Virtualna okruženja

Igor S. Pandžić, Tomislav Pejša, Mirko Sužnjević

Grafički procesor (GPU)

Efekti (1/3)

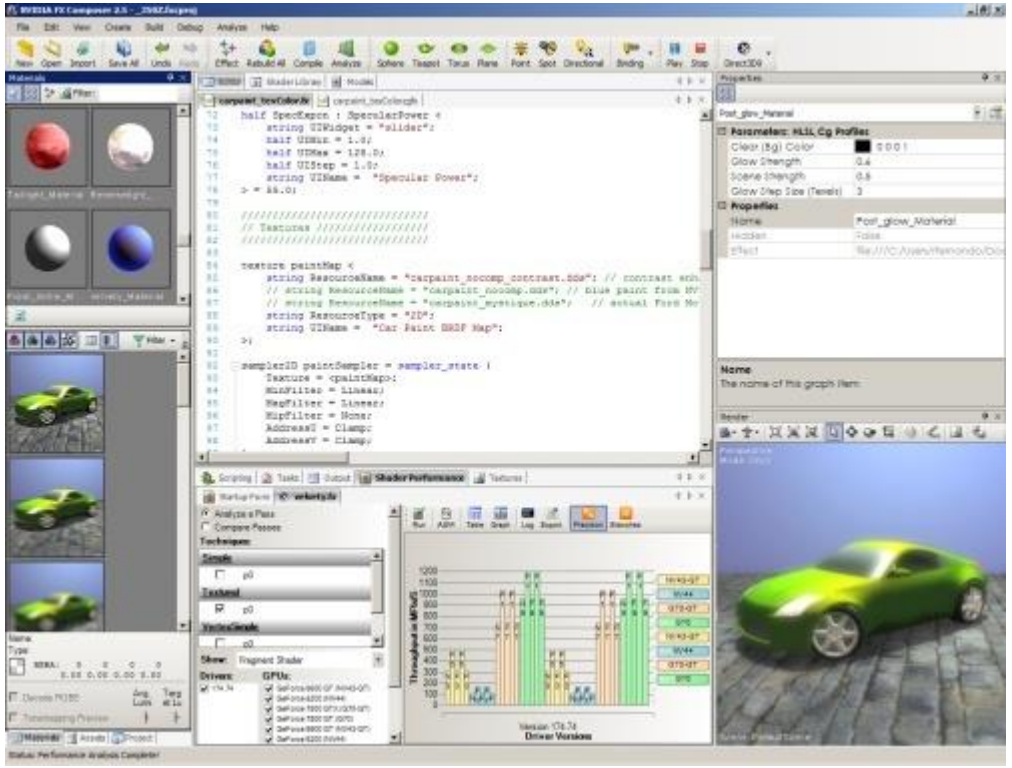
- Pojedini *shader* nije koristan sam za sebe
  - Željeni rezultat dobiva se kombiniranjem više *shadera* i postavki protočnog sustava
- Efekt: skup programa za sjenčanje i svih potrebnih postavki protočnog sustava (npr. uključen/isključen Z-spremnik)
- Zapisuje se u jedinstvenoj datoteci korištenjem jezika za efekte (HLSL FX, CgFX, COLLADA FX)
- Globalni parametri efekta
  - Kroz njih korisnik upravlja svojstvima efekta
  - Moguće ih postavljati kroz sučelje alata za razvoj efekata
  - Npr. efekt Phongovo sjenčanje – parametri materijala, svjetla

Efekti (2/3)

- *Shaderi* su organizirani u *prolaze* (engl. pass)
  - 1 prolaz iscrtavanja kroz grafički protočni sustav
  - Sastoji se od programa za sjenčanje vrhova i točaka (te eventualno geometrije) + postavke protočnog sustava
- Više prolaza čini *tehniku*
  - Slijed prolaza potrebnih za realizaciju efekta
  - Često može biti više tehnika (npr. za različite generacije grafičkog sklopovlja)

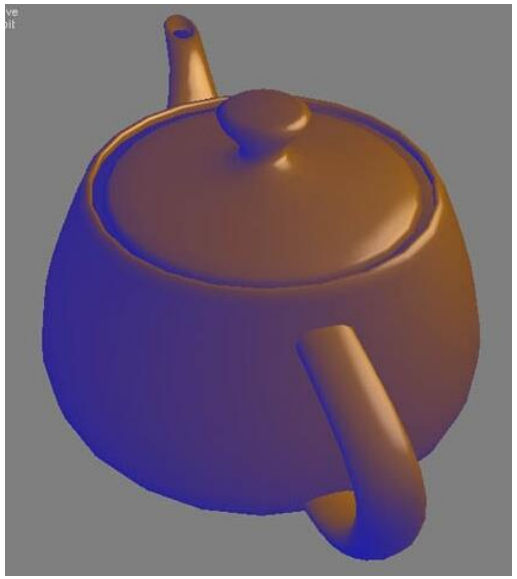
Efekti (3/3)

- Alati za razvoj efekata
  - NVIDIA FX Composer, AMD RenderMonkey



Primjer efekta

- Goochovo sjenčanje
  - Vrsta ne-foto realističnog sjenčanja
- Topli tonovi u smjeru svjetla, hladni od svjetla
  - Osnovni topli i hladni ton definira korisnik
  - Interpolacija boje u ovisnosti o kutu normale prema svjetlu
- Primjer pisan u jeziku HLSL FX



Konstante (uniformni parametri)

```
float4x4 WorldXf        : World
float4x4 WorldITXf     : WorldInverseTranspose
float4x4 WvpXf  : WorldViewProjection
```

- Sintaksa:

```
tip ime : semantika
```

- Semantika
  - Definira značenje parametra
  - Aplikacija prema semantici određuje na koju vrijednost inicijalizirati konstantu

Varijable koje zadaje korisnik

```
float3 Lamp0Pos : Position
<
   string Object = “PointLight0”;
   string UIName = “Lamp 0 Position”;
   string Space = “World”;
> = {-0.5f, 2.0f, 1.25f};
float3 WarmColor
<
   string UIName = “Gooch Warm Tone”;
   string UIWidget = “Color”;
> = {1.3f, 0.9f, 0.15f};
float3 CoolColor <...
```

- Podatke u <> koristi aplikacija, ne sam shader
  - Definiraju kako i pod kojim imenom prikazati parametre u sučelju alata

Program za sjenčanje vrhova (1/2)

- Ulazni parametri procesora vrhova

```
struct vertexIn
{
   float3 Position : POSITION;
   float3 Normal         : NORMAL;
};
```

- Izlazni parametri proc. vrhova / ulaz u proc. Točaka

```
struct vertexOut
{
   float4 HPosition : POSITION;
   float3 LightVec : TEXCOORD1;
   float3 WorldNormal : TEXCOORD2;
};
```

- Semantika ima analogan smisao – govori protočnom sustavu koji podatak primitiva zapisati u koju varijablu
- Semantika TEXCOORD ostala iz povijesnih razloga

Program za sjenčanje vrhova (1/2)

```
vertexOut std_VS( vertexIn IN )
{
   vertexOut OUT;
   float4 No = float4(IN.Normal,0);
       OUT.WorldNormal = mul(No,WorldITXf).xyz;
   float4 Po = float4(IN.Position,1);
   float4 Pw = mul(Po,WorldXf);
   OUT.LightVec = (Lamp0Pos - Pw.xyz);
   OUT.HPosition = mul(Po,WvpXf);
   return OUT;
}
```

- Računa normalu i vektor svjetla u globalnom koordinatnom sustavu
- Transformira vrh u prostor pogleda i projicira ga

Program za sjenčanje točaka

```
float4 gooch_PS(vertexOutput IN)
{
    float3 Ln = normalize(IN.LightVec);
    float3 Nn = normalize(IN.WorldNormal);
    float ldn = dot(Ln,Nn);
    float mixer = 0.5 * (ldn + 1.0);
    float4 result = lerp(CoolColor, WarmColor, mixer);
    return result;
}
```

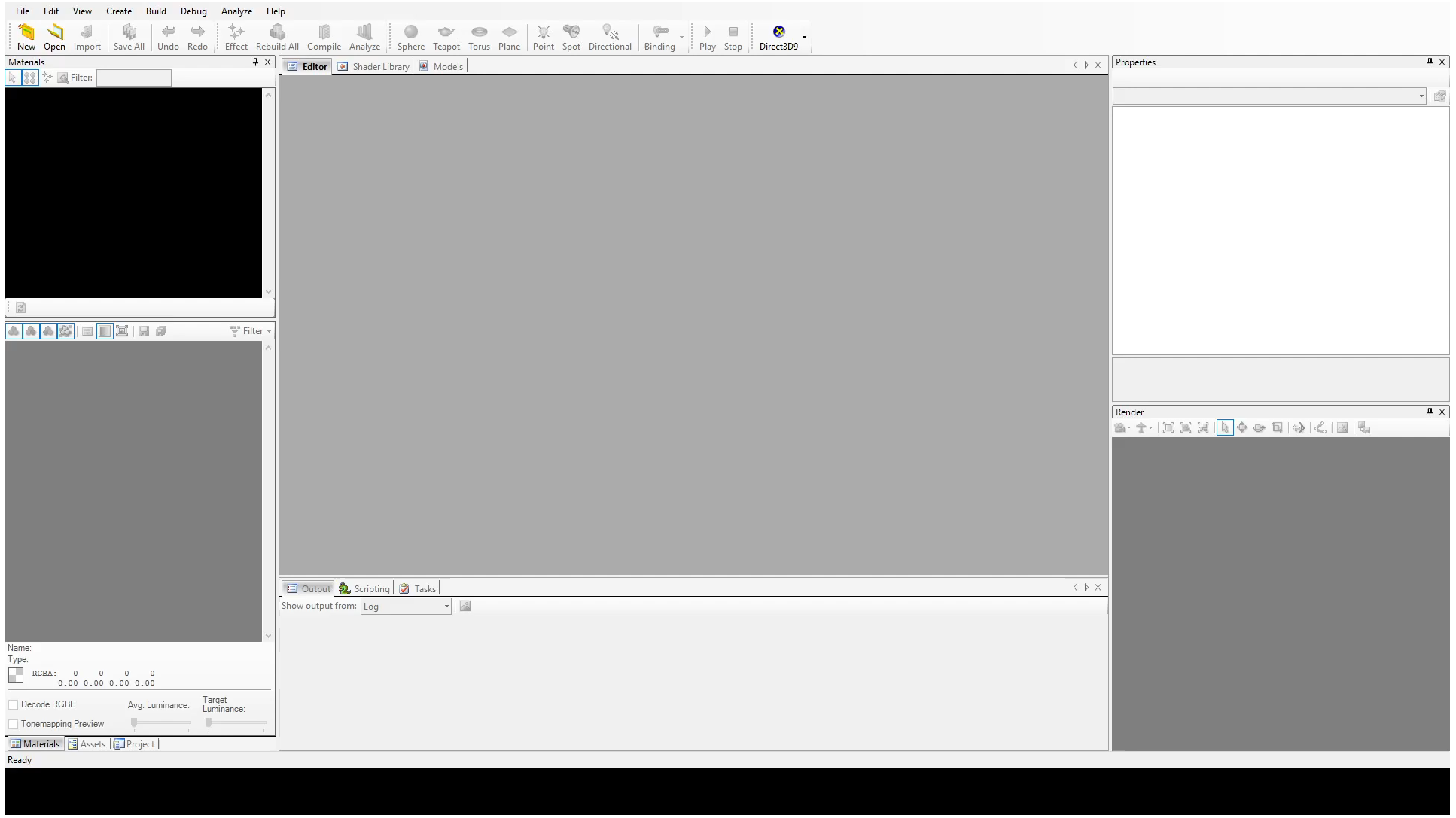
- Interpolira između toplog i hladnog tona u ovisnosti o kutu između normale i smjera svjetla

Definicija tehnike

```
technique Gooch < string Script = “Pass=p0;” >
{
   pass p0 <string Script = “Draw=geometry;” >
   {
      VertexShader = compile vs_2_0 std_VS();
        PixelShader = compile ps_2_a gooch_PS();
      ZEnable = true;
      ZWriteEnable = true;
      ZFunc = LessEqual;
        AlphaBlendEnable = false;
   }
}
```

- Naš efekt ima jednu tehniku, koja ima jedan prolaz
- Standardna uporaba Z-spremnika, prozirnosti nema

Primjer efekta



Kombiniranje više svjetala i materijala

- *Shaderi* nam omogućuju proizvoljne kombinacije broja i tipova svjetala, materijala i jednadžbi sjenčanja
- Kombinacije mogu varirati i unutar iste scene
- Kako ostvariti sve te kombinacije unutar programa za sjenčanje?
  1. Dinamičko grananje
  2. Übershader
  3. Višeprolazno osvjetljenje
  4. Odgođeno sjenčanje
- Definicije:
  - *M* tipova svjetala
  - *N* tipova materijala
  - *L* svjetala po predmetu

Dinamičko grananje

```
float4 my_shader(...)
{
   float4 color;
   for( i = 0; i < L; ++i )
   {
      color += light_compute( light[i],
         objMaterial );
   }
   return color;
}
```

Übershader

- Zaseban *shader* za svaku kombinaciju
- Broj kombinacija: $\frac{(M+L)!}{L!M!} \times N$
- Primjer: Half-Life 2 (2004.) – 1920 kombinacija!
- Nije praktično *shader* za svaku varijantu pisati zasebno
- Piše se jedan složeni *übershader*:
  - Preprocesorske naredbe za razne kombinacije
  - Višestruko prevođenje – različite varijante *shadera*
- Nepraktično za umjetnike

Višeprolazno osvjetljenje

- Zaseban prolaz iscrtavanja za svaki izvor svjetlosti
- Aditivno miješanje rezultata u spremnik boja u fazi stapanja
- Broj *shadera*: MxN
- *Shaderi* su jednostavni, ali performanse su slabije (aditivno miješanje, ponavljanje proračuna)

Odgođeno sjenčanje (1/3)

- Sjenčanje se radi *nakon* određivanja vidljivosti Z-spremnikom
- 1. prolaz:
  - Iscrta se scena bez sjenčanja, ali uz uključen Z-spremnik
  - MRT – podaci vidljive geometrije u svakoj točki se zapisuju u *G-spremnike* (dubine, normale, teksturne koordinate)
- Daljnji prolazi:
  - Iscrta se jedan pravokutnik koji prekriva cijeli zaslon, a na njega se „lijepi” slika scene
  - U procesoru točaka – uzorkuju se G-spremnici i obavlja sjenčanje

Odgođeno sjenčanje (2/3)

- Odlične performanse (sjenčaju se samo vidljivi fragmenti)
- Gotovo neograničen broj svjetala – puno „življe” i realnije scene
- Broj *shadera*: M+N ; *shaderi* za materijal (1. prolaz) odvojeni od *shadera* za svjetla (2. prolaz) – olakšan posao umjetnicima
- Nedostaci:
  - Velike potrebe za memorijom
  - Nema MSAA (no moguće implementirati u *pixel shaderu*)

Odgođeno sjenčanje (3/3)

- Primjer: S.T.A.L.K.E.R. (2007.)

GPGPU (1/2)

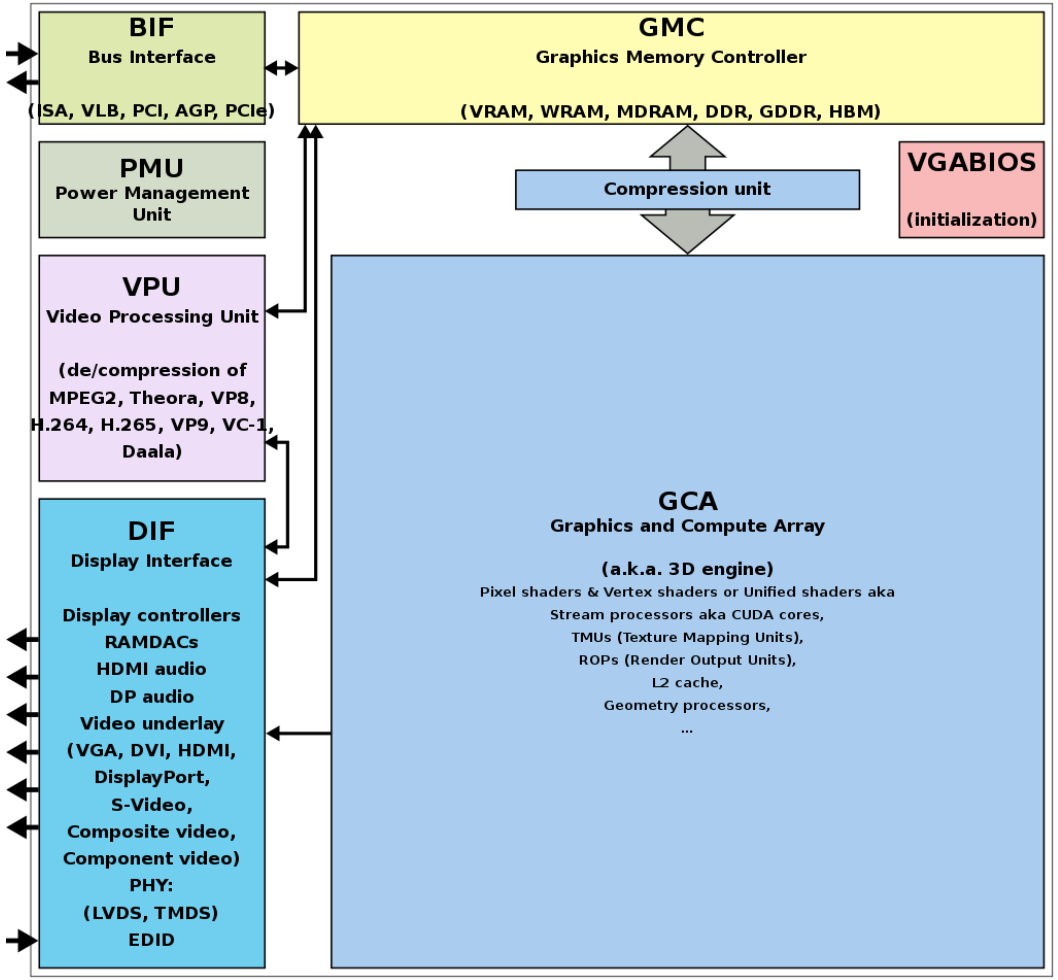
- Općeniti proračuni na grafičkom procesoru
- Strujna obrada podataka (engl. stream processing)
  - Struja – podaci; jezgre – operacije na podacima
  - Paralelna, nezavisna obrada velikog broja struja
  - Iscrtavanje je primjer struje obrade podataka!
- GPGPU sustavi i jezici:
  - NVIDIA CUDA
  - AMD APP SDK
  - OpenCL
  - DirectCompute

GPGPU (2/2)

- Danas – dedicirano grafičko sklopovlje za GPGPU
- Primjene:
  - Fizikalne simulacije
  - Ray tracing
  - Obrada slike, zvuka i videa
  - Kriptografija i kriptoanaliza
  - Baze podataka
  - Neuronske mreže
  - Linearna algebra
  - Sortiranje i pretraživanje

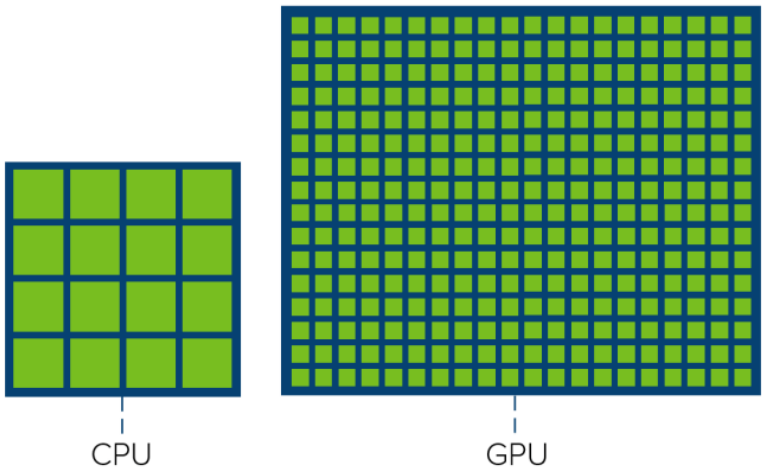
Grafičko sklopovlje – generička arhitektura

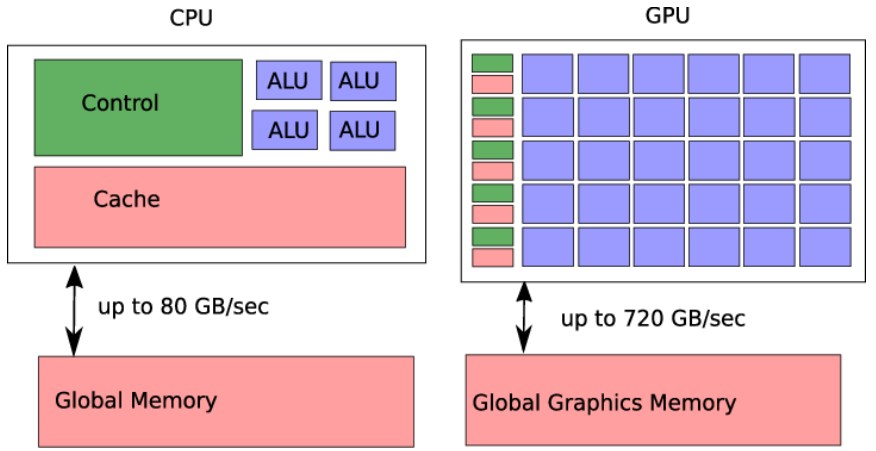
- Sabirnica (engl. bus)
- Memorija
- Jedinica za upravljanje napajanjem
- Jedinica za procesiranje videa
- Sučelje za prikaz slike
- VGABIOS
- Jedinica za procesiranje



Arhitektura jezgri ALU i CPU vs GPU

- ALU (Arithmetic Logic Unit) – komad sklopovlja optimiziran za izvođenje programa za jedan entitet (primjerci vrh ili fragment) (za GPU primjene)
  - Osnovne su funkcije za obradu brojeva s pomičnim zarezom i cijelim brojevima
  - Imaju funkciju za dodavanje i množenje u jednom koraku
  - Često imaju i druge funkcije poput usporedbe podataka, pohrane, čitanja, matematičkih operacija poput sinusa i kosinusa
- CPU – cilj izvršiti instrukciju što brže uz mogućnost mijenjanja operacija
- GPU – cilj optimizacija odnosno izvršenje velikog broja operacija istovremeno
- CPU ima puno manje jezgri od GPU-a
- MythBusters [video GPU vs CPU]

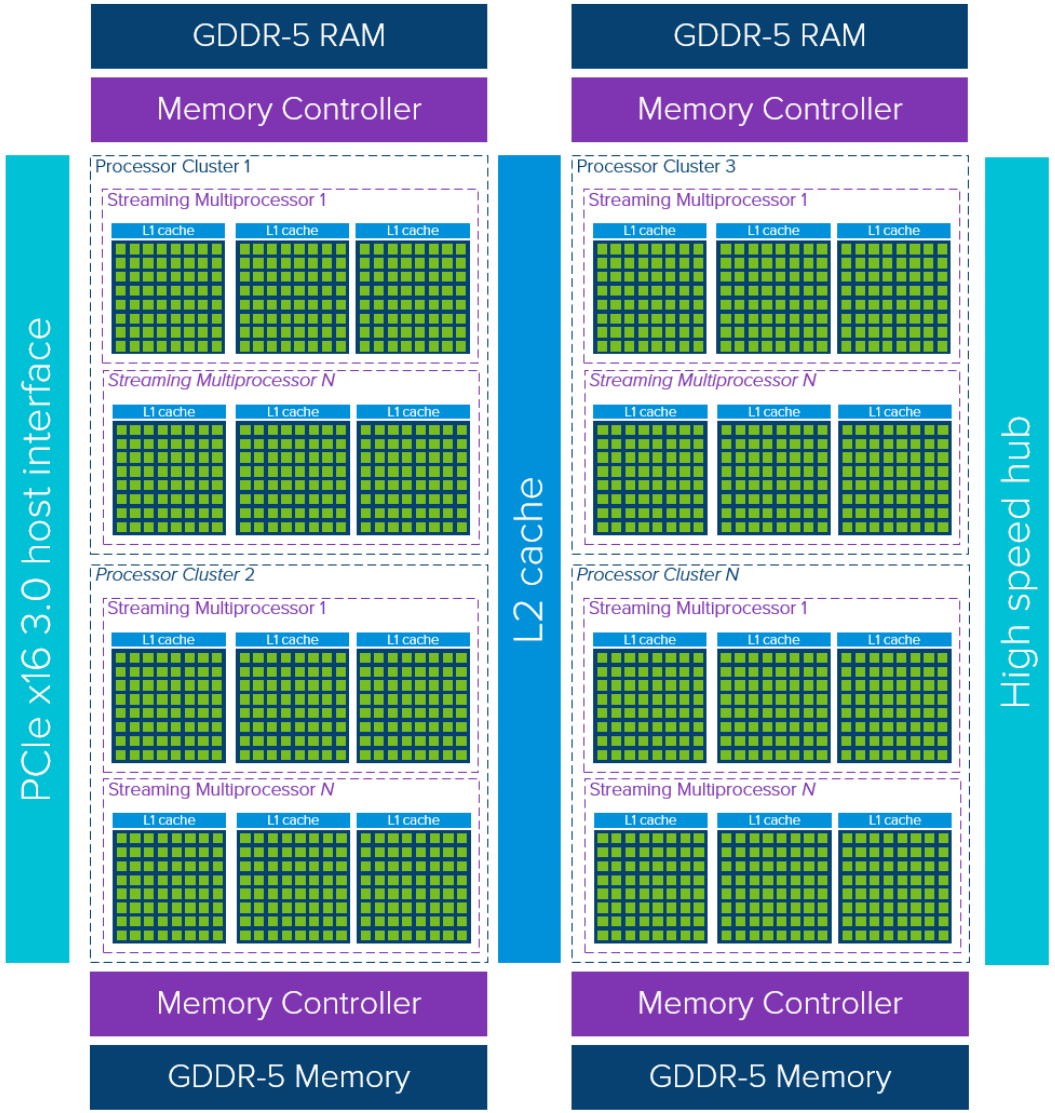




Organizacija današnjih GPU

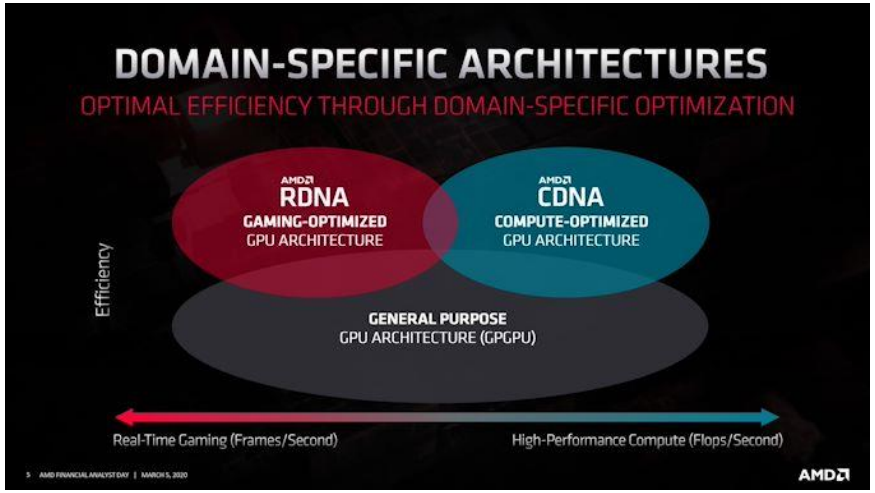
- Čip podijeljen na grupe jezgri koje rade neovisno jedni od drugih a sastoje se od više ALU-a
  - Compute Units (AMD)
  - Streaming Multiprocessors (Nvidia) - SM
- Više jezgri po pojedinom SM-u
- SM-ovi izvode istu operaciju na različitim elementima
- Svaki SM ima L1 predmemoriju te dijele L2 predmemoriju
- L2 predmemorija se veže sa CPU-om, memorijom sustava i drugim uređajima putem sabirnica (primjerice PCIe)
- Visoka propusnost
  - Ugrađena L2 predmemorija kojoj se može paralelno pristupiti
  - Najčešće spojeni na vrlo brze memorije koje se nalaze na samoj grafičkoj kartici DDR5, DDR6

Organizacija današnjih GPU - primjer



Mikroarhitektura GPU čipa

- Fizički raspored samog čipa, odnosno njegovih elemenata kao i implementacija programske i sklopoveske podrške za izvršavanje određenih instrukcija
- Arhitekture Nvidiae
  - Pascal 2016
  - Volta 2017
  - Turing 2019
  - Ampere 2020
- Arhitkture AMD-a
  - Tetrascale 2007
  - Graphics Core Next (GCN) 2012
  - Radeon DNA 2019
  - CDNA 2019 GPGPU
  - Radeon DNA 2

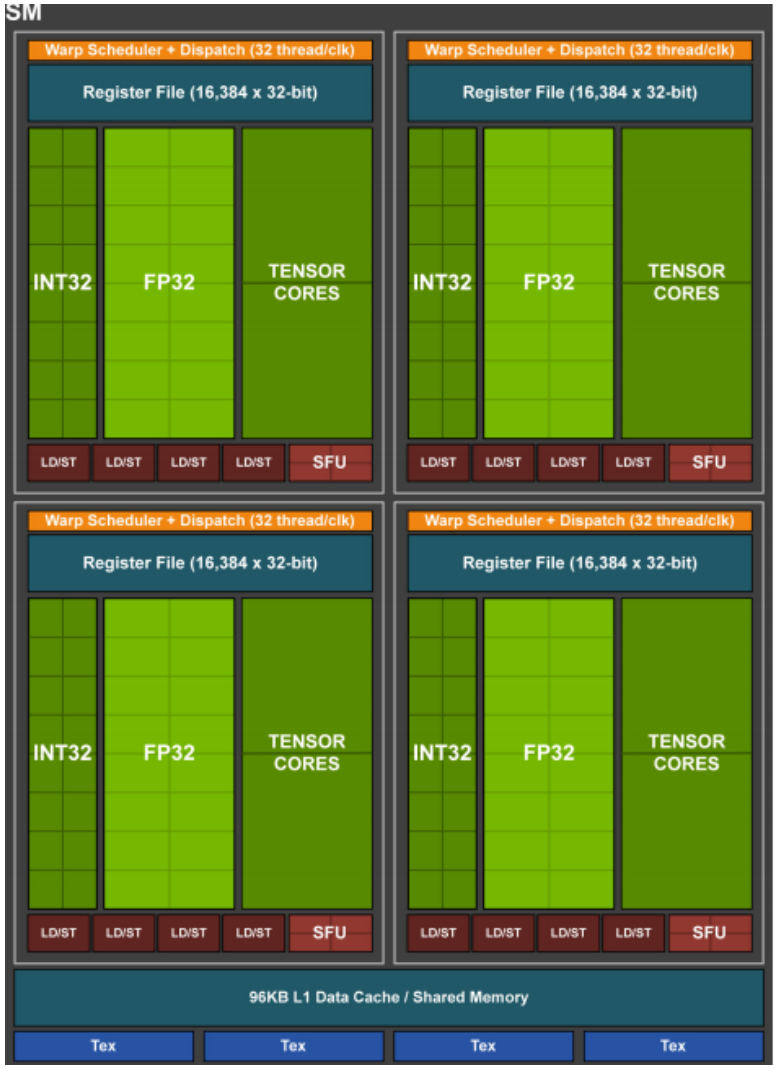


Pascal arhitektura

- Svaki SM ima 4 bloka od kojih svaki ima 32 ALU-a (svaki blok može procesirati 4 grupe dretvi s 32 dretve po grupi – na slici prikazana samo 2)
- CUDA jezgra – izvodi instrukciju
- SFU – izvode operacije nad brojevima s pomičnim zarezom (32 bita) poput korijenovanja, sinusa, kosinusa, logaritmiranja i potenciranja
- LD/ST – Load/Store jedinice koje pristupaju memoriji
- DP unit – Double Precission – 64 bitne operacije
- 2 L1 registra (24kB svaki)
- 8 teksturnih memorija
- Vjerojatno se koristi ranglista prioriteta za prebacivanje grupa dretvi

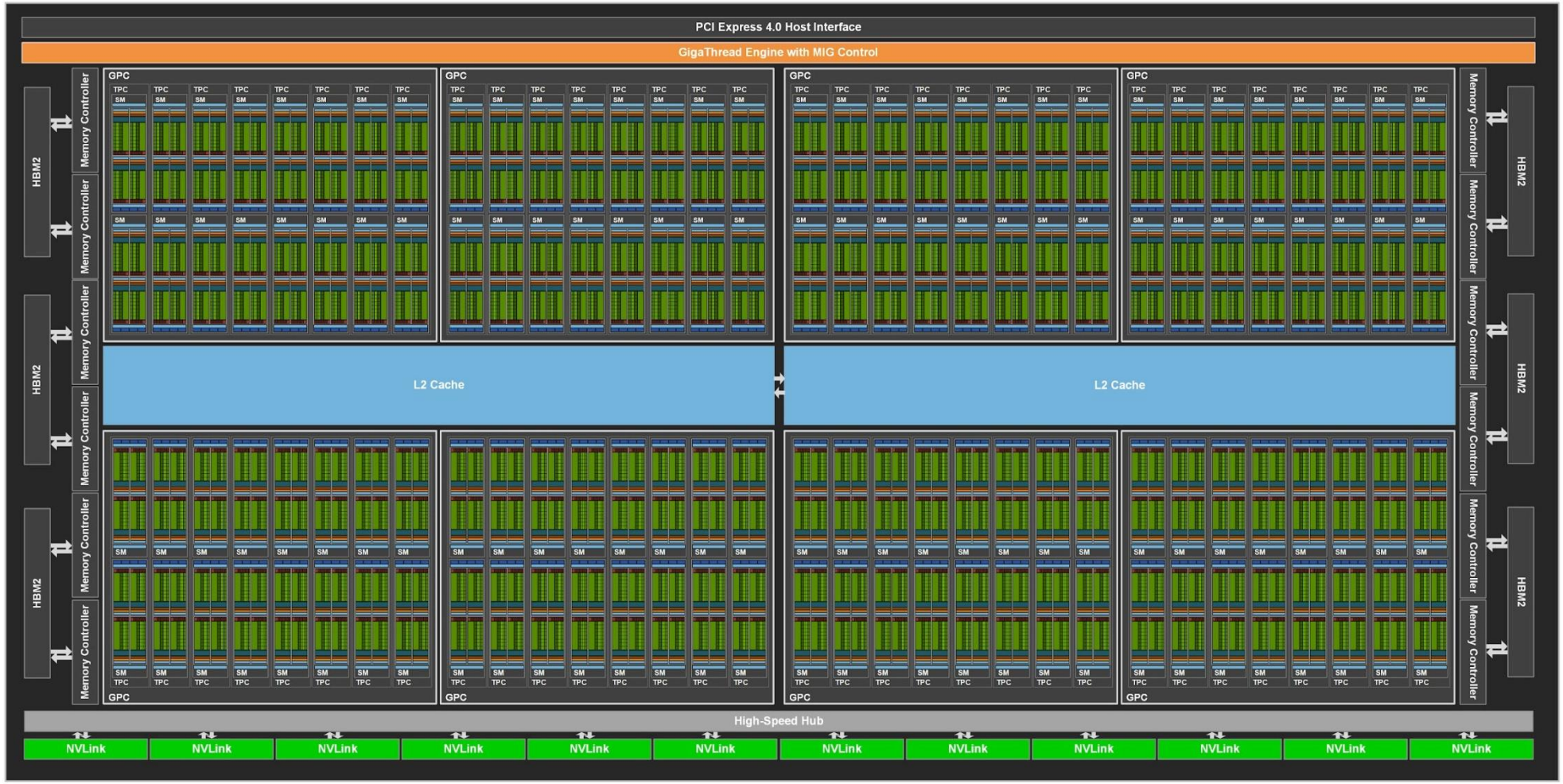
Turing arhitektura

- Omogućava sklopovski izračun praćenja zrake
- INT32 – jezgre za operacije nad cjelim brojevima
- FP32 – jezgre za operacije nad brojevima s pomičnim zarezom
- Tensor jezgre – specijalne jezgre koje su namjenjene prvenstveno operacijama s matricama te primjeni u AI području odnosno za GPGPU kartice



Ampere arhitektura

Primjer Ampeere GA100 kartice



Organizacija memorije

- Registri – najbrža memorija
- Lokalna memorija – memorija u koju se prelivaju podaci, može ići iz registara u L1, u L2 te u globalnu memoriju sustava (DRAM)
- Globalna memorija – današnji GPU imaju u prosjeku 2GB te do 150x sporija od operacija na registrima
- Dijeljena memorija
  - Unutar SM-a
  - Malo kašnjenje (~5ns)
  - Dijeli sklopovlje sa L1 memorijom
- L1/L2 predmemorija – drži podatke o pristupu lokalnoj i globalnoj memoriji
- Konstante i teksture