King Abdullah University of
Science and Technology

KAUST

CS 380 - GPU and GPGPU Programming
Lecture 2: Introduction, Pt. 2

Markus Hadwiger, KAUST

Reading Assignment #1 (until Sep 7)

Read (required):

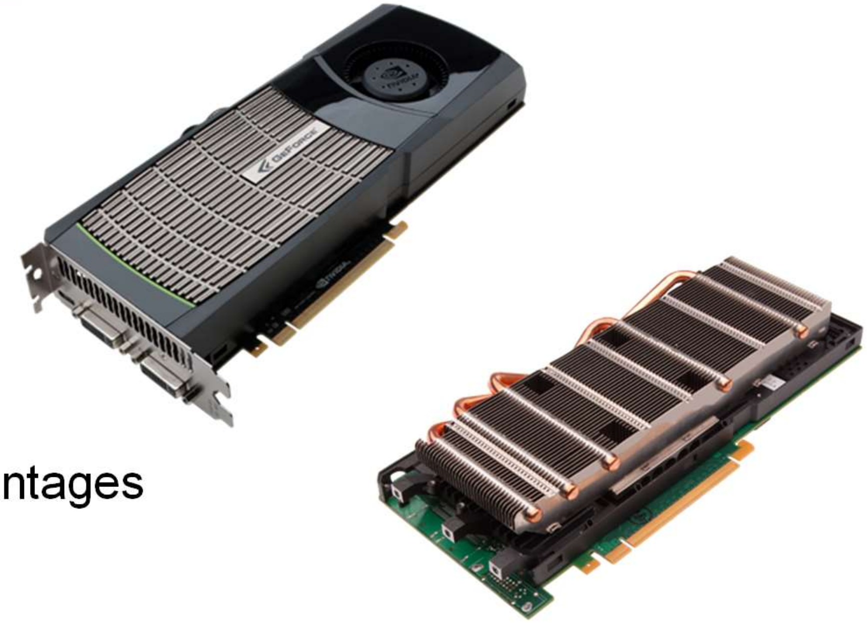
- Orange book, chapter 1 (*Review of OpenGL Basics*)
- Orange book, chapter 2 (*Basics*)

What are GPUs?

Graphics Processing Units

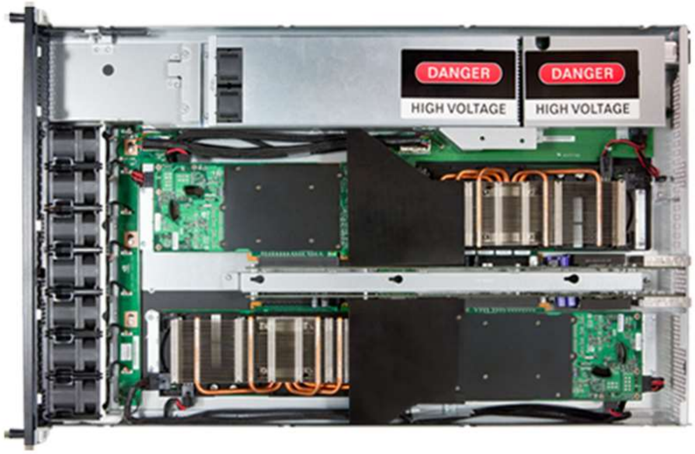
But evolved toward

- Very flexible, massively parallel floating point co-processors
- But not entirely programmable!
- Fixed-function parts have definite advantages (e.g., texture filtering, z-buffering)

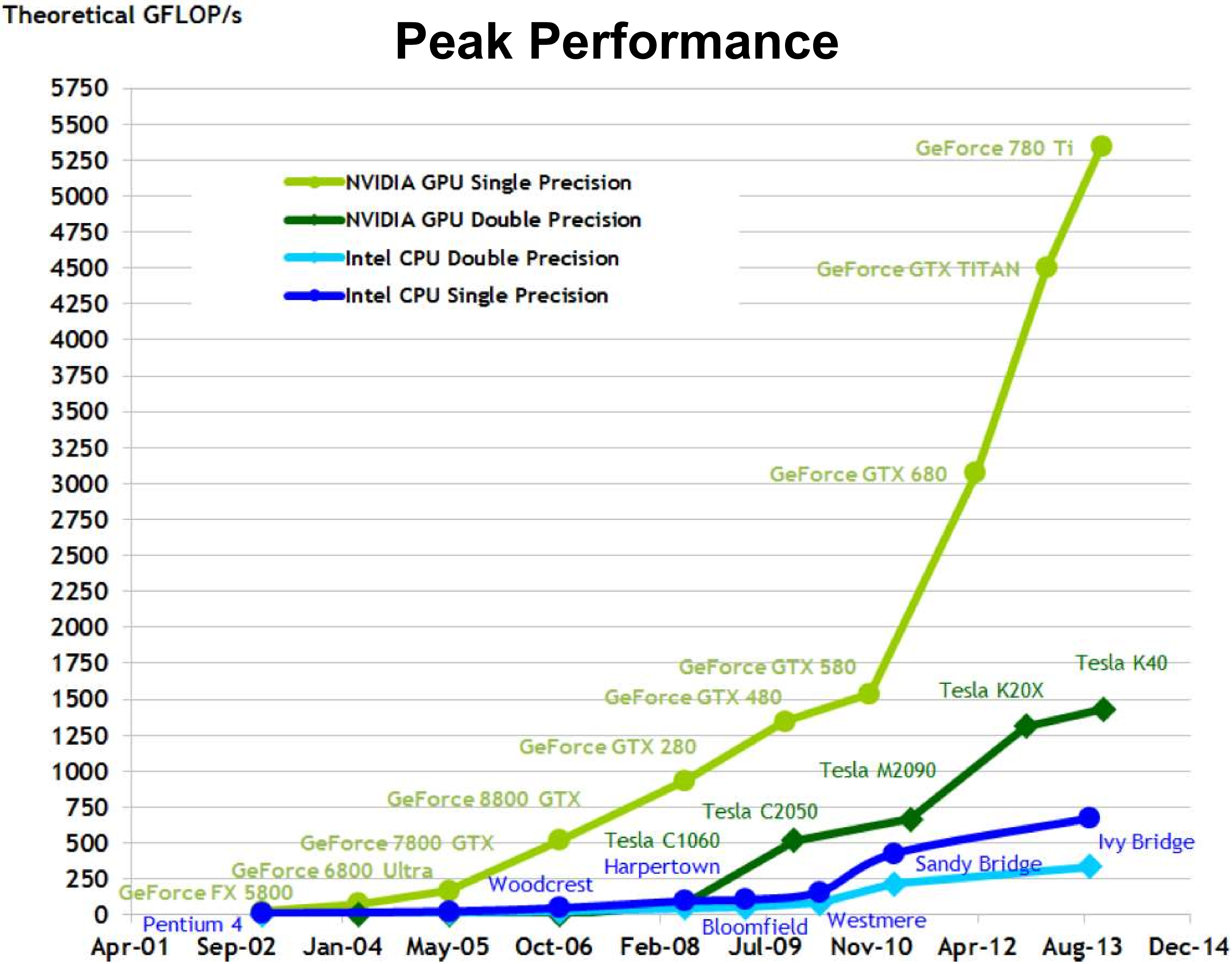


We will cover both perspectives

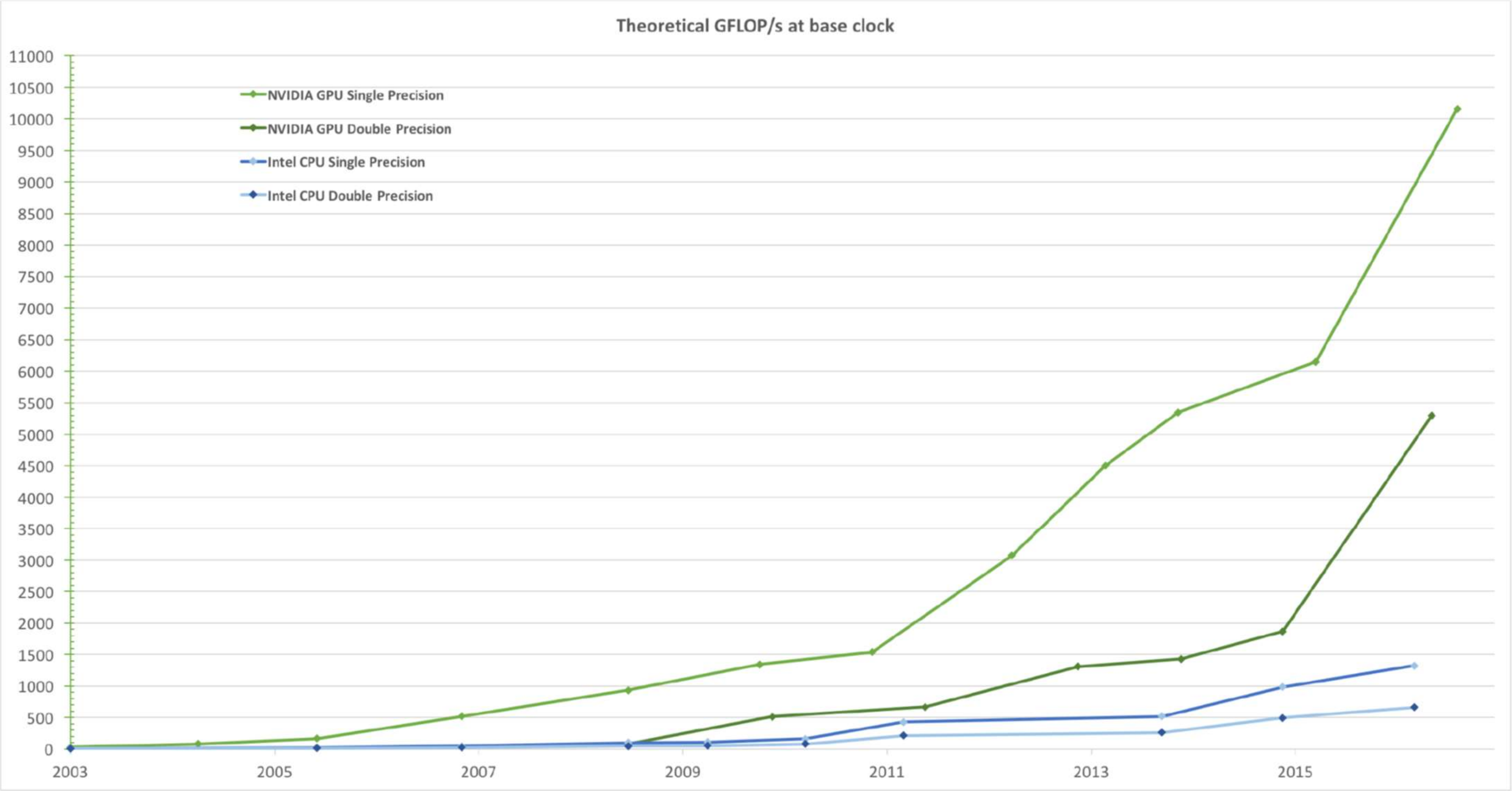
- GPUs for graphics
- GPU computing (GPGPU – general purpose computation on GPU)



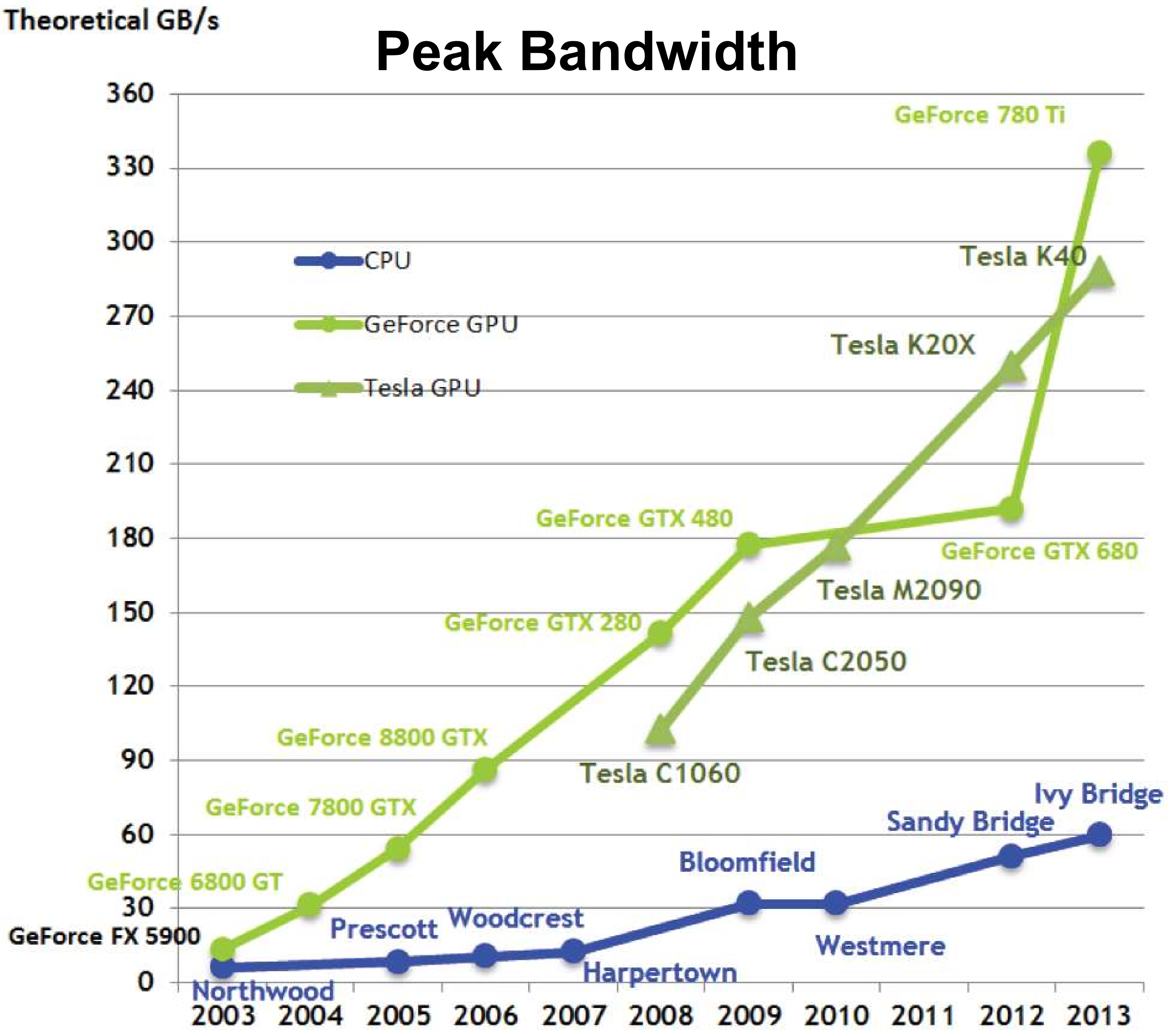
Peak Performance

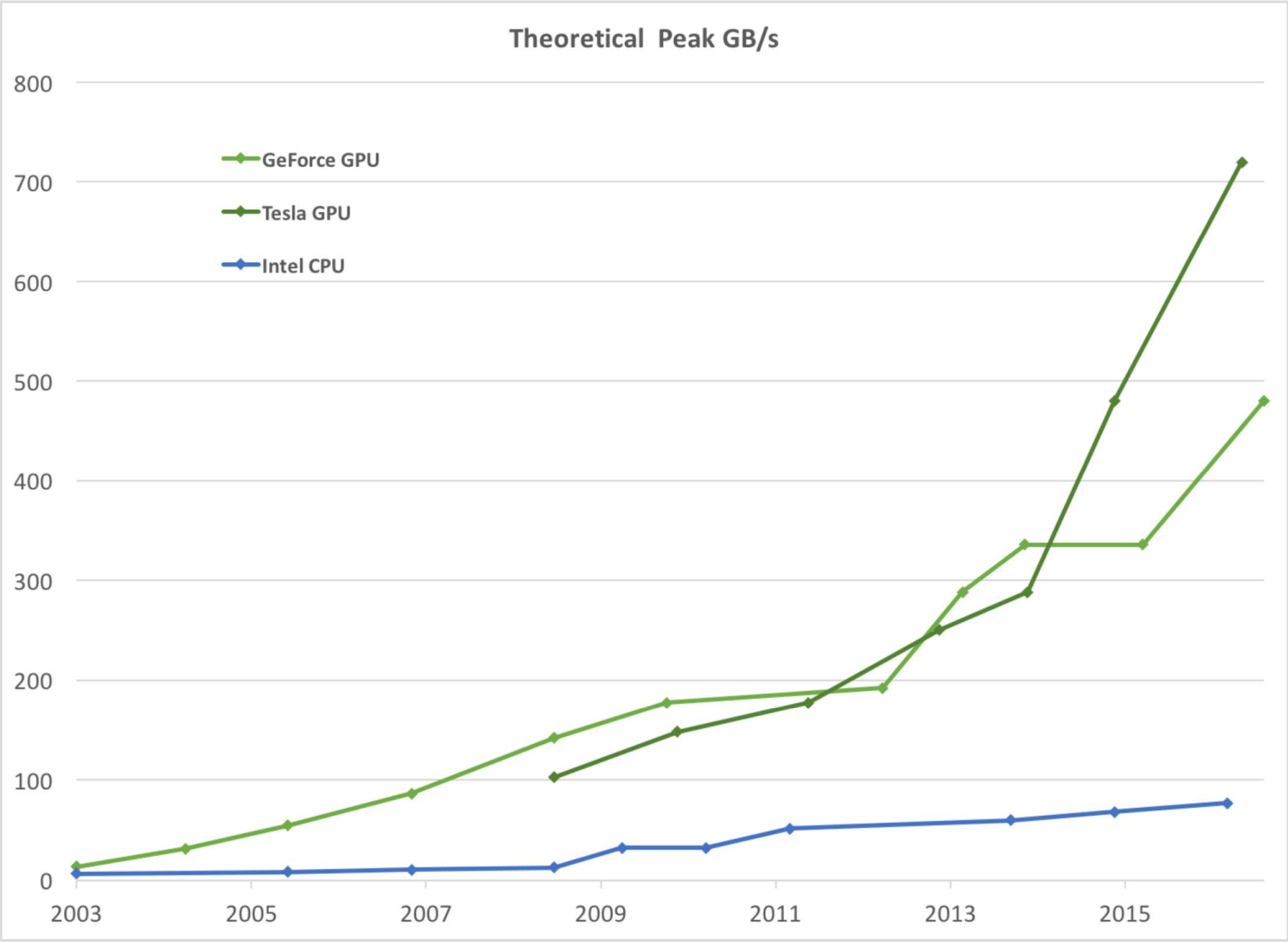


Peak Performance

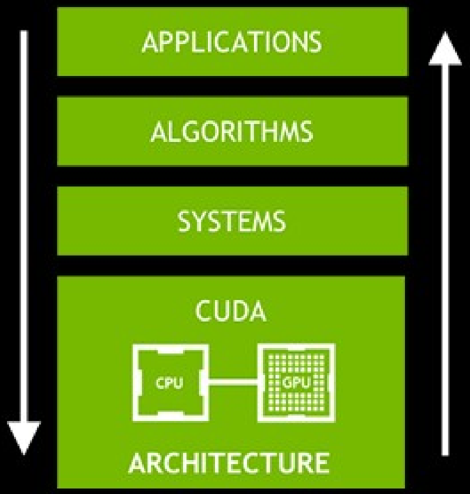


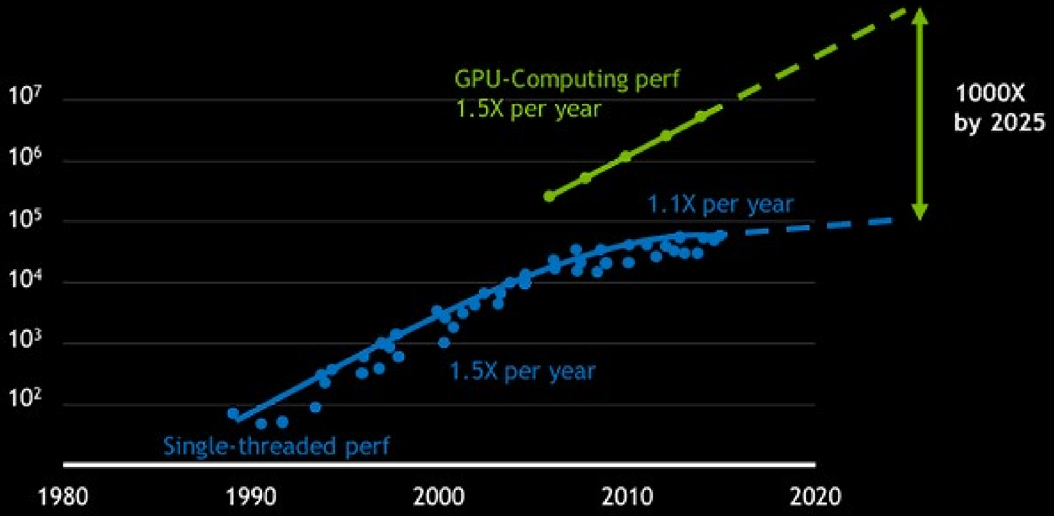
Peak Bandwidth




Theoretical Peak GB/s
800
700
600
500
400
300
200
100
0
2003
2005
2007
2009
2011
2013
2015
GeForce GPU
Tesla GPU
Intel CPU

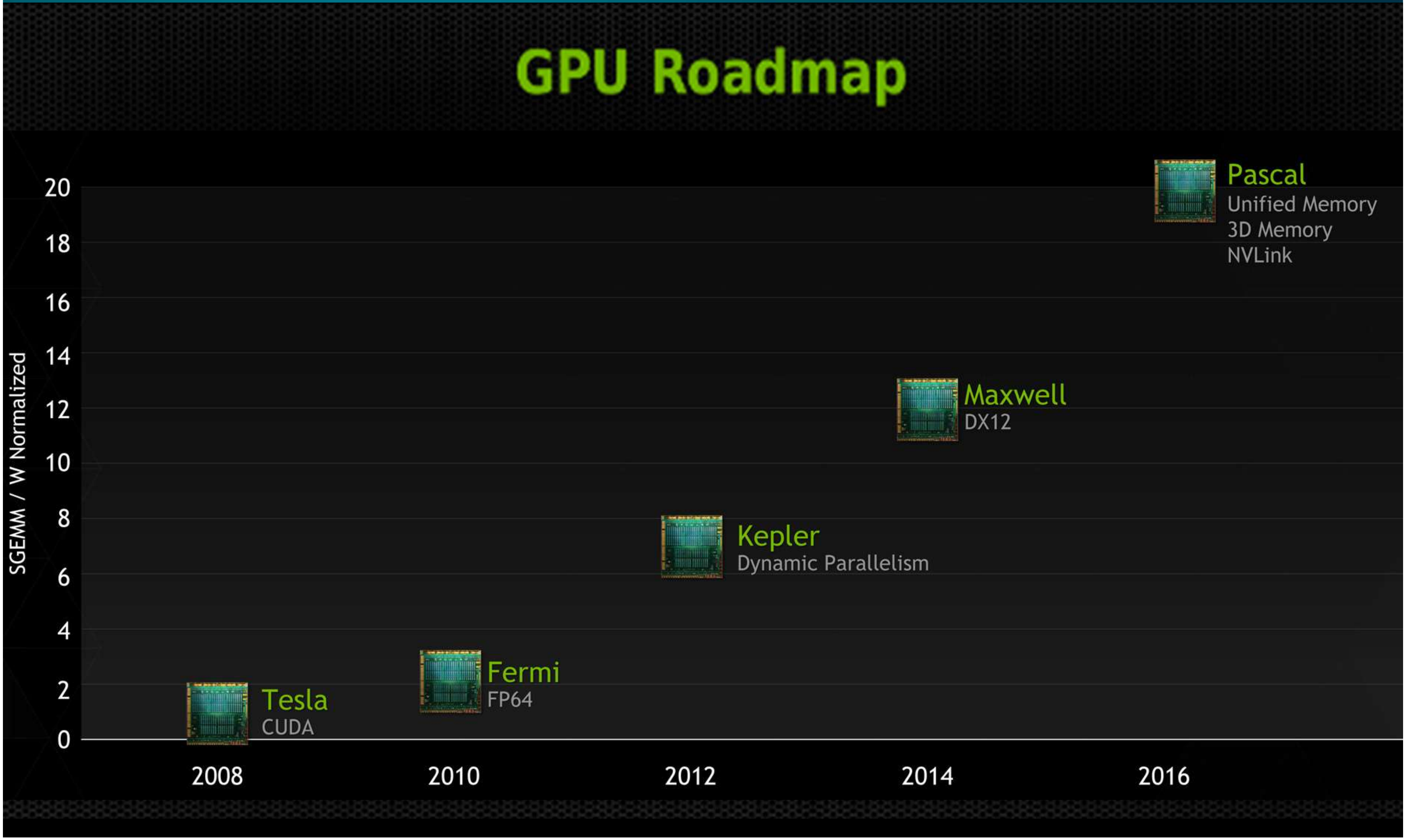
RISE OF GPU COMPUTING



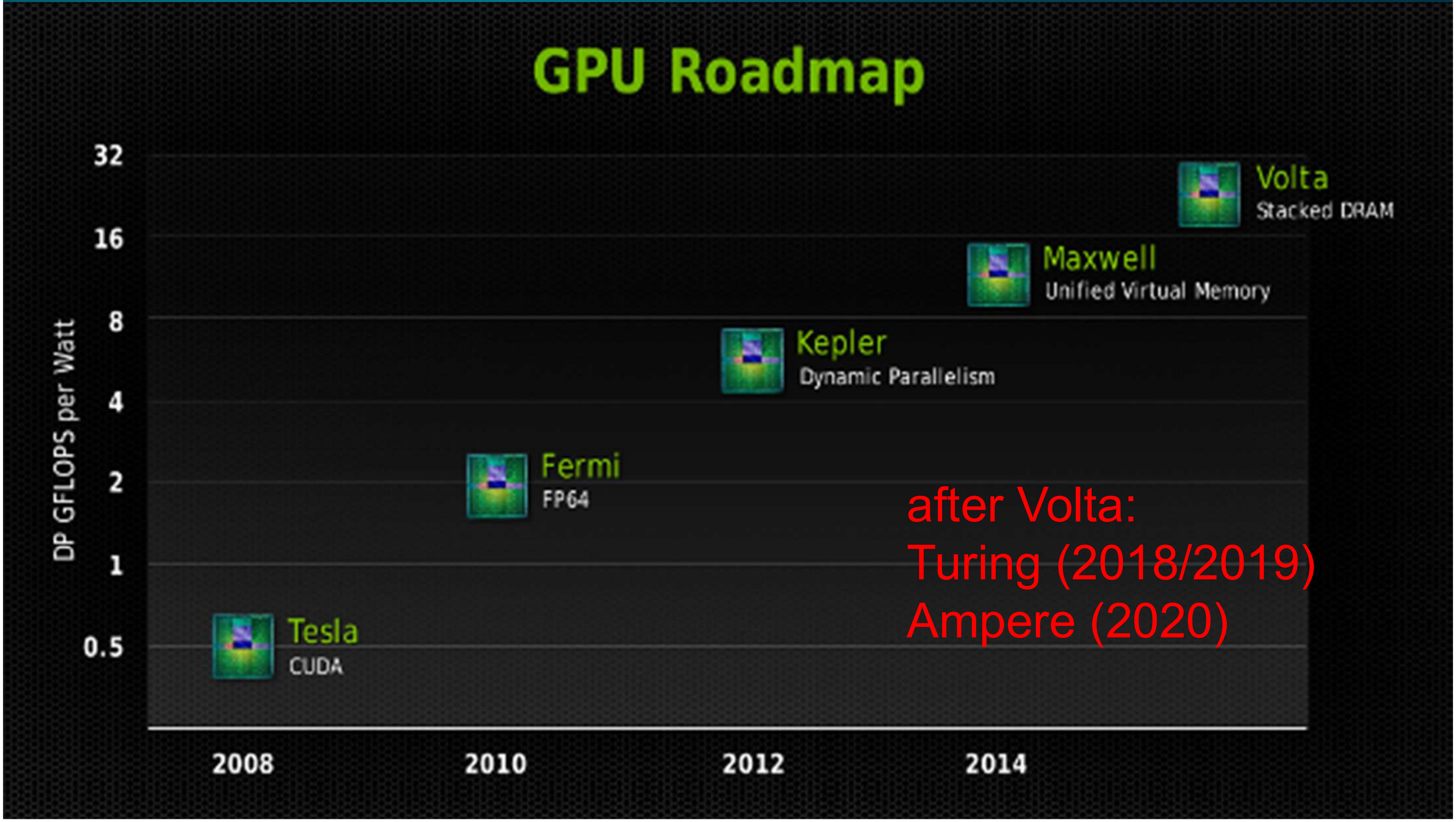


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp

GPU Architectures Over the Years



GPU Architectures Over the Years



after Volta:
Turing (2018/2019)
Ampere (2020)

Recent Updates

NVIDIA Ampere architecture (2020)

`https://en.wikipedia.org/wiki/Ampere_(microarchitecture)`

Promo presentation from Sep 1, 2020:

`https://www.nvidia.com/en-us/geforce/special-event/`

Geforce 30-series (Ampere):

`https://nvidia.com/en-us/geforce/graphics-cards/30-series/`

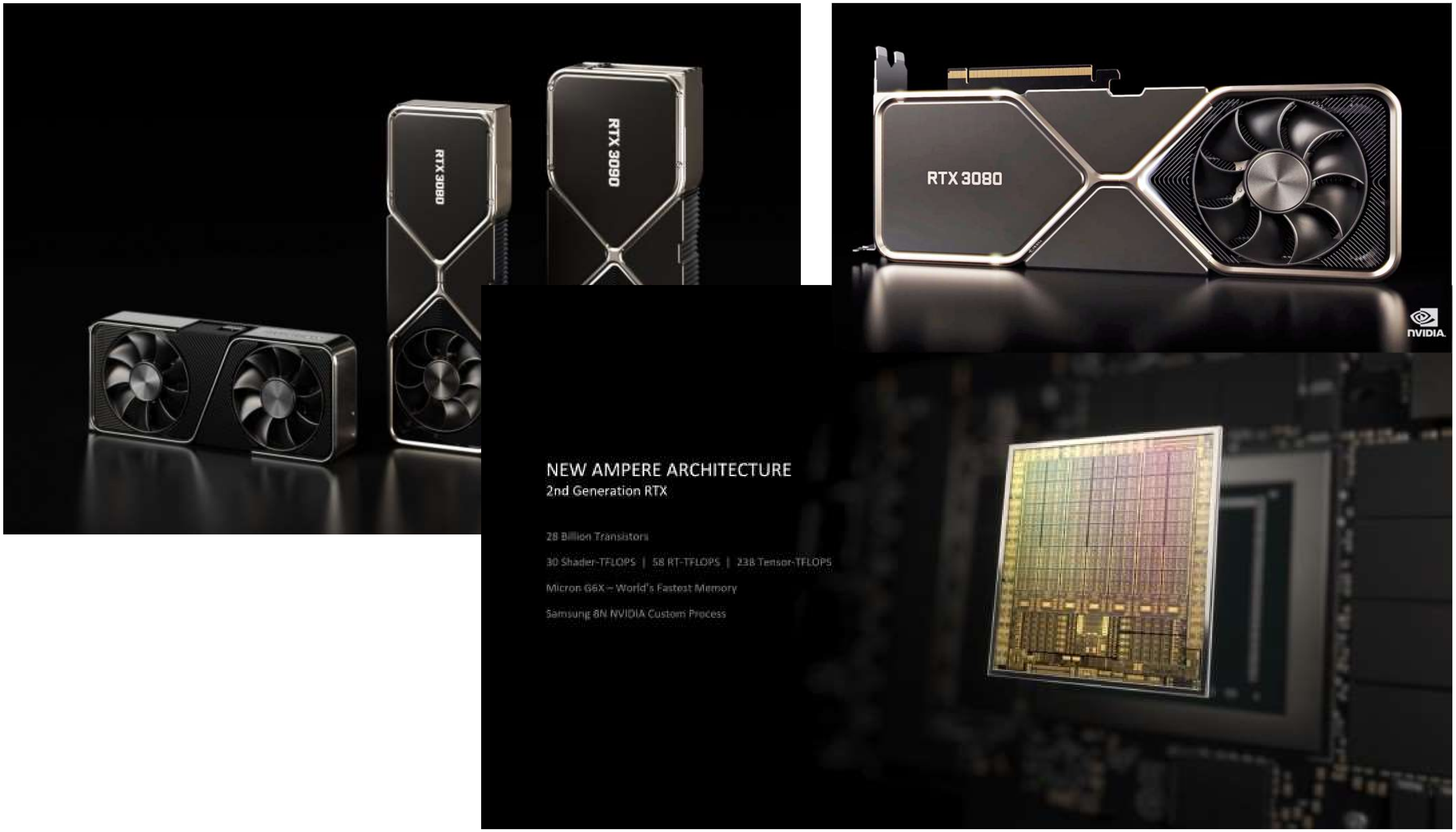
RTX 3090 has 10,496 CUDA cores

A100 (Ampere):

`https://www.nvidia.com/en-us/data-center/a100/`

A100 has 6,912 CUDA cores

Recent Updates



NEW AMPERE ARCHITECTURE
2nd Generation RTX

28 Billion Transistors

30 Shader-TFLOPS | 58 RT-TFLOPS | 238 Tensor-TFLOPS

Micron G6X – World's Fastest Memory

Samsung 8N NVIDIA Custom Process

Overviews and Specs

Wikipedia has many comprehensive lists of architectures and specs

```
https://en.wikipedia.org/wiki/
   List_of_Nvidia_graphics_processing_units
```

```
https://en.wikipedia.org/wiki/
   List_of_AMD_graphics_processing_units
```

What is in a GPU?

Lots of floating point processing power

- Stream processing cores

  different names:
  stream processors,
  CUDA cores, ...

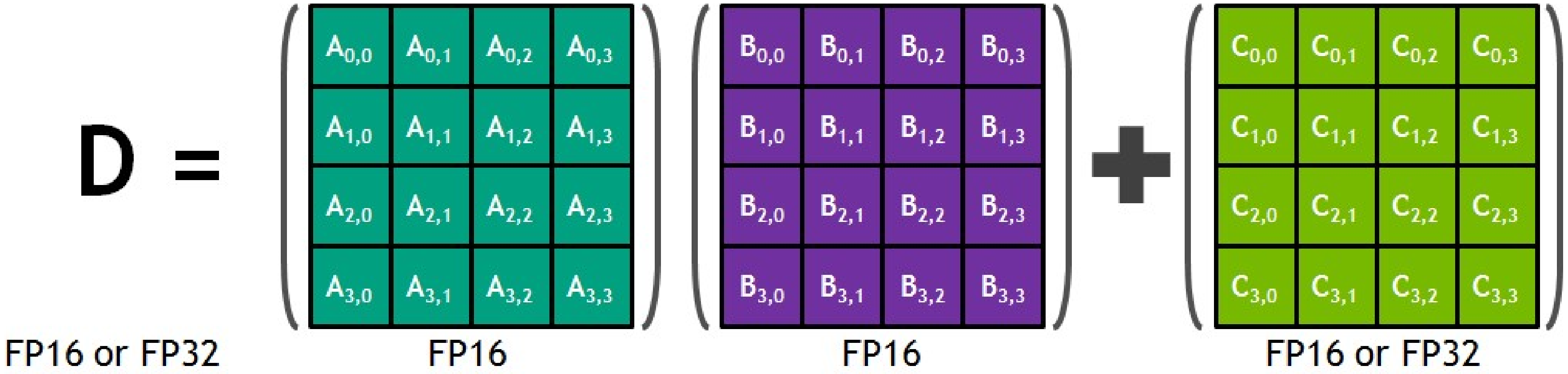
- Was vector processing, now scalar cores!

Still lots of fixed graphics functionality

- Attribute interpolation (per-vertex -> per-fragment)
- Rasterization (turning triangles into fragments/pixels)
- Texture sampling and filtering
- Depth buffering (per-pixel visibility)
- Blending/compositing (semi-transparent geometry, ...)
- Frame buffers

Example for “Special Cores”: Tensor Cores

Mixed-precision, fast matrix-matrix multiply and accumulate



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

From this, build larger sizes, higher dimensionalities, ...

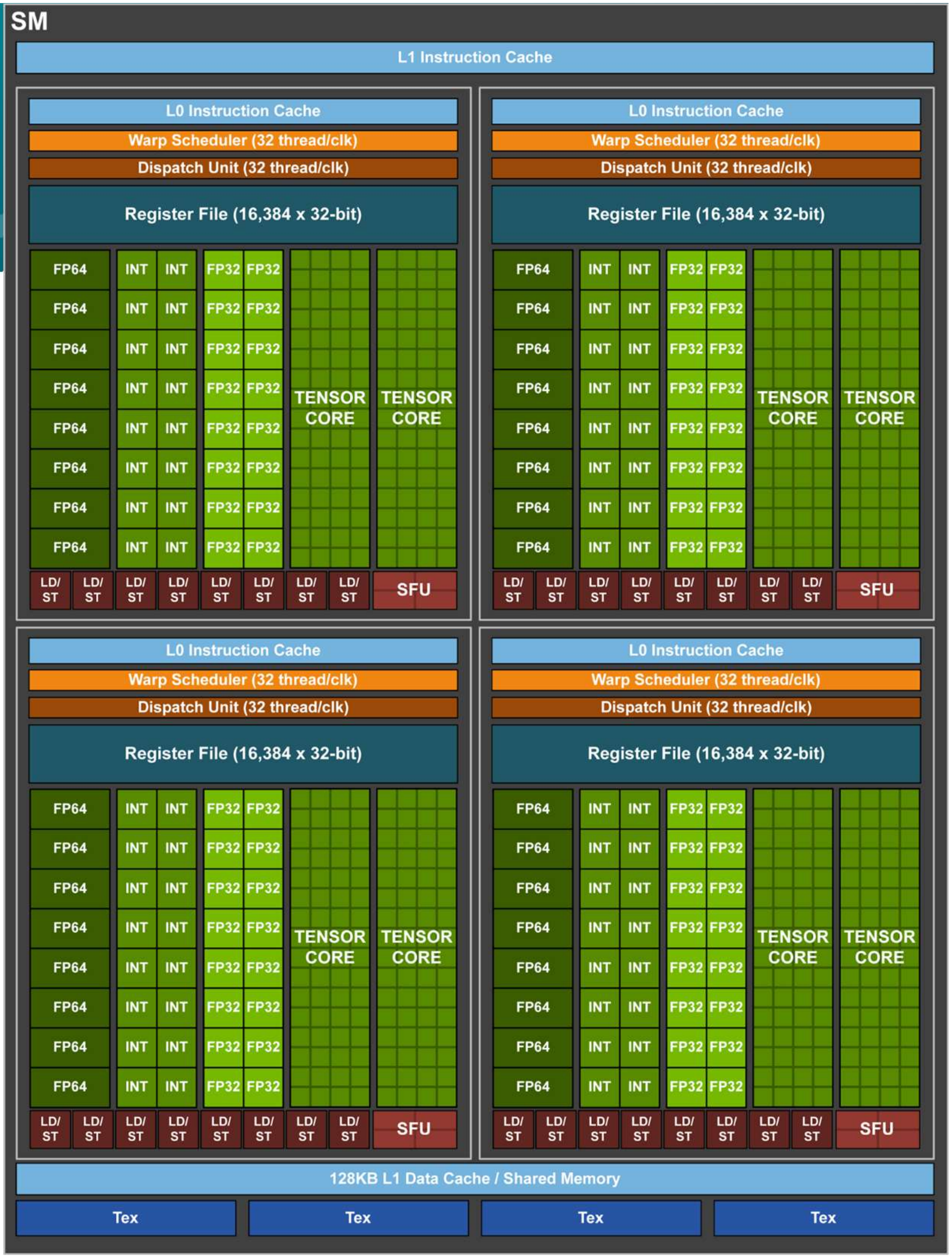
NVIDIA Volta SM

Multiprocessor: SM

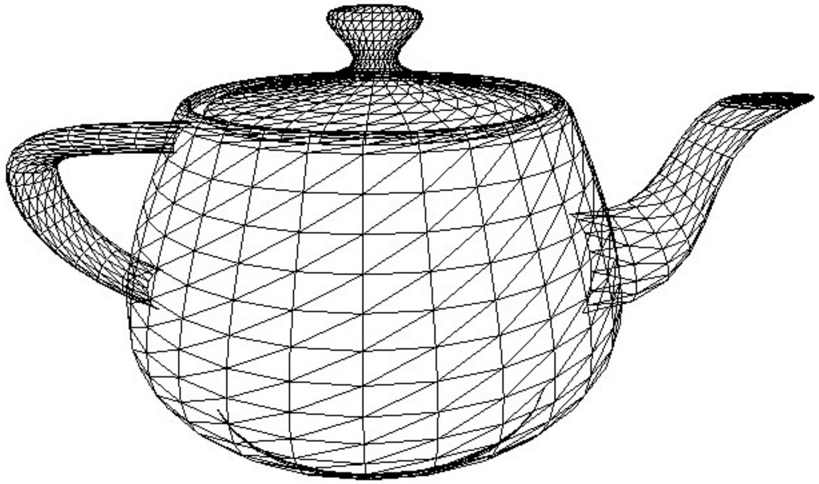
- 64 FP32 + INT32 cores
- 32 FP64 cores
- 8 tensor cores (FP16/FP32 mixed-precision)

4 partitions inside SM

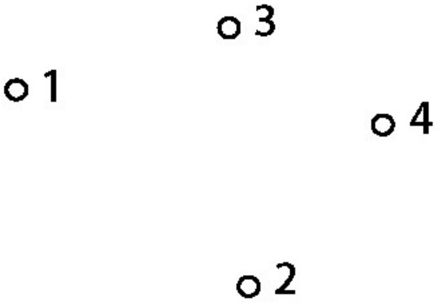
- 16 FP32 + INT32 cores each
- 8 FP64 cores each
- 8 LD/ST units each
- 2 tensor cores each
- Each has: warp scheduler, dispatch unit, register file


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
FP64
INT
FP32
TENSOR CORE
LD/ST
SFU
128KB L1 Data Cache / Shared Memory
Tex

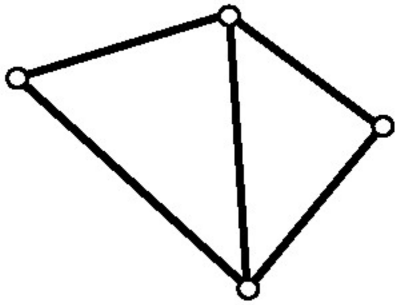
Real-time graphics primitives (entities)



Represent surface as a 3D triangle mesh



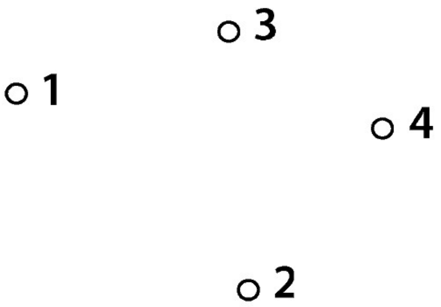
Vertices



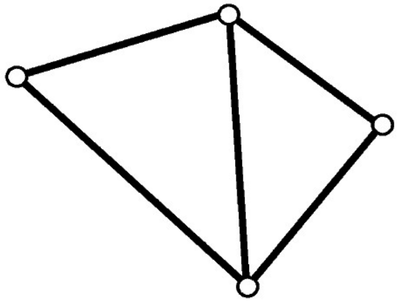
Primitives
(e.g., triangles, points, lines)

Courtesy Kayvon Fatahalian, CMU

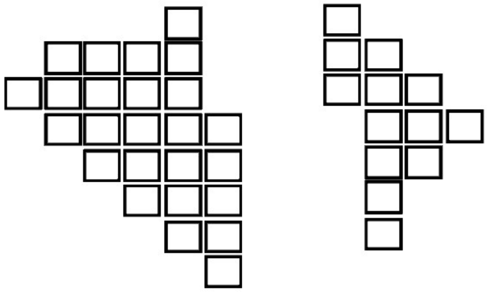
Real-time graphics primitives (entities)



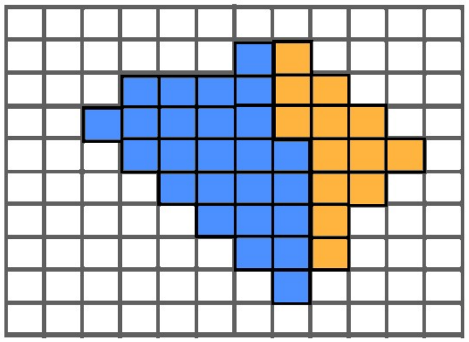
Vertices



Primitives
(e.g., triangles, points, lines)



Fragments

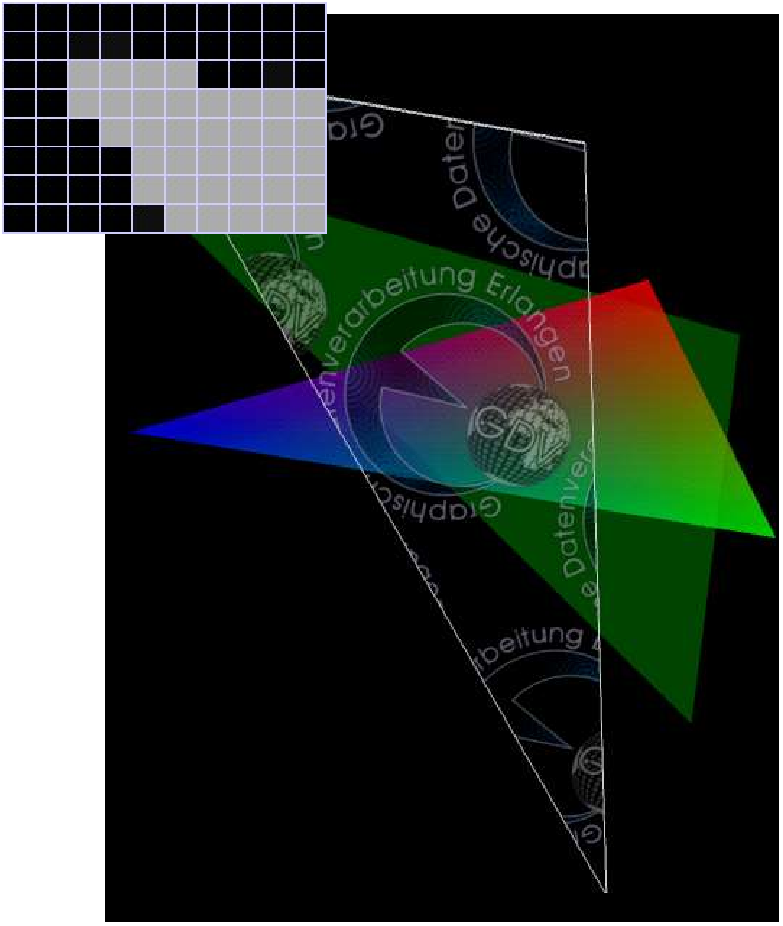


Pixels (in an image)

Courtesy Kayvon Fatahalian, CMU

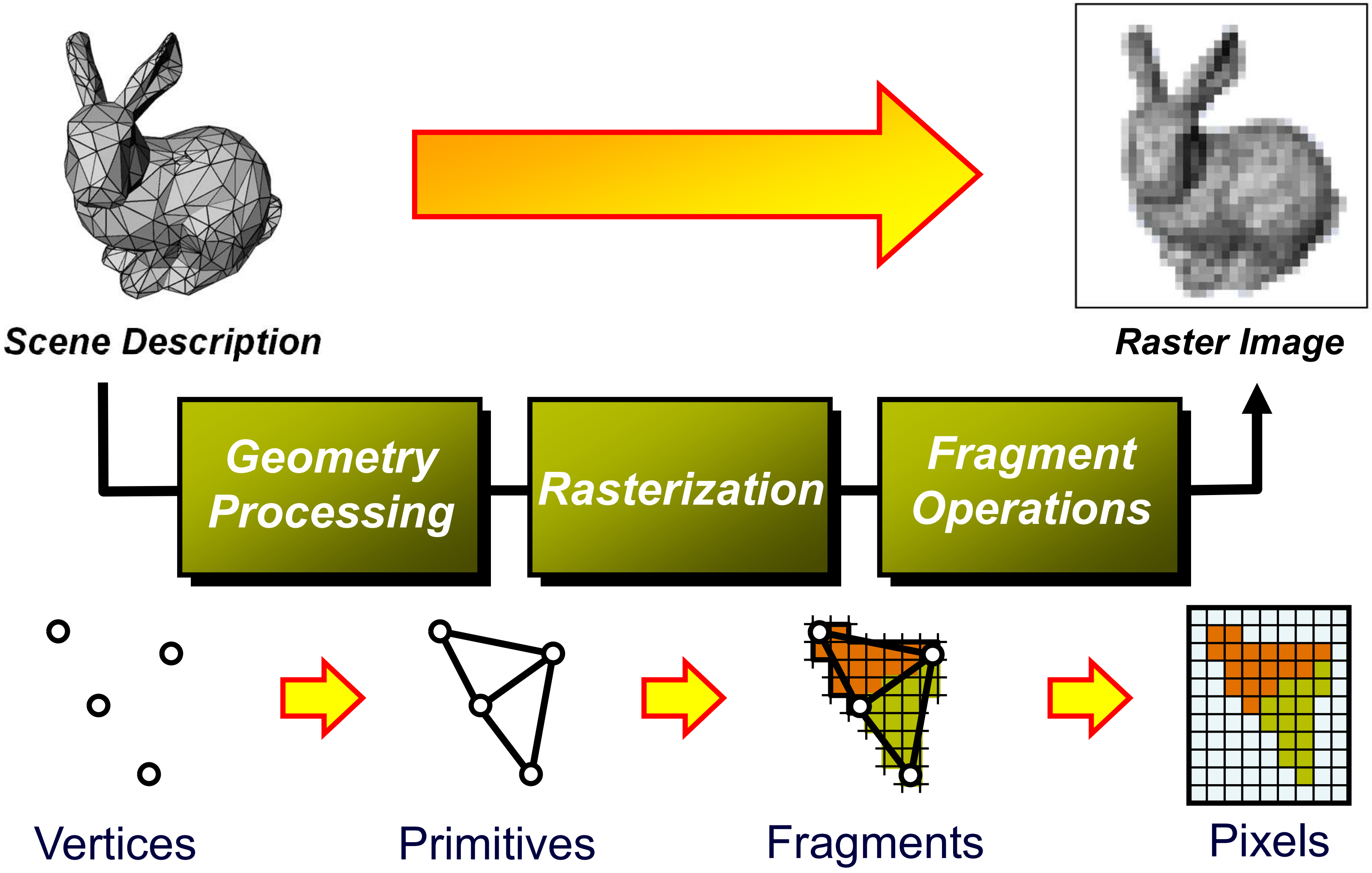
What can the hardware do?

- **Rasterization**
  - Decomposition into fragments
  - Interpolation of color
  - Texturing
    - Interpolation/Filtering
    - Fragment Shading



- **Fragment Operations**
  - Depth Test (Z-Test)
  - Alpha Blending (Compositing)

Graphics Pipeline



Scene Description

Raster Image

Geometry Processing

Rasterization

Fragment Operations

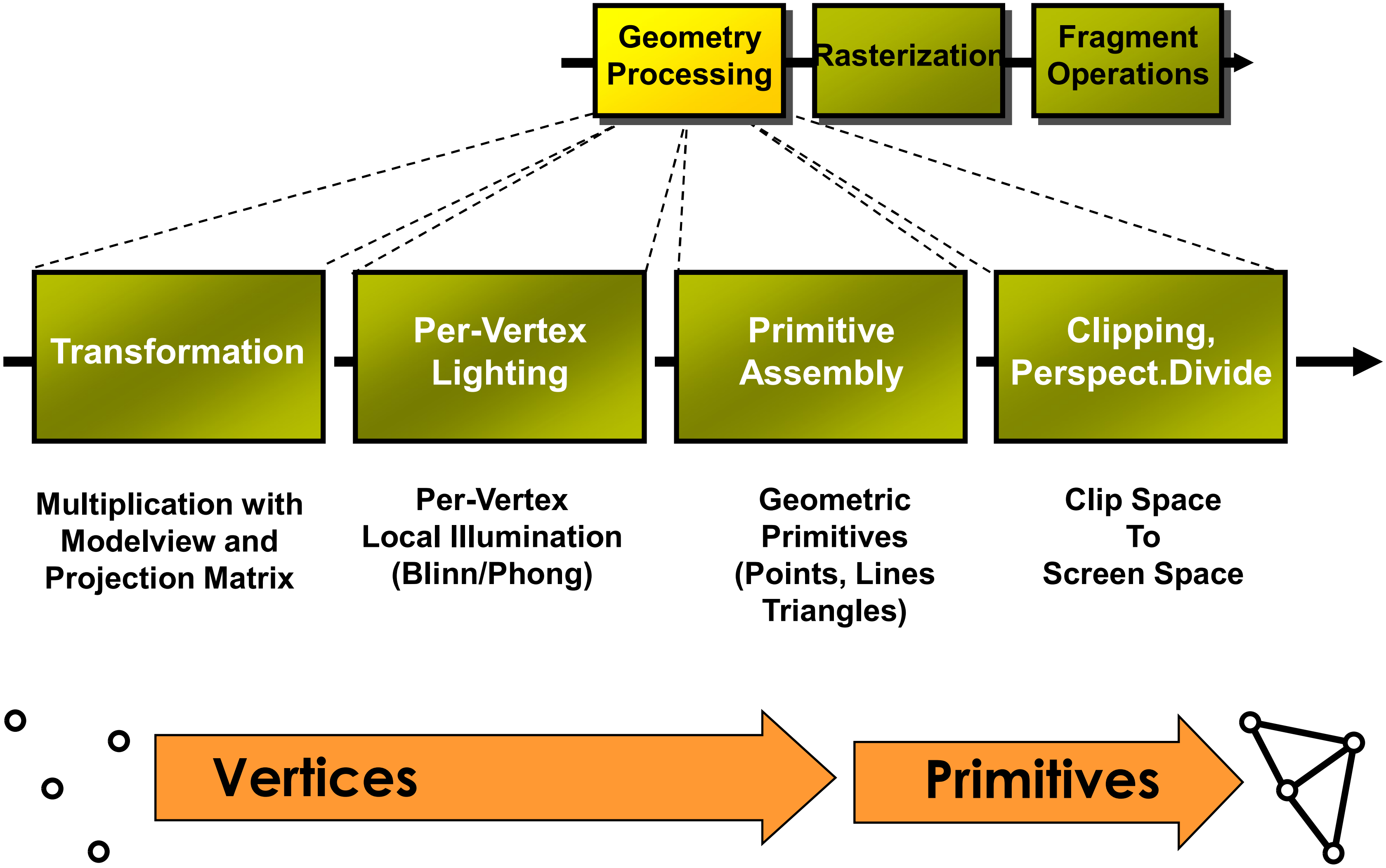
Vertices

Primitives

Fragments

Pixels

Geometry Processing



Geometry Processing

Rasterization

Fragment Operations

Transformation

Per-Vertex Lighting

Primitive Assembly

Clipping, Perspect.Divide

Multiplication with Modelview and Projection Matrix

Per-Vertex Local Illumination (Blinn/Phong)

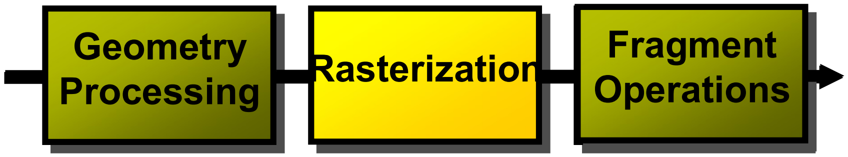
Geometric Primitives (Points, Lines Triangles)

Clip Space To Screen Space

Vertices

Primitives

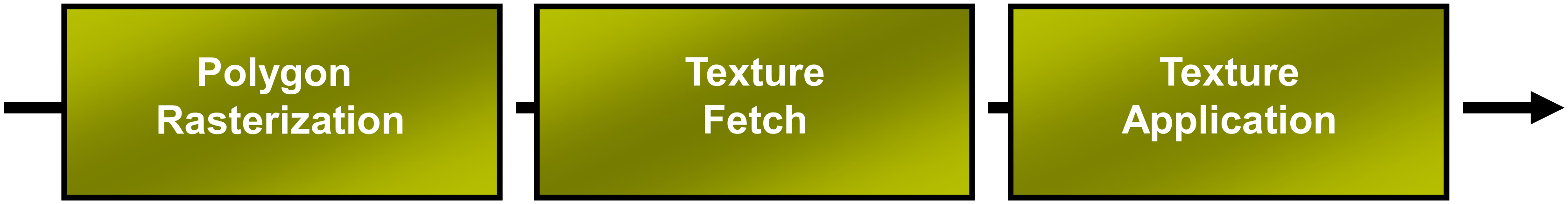
Rasterization



Geometry Processing

Rasterization

Fragment Operations



Polygon Rasterization

Texture Fetch

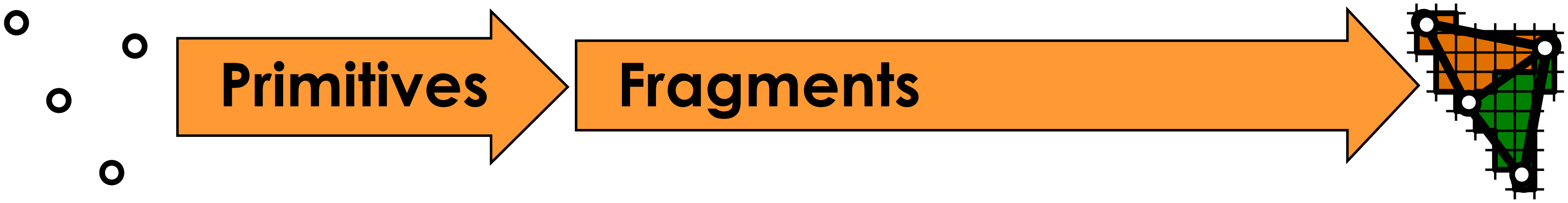
Texture Application

Decomposition of primitives into fragments

Interpolation of texture *coordinates*

Filtering of texture color

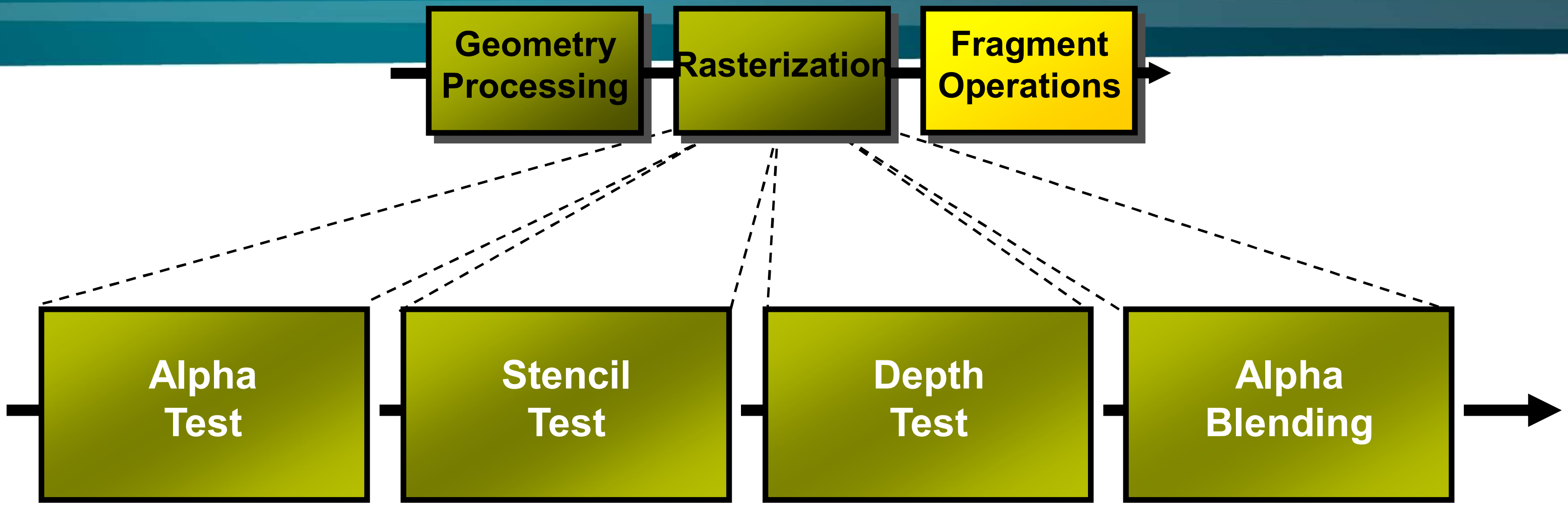
Combination of primary color with texture color



Primitives

Fragments

Fragment Operations



Geometry Processing

Rasterization

Fragment Operations

Alpha Test

Stencil Test

Depth Test

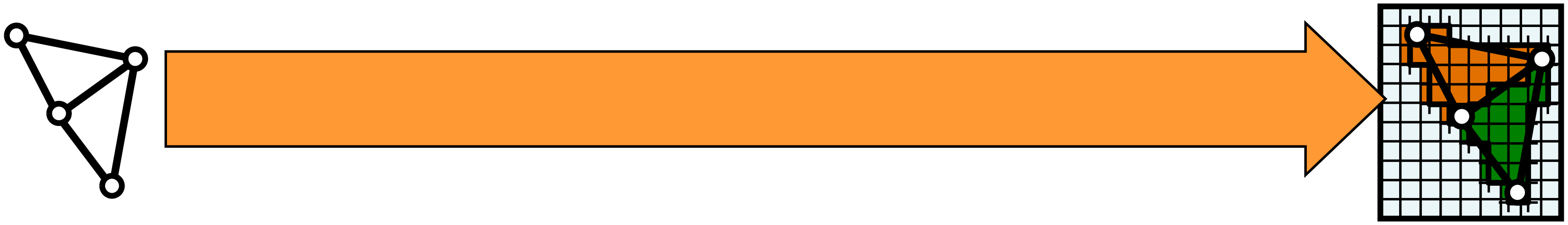
Alpha Blending

Discard all fragments within a certain alpha range

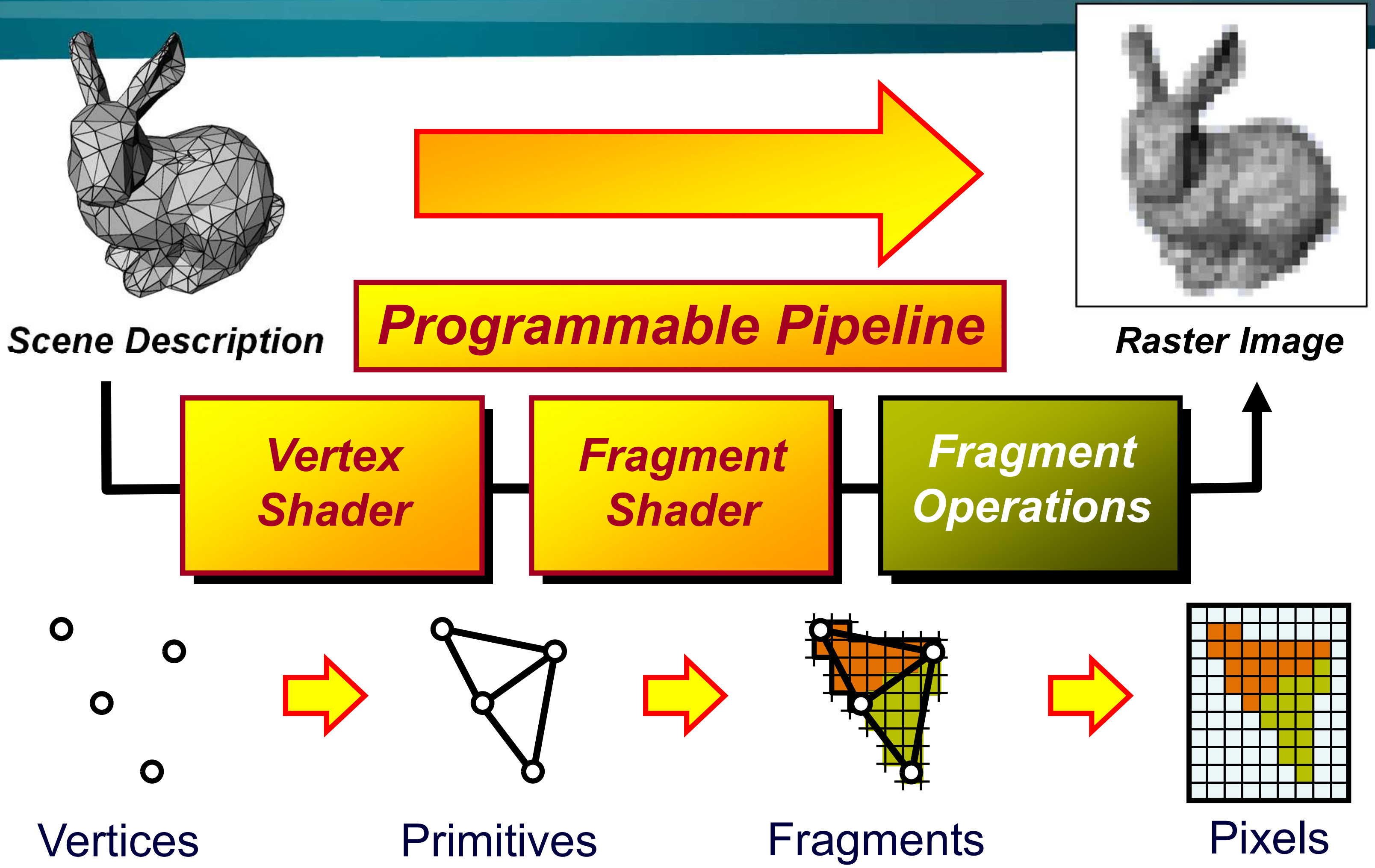
Discard a fragment if the stencil buffer is set

Discard all occluded fragments

Combination of primary color with texture color



Graphics Pipeline



Scene Description

Programmable Pipeline

Raster Image

Vertex Shader

Fragment Shader

Fragment Operations

Vertices

Primitives

Fragments

Pixels

Thank you.