHz



**2016**
Xeon Phi 7290F
72 cores, 1.5 GHz

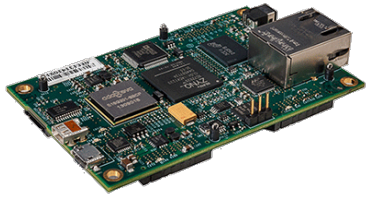
# Other CPU-Style Multicores/Manycores



**2006**
**IBM Cell BE**
8+1 cores, 3.2 GHz


**2007**
**Tilera Tile64**
64 cores, 900 MHz

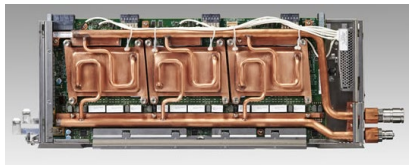

**2008**
**ClearSpeed CSX700**
192 cores, 250 MHz



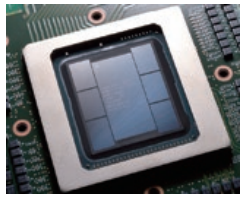
**2012**
**Parallella**
16 cores, 1GHz

**2013**
**IBM POWER9**
24 cores, 4 GHz


**2013**
**Oracle SPARC T5**
16 cores, 3.6 GHz



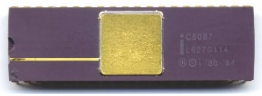
**2014**
**Fujitsu SPARC64 XIfx**
32+2 cores, 2.2 GHz



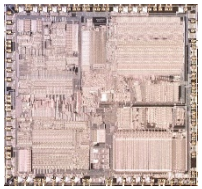
**2017**
**NEC SX-Aurora**
8 cores, 1.6 GHz

# Math Co-Processors and Sound Cards

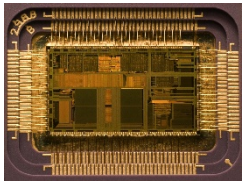
## x87 Co-Processors



**1980**
**Intel 8087**
x87 FPU, 10 MHz



**1987**
**Intel 80387**
IEEE754 x87 FPU, 33 MHz

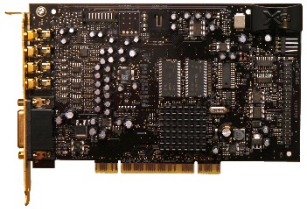


**1989**
**Intel 80486**
x87 FPU integrated


**1990**
**Weitek 4167**
x87 FPU, 33 MHz

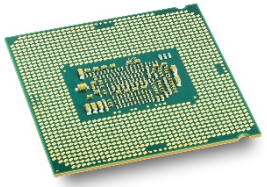
## Audio Co-Processors

**1989**
**Sound Blaster 1.0**
11 voice FM synthesizer



**2012**
**SoundBlaster X-Fi**
Advanced DSP, 48kHz

**2008**
**Platform Controller Hub**
Sound, clock, display



**2014**
**On chip PCH**
SoC design

# Graphics Processors (GPUs)

## Nvidia

**1995**
**Nvidia NV1**
2D/3D GPU

**2006**
**Nvidia G80**
CUDA


**2009**
**Nvidia Tesla C1060**
double GPU computing

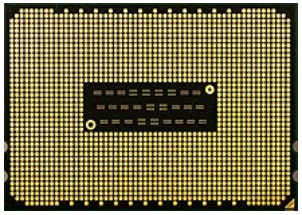
**2019**
**Nvidia V100**
Tensor cores, 125 Tflop/s

## ATI/AMD

**1992**
**ATI Mach32**
GUI acceleration

**2000**
**ATI Radeon**
DirexctX 7.0 3D, MPEG-1

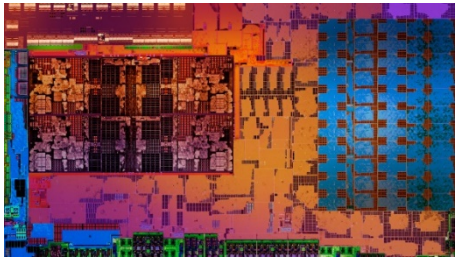
**2014**
**Radeon R9 295X2**
1.43 Tflop/s double



**2017**
**Opteron X3000 APU**
4 cores + 3rd gen GCN GPU

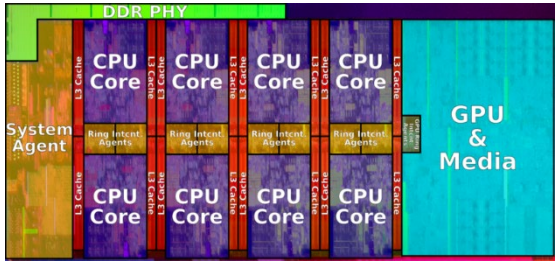
# Intel GPUs

**1998**
**Intel 740**
VGA card

**2008**
**Larrabee**
Discrete GPU

**2008**
**GMA X4500HD**
G45 chipset





**2018**
**UHD 630**
Coffe Lake, on-die

# Embedded and System on a Chip (SoC)

**Raspberry Pi**
Broadcom BCM2837B0
1.4 GHz, ARM Cortex A53


**Qualcomm**
Snapdragon 855
ARM CPU, GPU, DSP, CV modem


**Apple A12**
6 cores, ARM, 2.5 GHz


**Intel Quark**
1 core, 32 MHz

Qualcomm® Snapdragon™
## 855 Mobile Platform

Qualcomm snapdragon

SPECIFICATIONS & FEATURES

### Qualcomm® Artificial Intelligence Engine
- Qualcomm® Hexagon™ 690 Processor
- Qualcomm® Hexagon™ Vector Accelerator
- Qualcomm® Hexagon™ Tensor Accelerator
- Qualcomm® Hexagon™ Voice Assistant
- Qualcomm® All-Ways Aware™ Hub
- Qualcomm® Adreno™ 640 GPU
- Qualcomm® Kryo™ 485 CPU

### 5G Modem
- Snapdragon™ X50 5G Modem
- Supported Technologies: 5G NR
- Support for sub-6 GHz and mmWave
- Supported Spectrum:
  - mmWave: 800 MHz bandwidth, 8 carriers, 2x2 MIMO
  - Sub-6 GHz: 100 MHz bandwidth, 4x4 MIMO

### 4G Modem
- Snapdragon™ X24 LTE modem
- Cellular Technologies: LTE FDD, LTE TDD with CBRS support, LAA, LTE Broadcast, WCDMA, TD-SCDMA, CDMA 1x, EV-DO, GSM/EDGE
- Downlink: LTE Cat 20 up to 2.0 Gbps, 7x20 MHz carrier aggregation, 256-QAM, 4x4 MIMO on five carriers
- Uplink: LTE Cat 13 up to 318 Mbps, 3x20 MHz carrier aggregation, 256-QAM, Uplink Data Compression
- Multi SIM with support for Dual SIM Dual VoLTE (DSDV) and LTE Dual SIM Dual Standby (DSDS) +LAA
- Next-generation Calling Services including VoLTE with SRVCC to 3G and 2G, HD and Ultra HD Voice (EVS), and CSFB to 3G and 2G

### Wi-Fi
- Qualcomm® Wi-Fi 6-ready mobile platform
  - Wi-Fi Standards: 802.11ax-ready, 802.11ac Wave 2, 802.11a/b/g, 802.11n
  - Wi-Fi Spectral Bands: 2.4 GHz, 5 GHz
  - Channel Utilization: 20/40/80 MHz
  - MIMO Configuration: 2x2 (2-stream)
  - MU-MIMO
  - Dual-band simultaneous (DBS)
  - Key Features: 8x8 sounding (up to 2x improvement over 4x4 sounding devices), Target Wakeup Time for up to 67% better power efficiency, latest security with WPA3
- Qualcomm® 60 GHz Wi-Fi mobile platform
  - Wi-Fi Standards: 802.11ad, 802.11ay
  - Wi-Fi Spectral Band: 60 GHz
  - Peak speed: 10 Gbps
  - Key Features: multi-gigabit speeds, wire-equivalent latency, always-on ambient Wi-Fi sensing

### Camera
- Qualcomm® Spectra™ 380 Image Sensor Processor
- Dual 14-bit ISPs
- Hardware accelerator for computer vision (CV-ISP)
- Up to 22 MP dual camera
- Up to 48 MP single camera
- Rec. 2020 color gamut video capture
- Up to 10-bit color depth video capture
- Slow motion video capture at 720p HDR, 480fps
- HEIF photo capture, HEVC video capture
- Video Capture Formats: HDR10+, HDR10, HLG
- 4K HDR Video Capture with Portrait Mode (Bokeh)
- Multi-frame Noise Reduction (MFNR)
- Real-time object classification, segmentation and replacement

### Audio
- Low-power Audio Subsystem with AI
- Hexagon Voice Assistant Accelerator for advanced voice use cases
- Qualcomm Aqstic™ Audio Technology
  - Native DSD support, PCM up to 384 kHz/32-bit
  - Supports two wake words simultaneously (Google Assistant, Amazon Alexa, Baidu, Cortana)
- Always-on echo cancellation and noise suppression; up to 4 mic far-field mic support for better voice recognition in tough conditions
- Qualcomm® aptX™ Adaptive audio technology
  - aptX Adaptive adjusts performance for optimum quality, latency, and efficiency
- Qualcomm TrueWireless™ Stereo Plus technology

### Display
- Maximum On-Device Display Support: Up to 4K HDR
- Maximum External Display Support: Up to two 4K HDR displays
- 10-bit color depth, Rec. 2020 color gamut

### CPU
- 8x Qualcomm® Kryo™ 485
- Up to 2.84 GHz
- 64-bit Architecture
- 7nm Process Technology

### Visual Subsystem
- Adreno 640 GPU
- Vulkan® 1.1 API support
- HDR gaming (10-bit color depth, Rec. 2020 color gamut)
- Physically Based Rendering
- API Support: OpenGL® ES 3.2, OpenCL™ 2.0 FP, Vulkan 1.1
- Hardware-accelerated H.265 and VP9 decoder
- HDR Playback Codec support for HDR10+, HDR10, HLG and Dolby Vision
- Volumetric VR video playback
- 8K 360 VR video playback

### RF Front-End
- Comprehensive 5G/4G modem-to-antenna solution
- QTM052 5G mmWave antenna module
- Qualcomm Adaptive Antenna Tuning
- Qualcomm Envelope Tracking
- High-power transmit (HPUE)

### Security
- Secure Processing Unit featuring mobile payments and Qualcomm® 3D Sonic Sensor
- Biometric Authentication: Fingerprint, Iris, Voice, Face
- On-Device: Qualcomm® Mobile Security, Key Provisioning Security, Qualcomm® Processor Security, Qualcomm® Content Protection, Qualcomm® Trusted Execution Environment, Camera Security, Crypto Engine, Malware Protection, Secure Boot, Secure Token

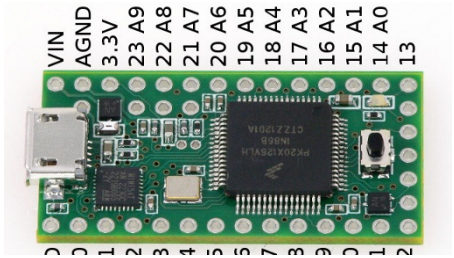
### Memory
- Memory Speed: 2133 MHz
- Memory Type: 4x16-bit, LPDDR4x
- Memory Density: up to 16 GB

### Connectivity Specifications
- Bluetooth Version: 5.0
- Bluetooth Speed: 2 Mbps
- Satellite Systems Support: Dual frequency GNSS, GPS, GLONASS, Beidou, Galileo, QZSS, SBAS
- Near Field Communications (NFC): Supported
- USB Version 3.1; USB Type-C Support

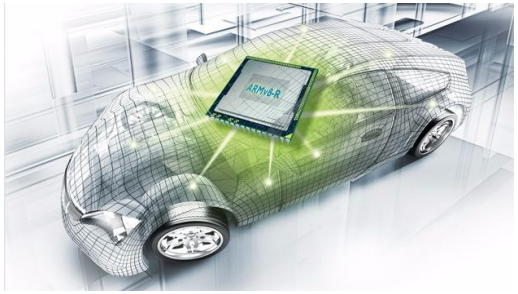
### Charging
- Qualcomm® Quick Charge™ 4+ technology

# ARM

## Processor classes



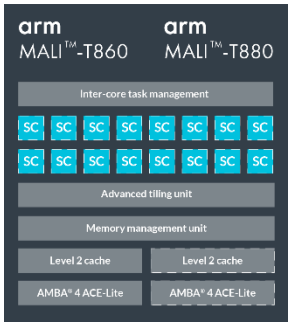

**Cortex-M**
Targets SoC



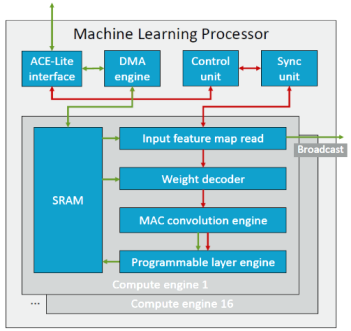
**Cortex-R**
Real-time


**Cortex-A**
Application processors

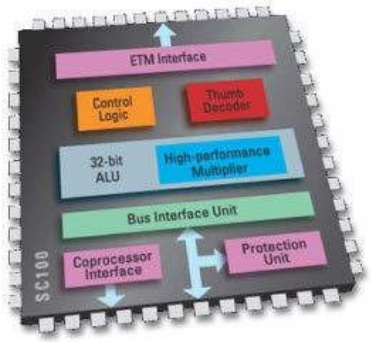
## Accelerators and special features




**Mali GPU**
ARM GPU




**Trillium**
Machine learning




**SecurCore**
Tamper proof, security

# Field Programmable Gate Arrays (FPGAs)

**PicoZed**
Xilinx Zynq-7000 SoC

**Xilinx Virtex**
UltraScale+ HBM VCU128-ES1

- 8GB of on-chip High Bandwidth Memory (HBM)
- Multiple external memory interfaces (RLDRAM3, QDR-IV, DDR4)
- Quad 32Gbps QSFP28 Interfaces
- PCIe Gen3 x16 & Gen4 x8
- VITA 57.4 FMC+ Interface
- 10/100/1000 Mbps Ethernet

**Intel Arria 10 GX**
1,150K logic elements

**BittWare TeraBox**
8 FPGA cards, 2 TB DDR4

**24 TeraFLOPS processing: 16x Altera Arria 10 or Stratix V FPGAs**
- Up to 18 million logic elements (Arria 10 GX)
- Up to 62,000 multipliers (Stratix V GS)

**1.28 Terabits/sec I/O**

**128x 10GigE, 32x 40GigE, 32x 100 GigE, or 32x QDR Infiniband**

**6.5 Terabits/sec memory bandwidth**
- Up to 64 banks DDR3-1600 (512 GBytes)
- DDR4, QDRII+, and RLDRAM3 memory options

**4U or 5U Rackmount PCIe system (server, industrial, or expansion)**
- Dual socket Intel Ivy Bridge with up to 12 cores
- Up to 768 GBytes of system memory
- 8 Gen3 x16 PCIe slots

**Complete software support**
- Windows and Linux 64 drivers, interface libraries, and hardware management
- FPGA development kit for Arria 10 and Stratix V

# Memory and Packaging

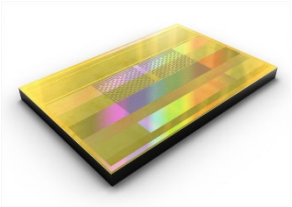
## Memory chips

**DDR4 DIMM**
Standard memory

**SO-DIMM**
Laptops


**GDDR5**
For GPUs

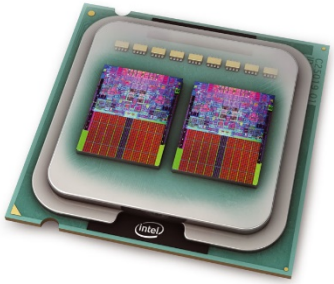
**HMC**
3D memory on PCB



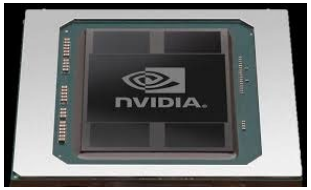
**HBM2**
For SoC/SiP

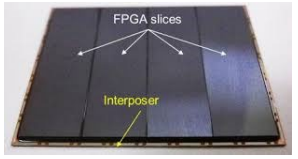
## Integration

**Printed circuit board (PCB)**
Traditional circuit board



**Multi chip packaging (MCP)**
Tighter integration




**Silicon interposer**
Chip-on-chip integration

# Bulk Storage and Busses

## Bulk storage (disks)


**Hard drive**
Standard disk


**SSD**
Flash as disk drive



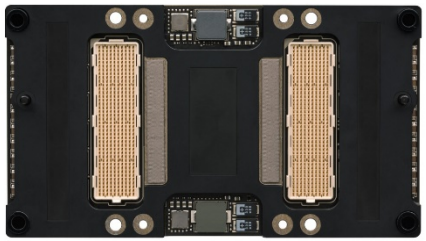
**NVMe**
HDD on PCIe



**SD card**
Memory for cell phone etc

## Extension busses

**PCIe x1, x4, x8, x16**
Desktop/laptop standard



**NVLink**
Fast GPU interconnect

**Coherent Accelerator Processor Interface (CAPI)**
IBM/server class open standard

# Outline

- Multicores, Manycores, accelerators
- **Experimental accelerators**
- Latest Nvidia GPUs
- CUDA
- Summary

***Based on "15-740 Computer Architecture" by Nathan Beckmann***

# Example 1: DianNao

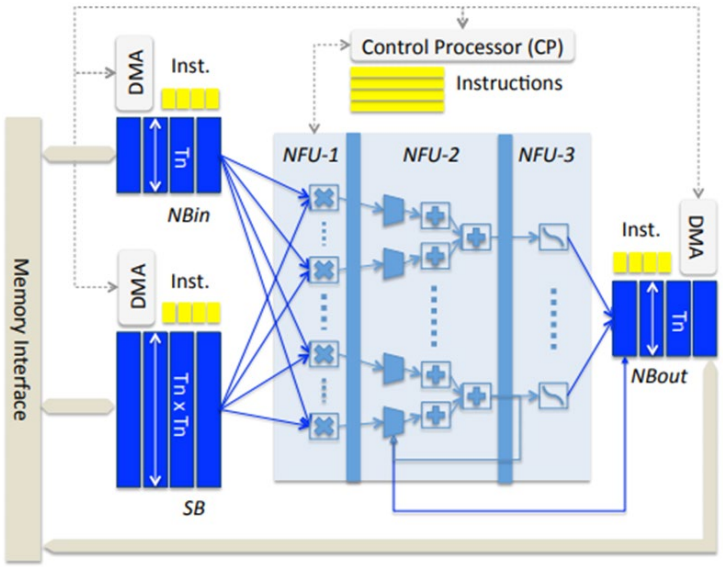



**Figure 11.** *Accelerator.*

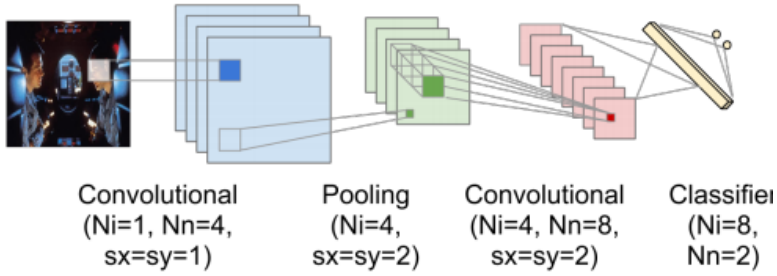



**Figure 1.** *Neural network hierarchy containing convolutional, pooling and classifier layers.*

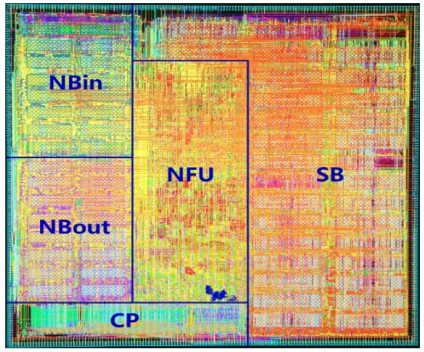
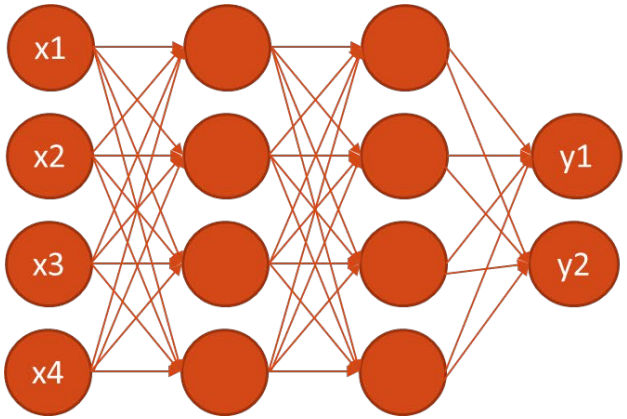
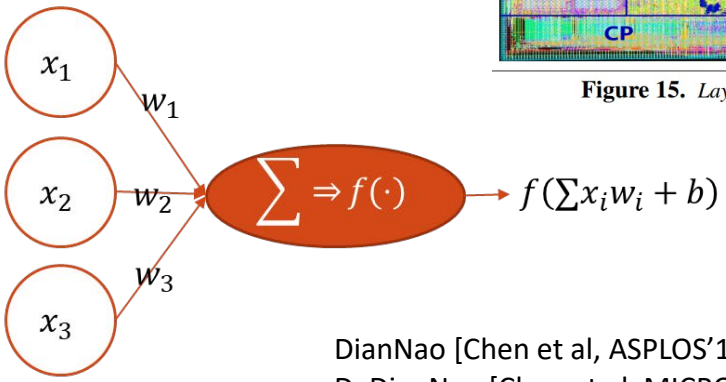


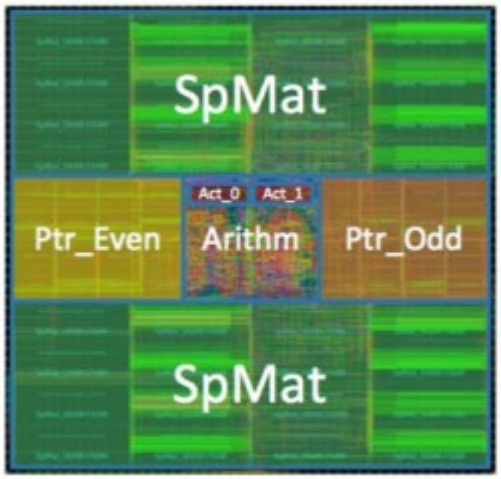

**Figure 15.** *Layout (65nm).*

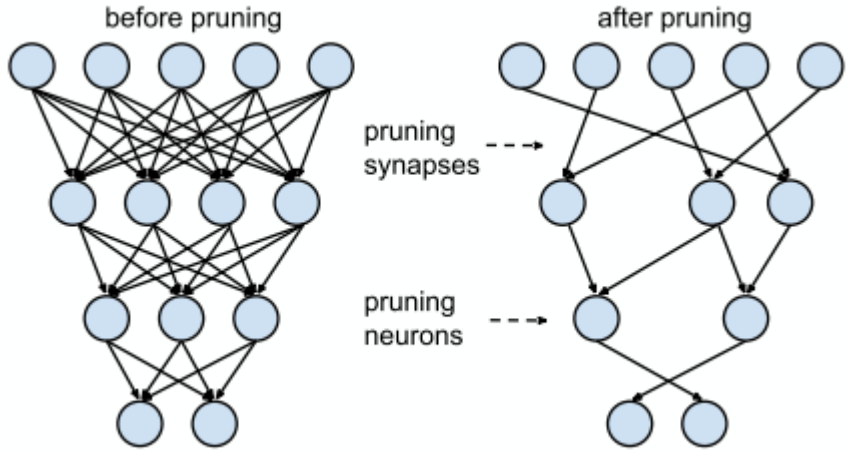


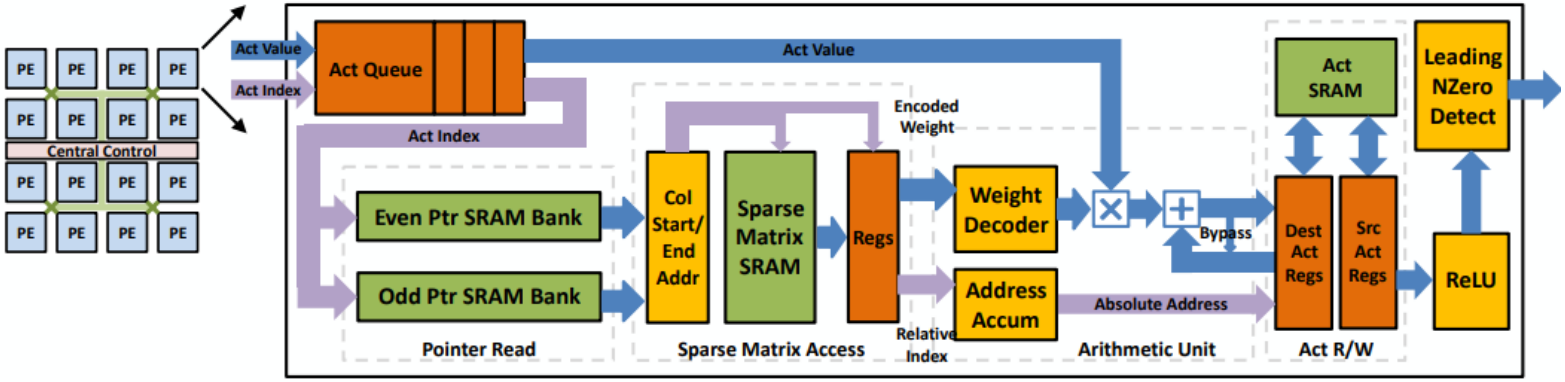




DianNao [Chen et al, ASPLOS'14, Best paper]
DaDianNao [Chen et al, MICRO'14, Best paper]
PuDianNao [Liu et al, ASPLOS'15]
ShiDianNao [Du et al, ISCA'15]

# Example 2: EIE: Sparse Neural Networks



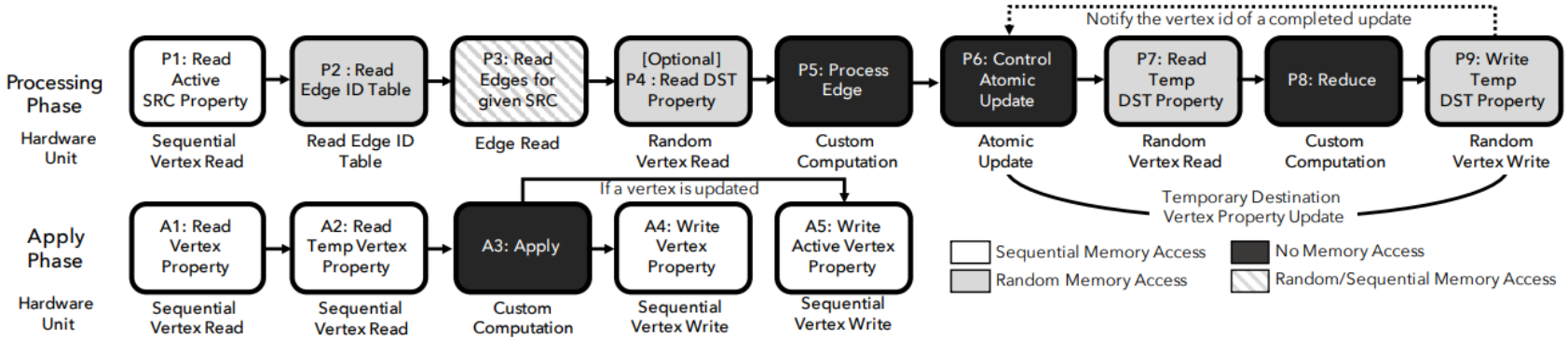




EIE: Efficient Inference Engine

**[Han et al, ISCA'16]**

# Example 3: Graphicianado



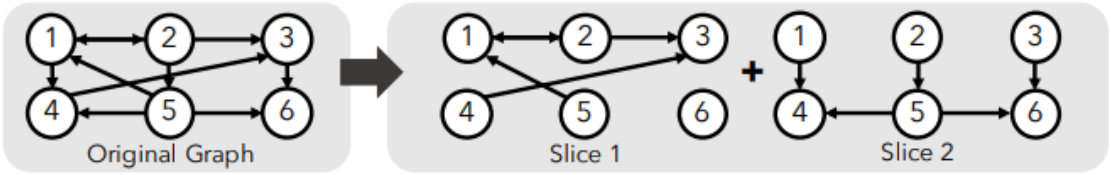




Fig. 10: Graph slicing Example.

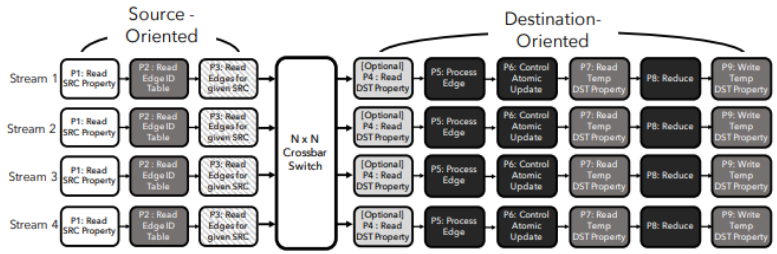



Fig. 7: Parallel implementation of Graphicionado. This diagram omits the *Apply* phase pipeline which is parallelized in a similar manner.

**GraphMat Processing Model**

```
1 For each Vertex V
2   For each incoming edge E(U,V) from active vertex U
3     Res ← Process_Edge (E_weight, U_prop, [OPTIONAL]V_prop)
4     V_temp ← Reduce(V_temp, Res)
5   End
6 End
7 For each Vertex V,
8   V_prop ← Apply(V_temp, V_prop, V_const)
9 End
```

**[Ham et al, MICRO'16, Best paper]**

# Example 4: Plasticine

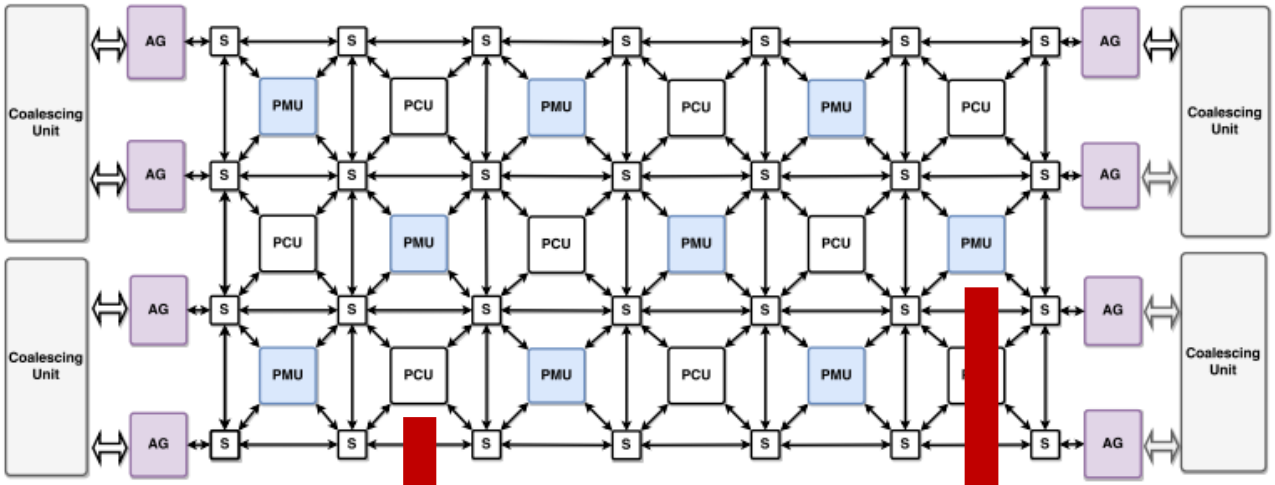



Figure 5: Plasticine chip-level architecture ( ual organization 16 x 8). All three networks have t same structure.
PCU:Pattern Compute Unit, PMU: Pattern emory Unit, AG: Address Generator, S: Switch Bo

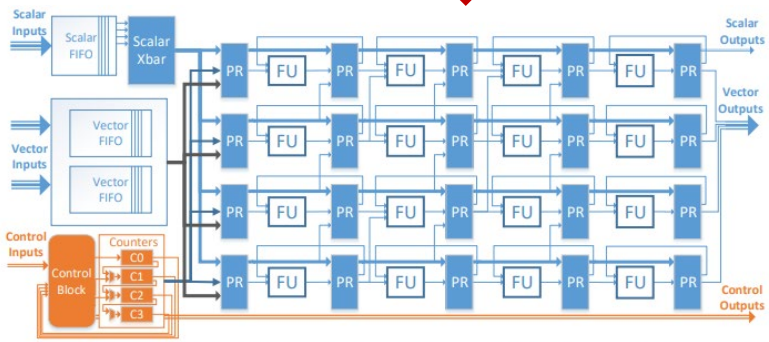



Figure 3: Pattern Compute Unit (PCU) architecture. We show only 4 stages and 4 SIMD lanes, and omit some control signals.

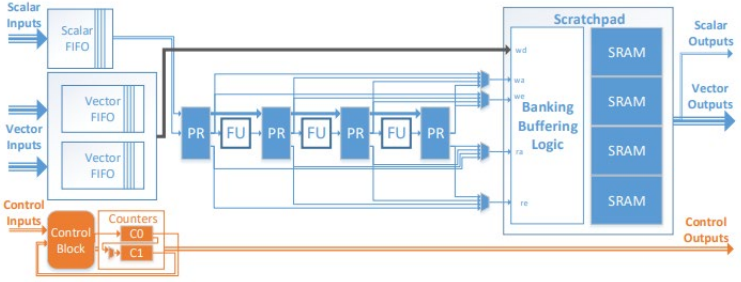


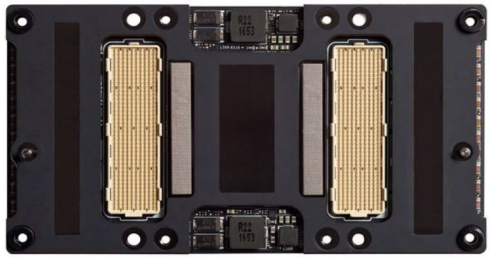

Figure 4: Pattern Memory Unit (PMU) architecture: configurable scratchpad, address calculation datapath, and control.

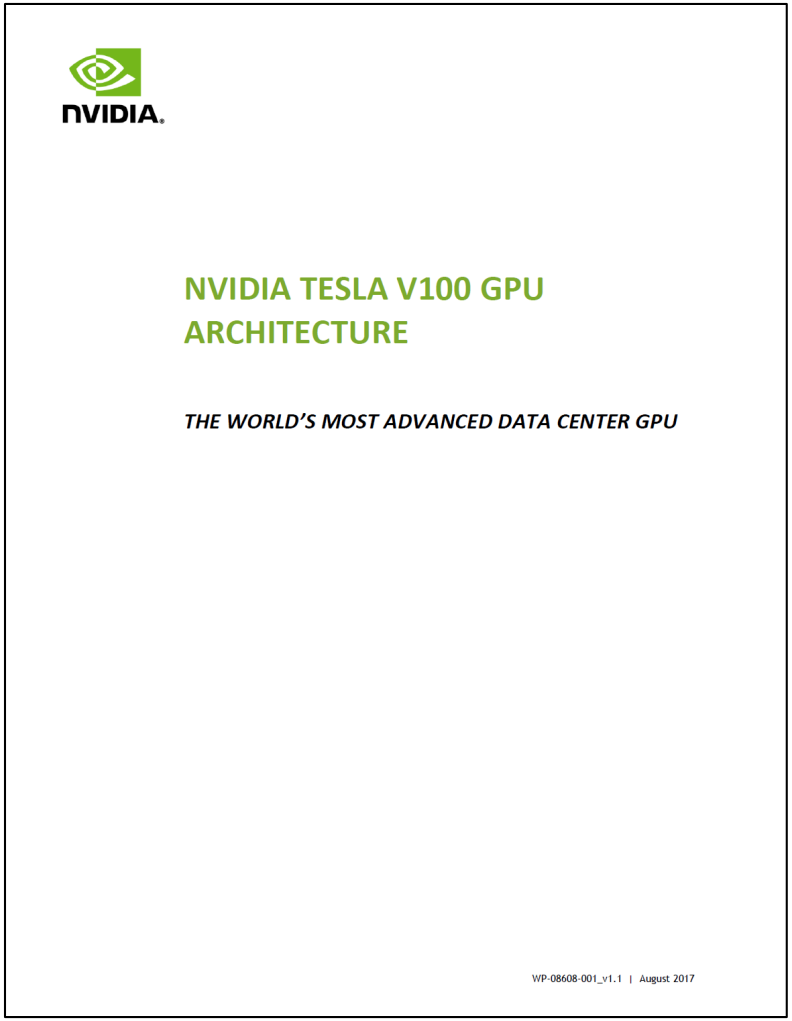
Pattern based reconfigurable computing

**[Prabhakar et al, ISCA'17]**

# Outline

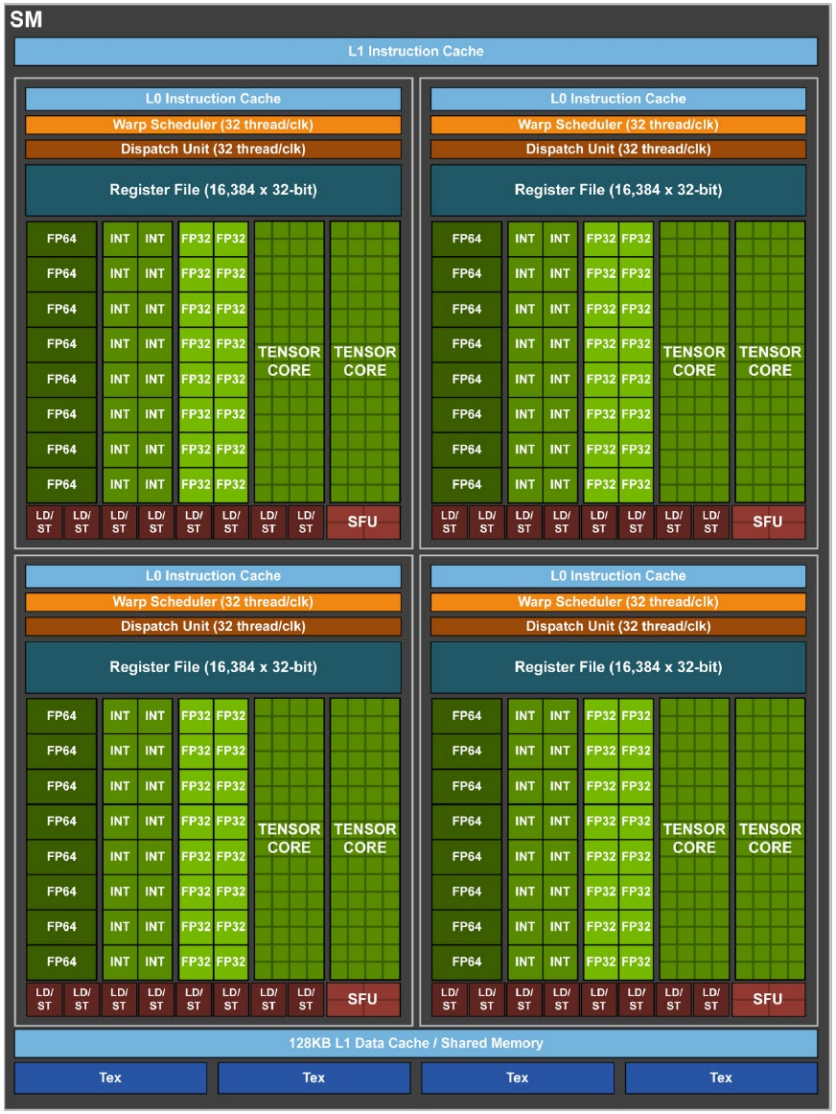
- Multicores, Manycores, accelerators
- Experimental accelerators
- **Latest Nvidia GPUs**
- CUDA
- Summary



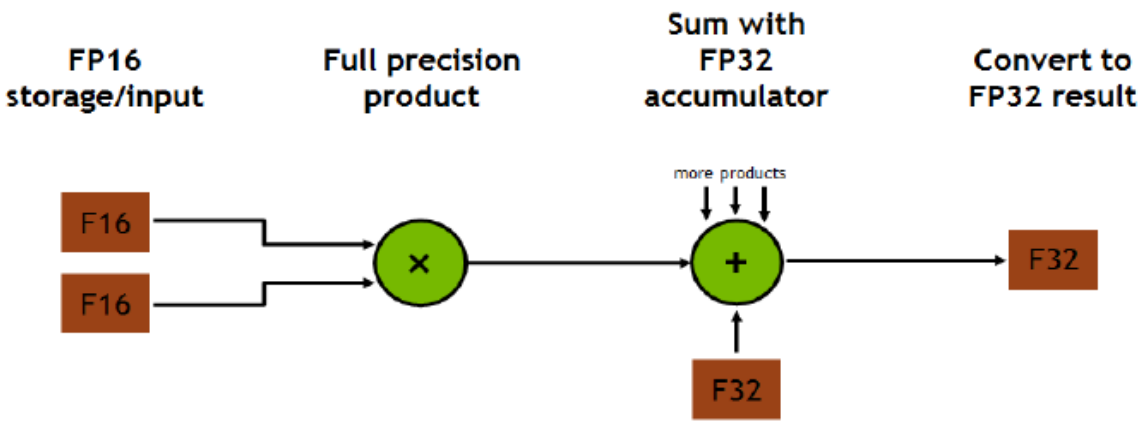
# Volta GV100 with 84 SM Units

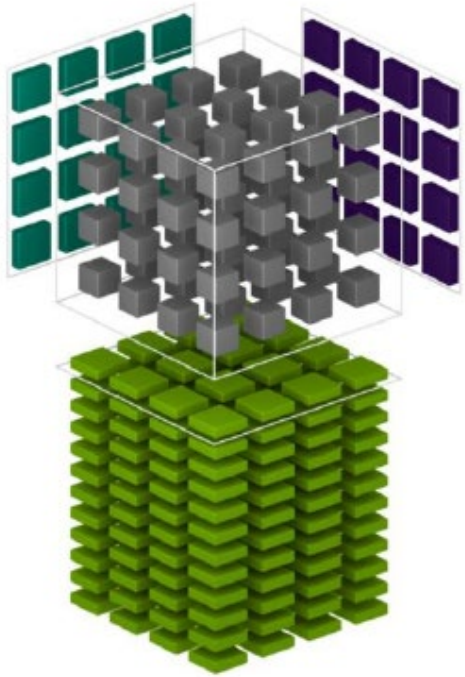
# Volta GV100 Streaming Multiprocessor (SM)



# Tensor Cores

$$D = \underbrace{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}}_{\text{FP16 or FP32}} \underbrace{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}}_{\text{FP16}} + \underbrace{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}_{\text{FP16 or FP32}}$$



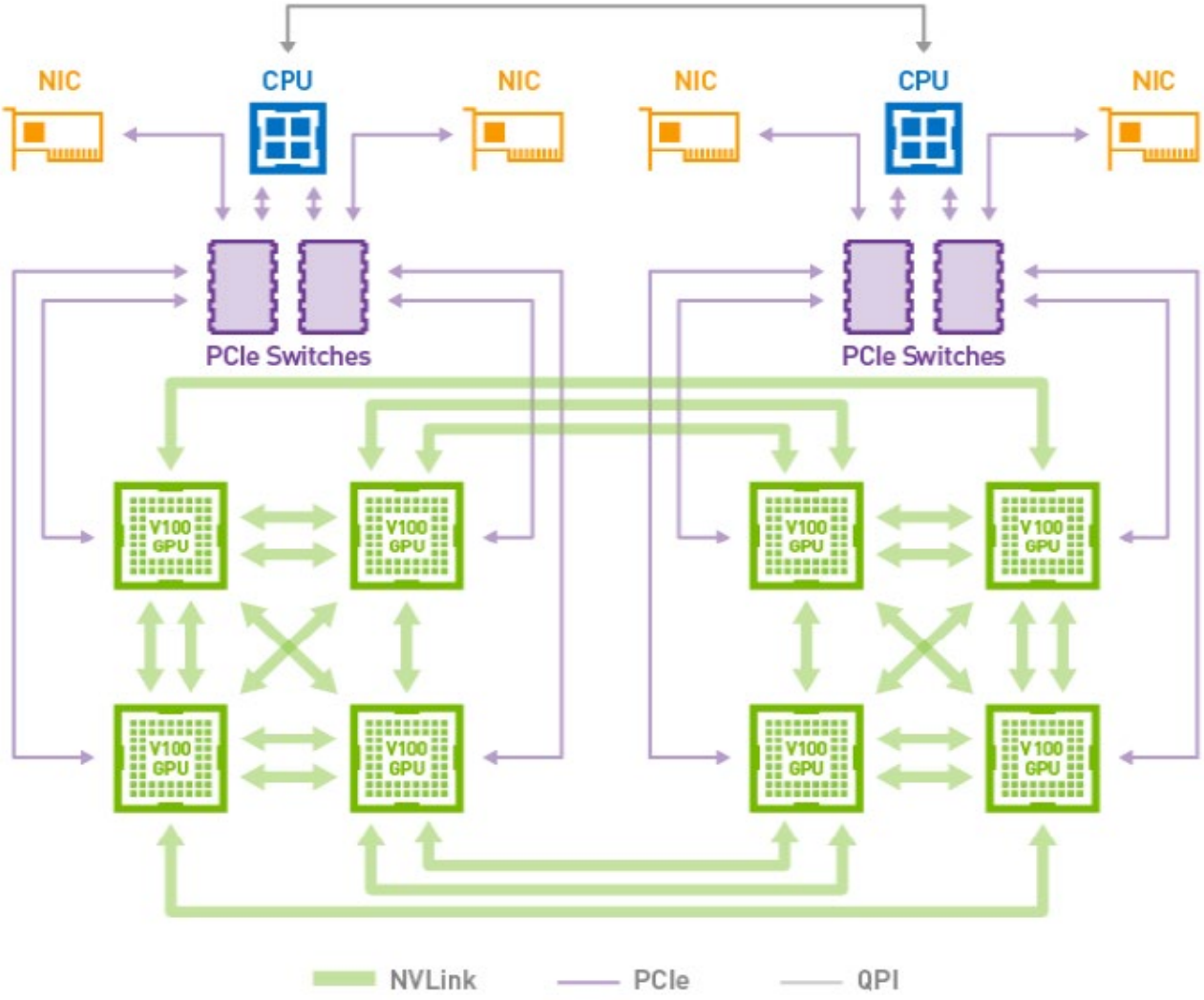



# Volta GV100 with 84 SM Units

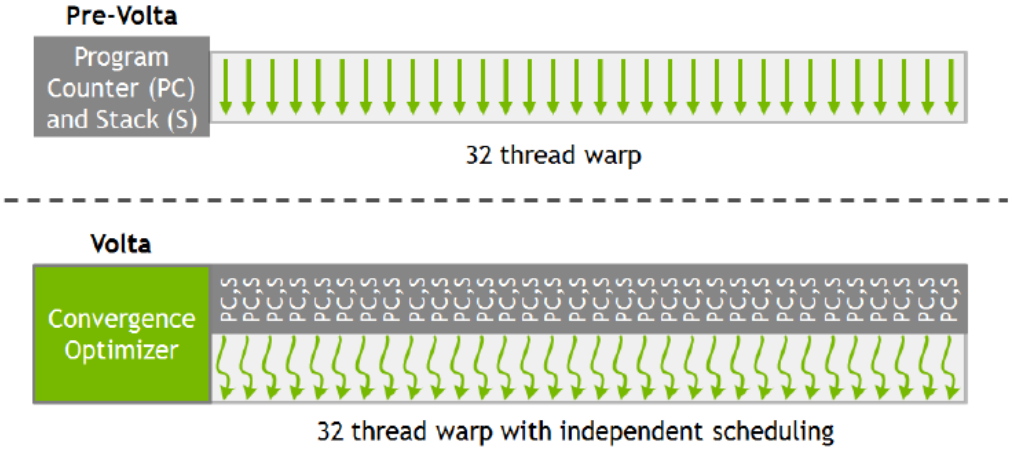
| Tesla Product | Tesla K40 | Tesla M40 | Tesla P100 | Tesla V100 |
|---|---|---|---|---|
| GPU | GK180 (Kepler) | GM200 (Maxwell) | GP100 (Pascal) | GV100 (Volta) |
| SMs | 15 | 24 | 56 | 80 |
| TPCs | 15 | 24 | 28 | 40 |
| FP32 Cores / SM | 192 | 128 | 64 | 64 |
| FP32 Cores / GPU | 2880 | 3072 | 3584 | 5120 |
| FP64 Cores / SM | 64 | 4 | 32 | 32 |
| FP64 Cores / GPU | 960 | 96 | 1792 | 2560 |
| Tensor Cores / SM | NA | NA | NA | 8 |
| Tensor Cores / GPU | NA | NA | NA | 640 |
| GPU Boost Clock | 810/875 MHz | 1114 MHz | 1480 MHz | 1530 MHz |
| Peak FP32 TFLOPS[1] | 5 | 6.8 | 10.6 | 15.7 |
| Peak FP64 TFLOPS[1] | 1.7 | .21 | 5.3 | 7.8 |
| Peak Tensor TFLOPS[1] | NA | NA | NA | 125 |
| Texture Units | 240 | 192 | 224 | 320 |
| Memory Interface | 384-bit GDDR5 | 384-bit GDDR5 | 4096-bit HBM2 | 4096-bit HBM2 |
| Memory Size | Up to 12 GB | Up to 24 GB | 16 GB | 16 GB |
| L2 Cache Size | 1536 KB | 3072 KB | 4096 KB | 6144 KB |
| Shared Memory Size / SM | 16 KB/32 KB/48 KB | 96 KB | 64 KB | Configurable up to 96 KB |
| Register File Size / SM | 256 KB | 256 KB | 256 KB | 256KB |
| Register File Size / GPU | 3840 KB | 6144 KB | 14336 KB | 20480 KB |
| TDP | 235 Watts | 250 Watts | 300 Watts | 300 Watts |
| Transistors | 7.1 billion | 8 billion | 15.3 billion | 21.1 billion |
| GPU Die Size | 551 $mm^2$ | 601 $mm^2$ | 610 $mm^2$ | 815 $mm^2$ |
| Manufacturing Process | 28 nm | 28 nm | 16 nm FinFET+ | 12 nm FFN |

[1] Peak TFLOPS rates are based on GPU Boost Clock

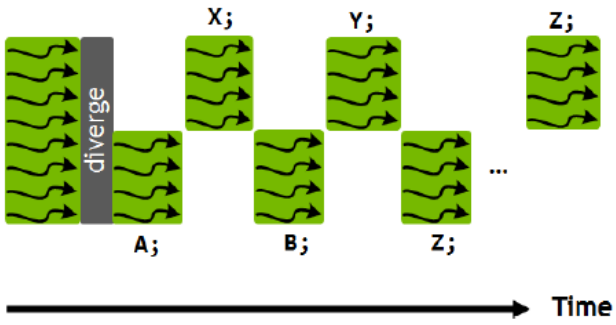
# Multi-GPU Systems: NVLink



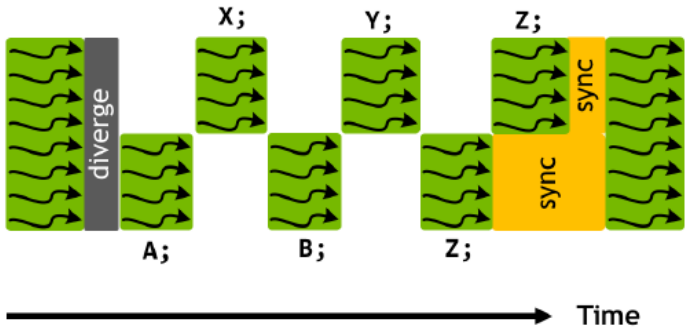
# Thread Divergence




```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```




```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
__syncwarp()
```

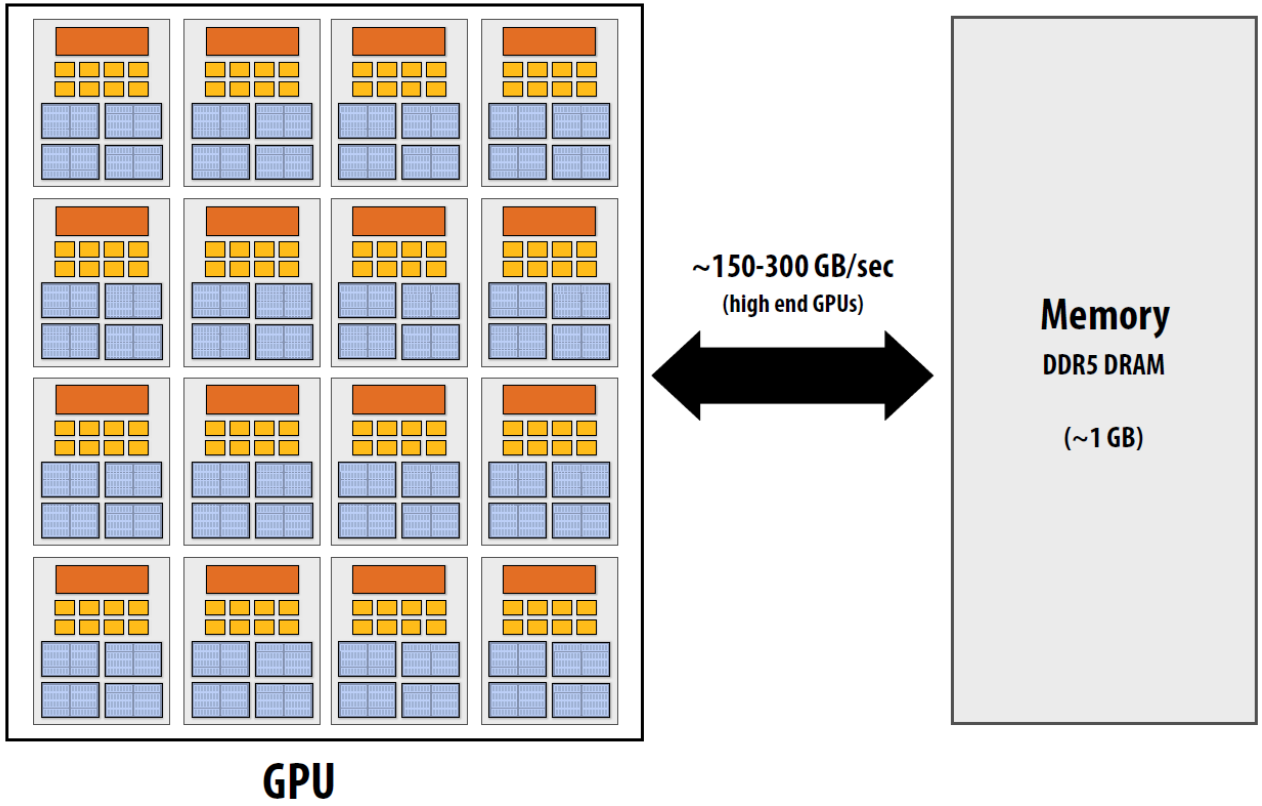


# Outline

- Multicores, Manycores, accelerators
- Experimental accelerators
- Latest Nvidia GPUs
- **CUDA**
- Summary

***Based on "15-418/15-618: Parallel Computer Architecture and Programming" by Randy Bryant and Nathan Beckmann***

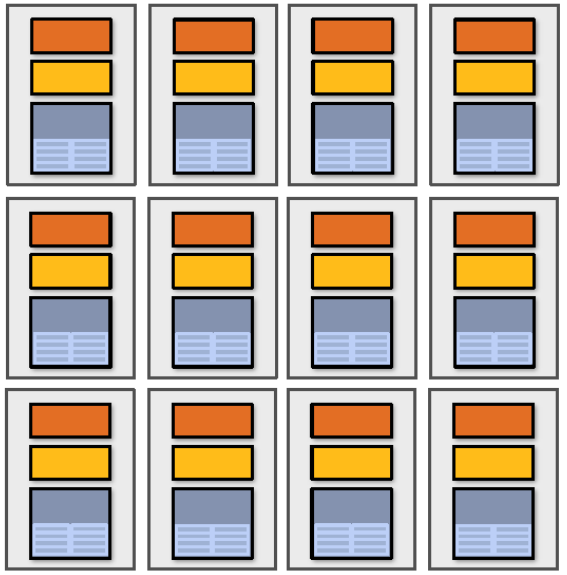
# GPU Architecture




- Multi-core chip
- SIMD execution within a single core (many execution units performing the same instruction)
- Multi-threaded execution on a single core (multiple threads executed concurrently by a core)

# NVIDIA Tesla architecture (2007)

- (GeForce 8xxx series GPUs)
  First alternative, non-graphics-specific ("compute mode") interface to GPU hardware
- Lets say a user wants to run a non-graphics program on the GPU's programmable cores…
  - Application can allocate buffers in GPU memory and copy data to/from buffers
  - Application (via graphics driver) provides GPU a single kernel program binary
  - Application tells GPU to run the kernel in an SPMD fashion ("run N instances")
  - Go! (launch(myKernel, N))



# CUDA Programming Language

- **Introduced in 2007 with NVIDIA Tesla architecture**
- **"C-like" language to express programs that run on GPUs using the compute-mode hardware interface**
- **Relatively low-level: CUDA's abstractions closely match the capabilities/performance characteristics of modern GPUs (design goal: maintain low abstraction distance)**
- **Note: OpenCL is an open standards version of CUDA**
  - CUDA only runs on NVIDIA GPUs
  - OpenCL runs on CPUs and GPUs from many vendors
  - Almost everything we say about CUDA also holds for OpenCL

# Basic CUDA Syntax

- **"Host" code:** serial execution
  Running as part of normal C/C++ application on CPU
- **Bulk launch of many CUDA threads**
  "launch a grid of CUDA thread blocks"
  Call returns when all threads have terminated
- **SPMD execution of device kernel function:**
- "CUDA device" code: kernel function (**`__global__`** denotes a CUDA kernel function) runs on GPU
- Each thread computes its overall grid thread id from its position in its block (**`threadIdx`**) and its block's position in the grid (**`blockIdx`**)

```
const int Nx = 12;
const int Ny = 6;

dim3 threadsPerBlock(4, 3, 1);
dim3 numBlocks(Nx/threadsPerBlock.x,
               Ny/threadsPerBlock.y, 1);

// assume A, B, C are allocated Nx x Ny float arrays

// this call will trigger execution of 72 CUDA threads:
// 6 thread blocks of 12 threads each
matrixAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
```

```
// kernel definition
__global__ void matrixAdd(float A[Ny][Nx],
                          float B[Ny][Nx],
                          float C[Ny][Nx])
{
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  int j = blockIdx.y * blockDim.y + threadIdx.y;

  C[j][i] = A[j][i] + B[j][i];
}
```

# Clear Separation of Host and Device Code

- **Separation of execution into host and device code is performed statically by the programmer**

**"Host" code : serial execution on CPU**

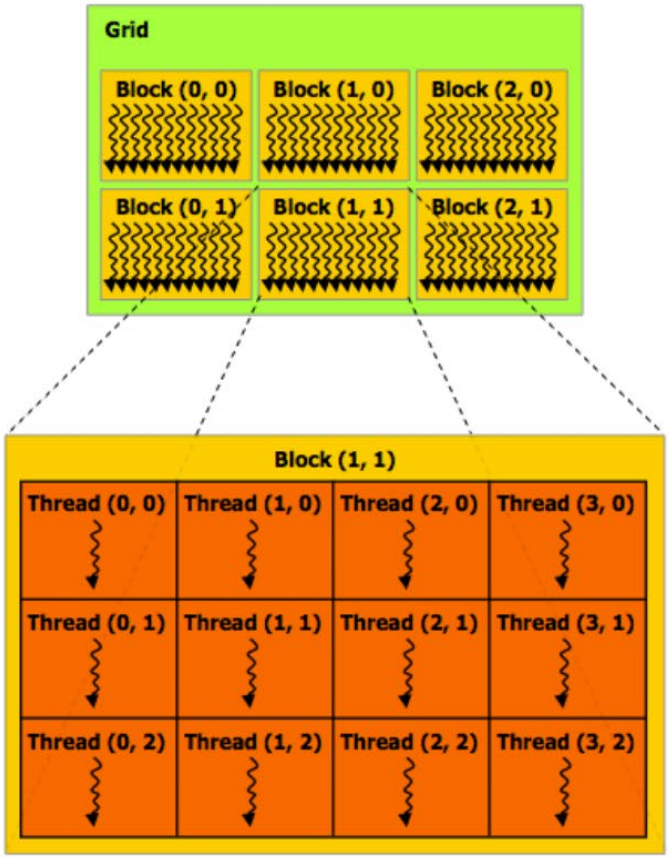
```
const int Nx = 12;
const int Ny = 6;

dim3 threadsPerBlock(4, 3, 1);
dim3 numBlocks(Nx/threadsPerBlock.x,
               Ny/threadsPerBlock.y, 1);

// assume A, B, C are allocated Nx x Ny float arrays

// this call will cause execution of 72 threads
// 6 blocks of 12 threads each
matrixAddDoubleB<<<numBlocks, threadsPerBlock>>>(A, B, C);
```

**"Device" code (SPMD execution on GPU)**

```
__device__ float doubleValue(float x)
{
   return 2 * x;
}

// kernel definition
__global__ void matrixAddDoubleB(float A[Ny][Nx],
                                 float B[Ny][Nx],
                                 float C[Ny][Nx])
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   int j = blockIdx.y * blockDim.y + threadIdx.y;

   C[j][i] = A[j][i] + doubleValue(B[j][i]);
}
```

# Number of SPMD Threads is Explicit in Program

- Number of kernel invocations is not determined by size of data collection (a kernel launch is not map(kernel, collection) as was the case with graphics shader programming)




**Regular application thread running on CPU (the "host")**

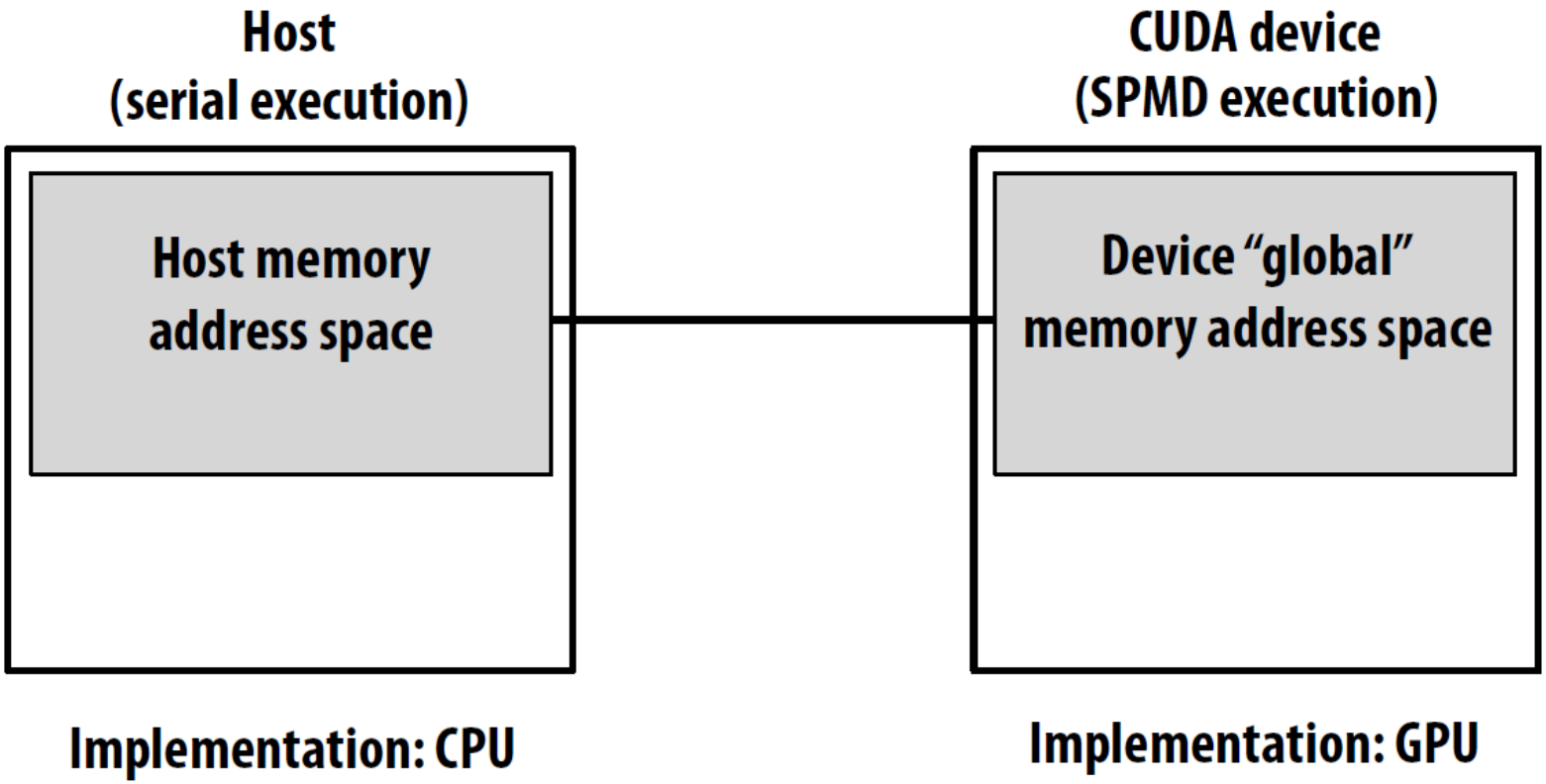
```
const int Nx = 11;  // not a multiple of threadsPerBlock.x
const int Ny = 5;   // not a multiple of threadsPerBlock.y

dim3 threadsPerBlock(4, 3, 1);
dim3 numBlocks((Nx+threadsPerBlock.x-1)/threadsPerBlock.x,
               (Ny+threadsPerBlock.y-1)/threadsPerBlock.y, 1);

// assume A, B, C are allocated Nx x Ny float arrays

// this call will cause execution of 72 threads
// 6 blocks of 12 threads each
matrixAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
```

**CUDA kernel definition**

```
__global__ void matrixAdd(float A[Ny][Nx],
                          float B[Ny][Nx],
                          float C[Ny][Nx])
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   int j = blockIdx.y * blockDim.y + threadIdx.y;

   // guard against out of bounds array access
   if (i < Nx && j < Ny)
      C[j][i] = A[j][i] + B[j][i];
}
```

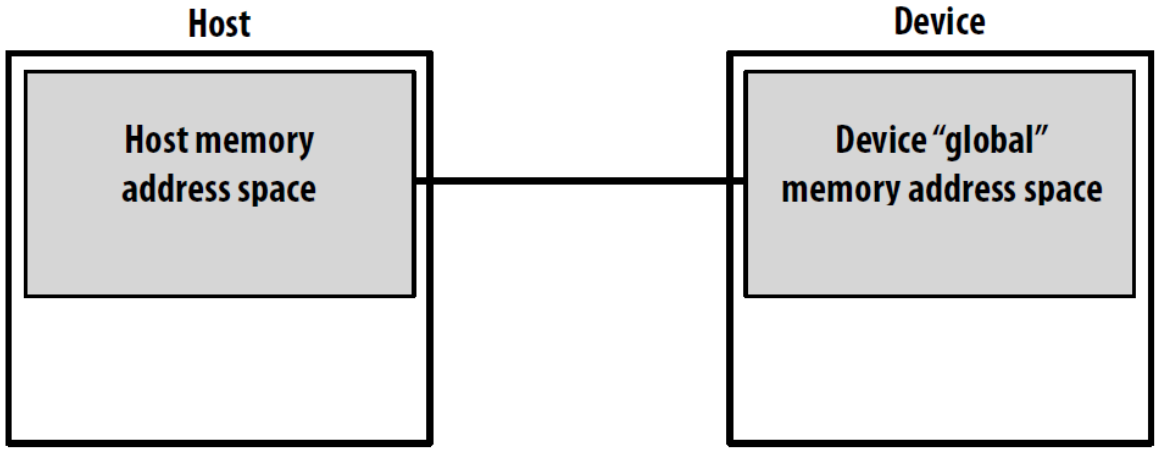
# CUDA Memory Model

- Distinct host and device address spaces



# memcpy Primitive

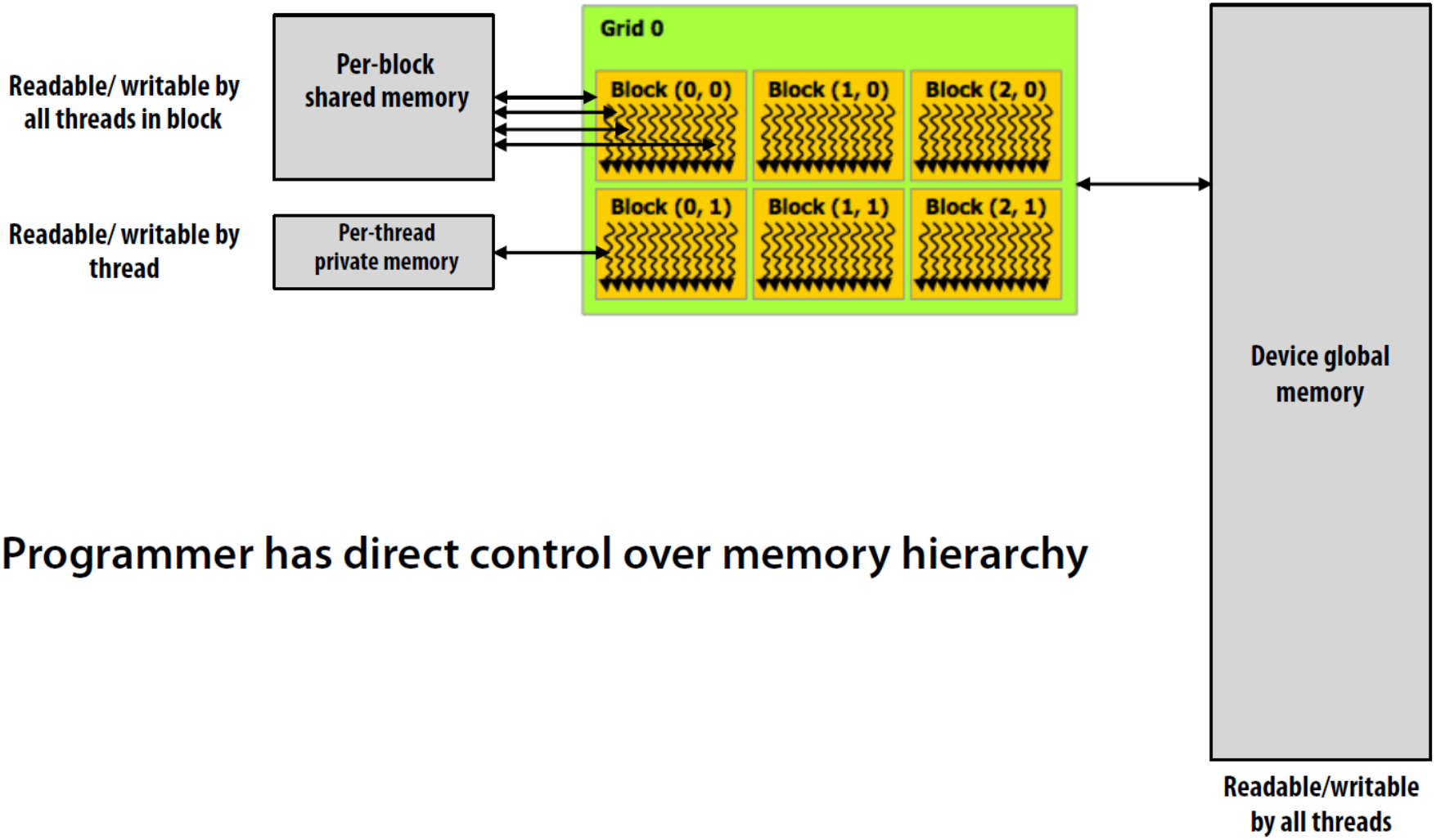
- Move data between address spaces




```
float* A = new float[N];       // allocate buffer in host mem

// populate host address space pointer A
for (int i=0 i<N; i++)
   A[i] = (float)i;

int bytes = sizeof(float) * N
float* deviceA;                 // allocate buffer in
cudaMalloc(&deviceA, bytes);    // device address space

// populate deviceA
cudaMemcpy(deviceA, A, bytes, cudaMemcpyHostToDevice);

// note: deviceA[i] is an invalid operation here (cannot
// manipulate contents of deviceA directly from host.
// Only from device code.)
```

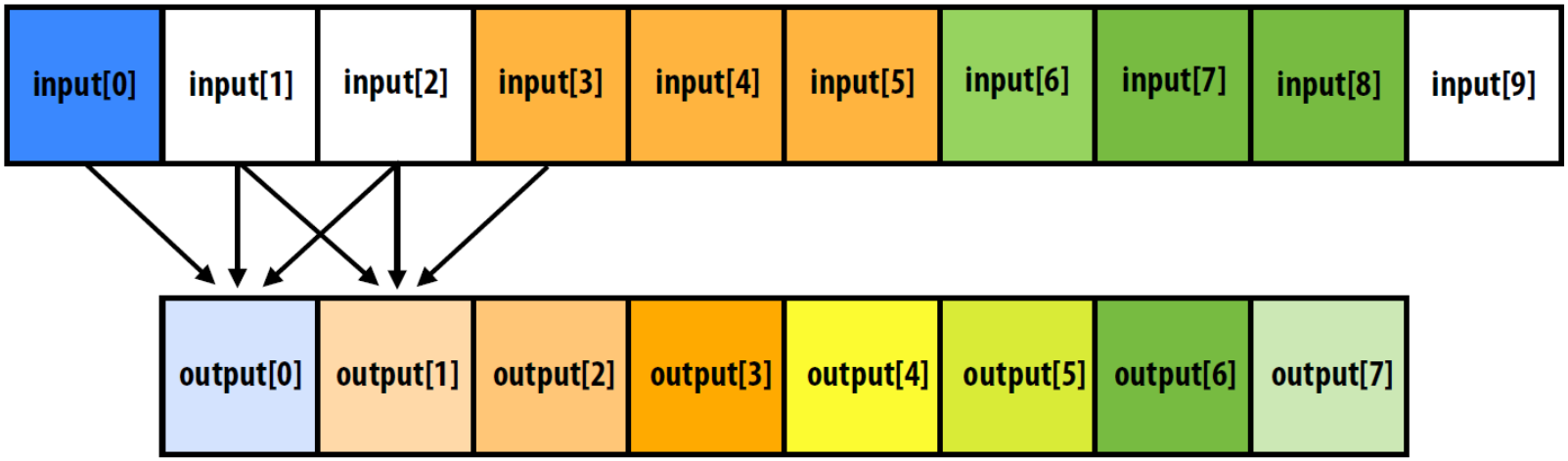
# CUDA device Memory Model

- Three distinct types of memory visible to kernels




**Programmer has direct control over memory hierarchy**

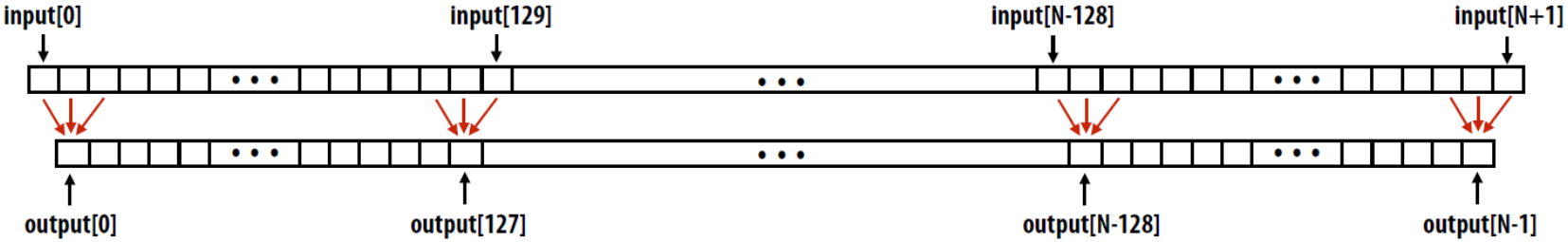
# CUDA Example: 1D Convolution




```
output[i] = (input[i] + input[i+1] + input[i+2]) / 3.f;
```

# 1D Convolution in CUDA

## One thread per output element




**CUDA Kernel**

```
#define THREADS_PER_BLK 128

__global__ void convolve(int N, float* input, float* output) {

   int index = blockIdx.x * blockDim.x + threadIdx.x;  // thread local variable

   float result = 0.0f;  // thread-local variable
   for (int i=0; i<3; i++)
     result += input[index + i];

   output[index] = result / 3.f;
}
```

each thread computes result for one element

write result to global memory

**Host code**

```
int N = 1024 * 1024
cudaMalloc(&devInput, sizeof(float) * (N+2) );  // allocate array in device memory
cudaMalloc(&devOutput, sizeof(float) * N);      // allocate array in device memory

// Initialize contents of devInput here ...

convolve<<<N/THREADS_PER_BLK, THREADS_PER_BLK>>>(N, devInput, devOutput);
```

# CUDA Synchronization Constructs

- **`__syncthreads()`**
  Barrier: wait for all threads in the block to arrive at this point

- **Atomic operations**
  e.g., `float atomicAdd(float* addr, float amount)`
  Atomic operations on both global memory and shared memory variables

- **Host/device synchronization**
  Implicit barrier across all threads at return of kernel

# CUDA Abstractions

- **Execution: thread hierarchy**
  - Bulk launch of many threads
  - Two-level hierarchy: threads are grouped into thread blocks
- **Distributed address space**
  - Built-in memcpy primitives to copy between host and device address spaces
  - Three different types of device address spaces
  - Per thread, per block (“shared”), or per program (“global”)
- **Barrier synchronization primitive for threads in thread block**
- **Atomic primitives for additional synchronization**
  shared and global variables

# Summary

- Multicores, Manycores, accelerators
- Experimental accelerators
- Latest Nvidia GPUs
- CUDA