Haladó Grafika EA

Inkrementális képszintézis GPU-n

Pipeline

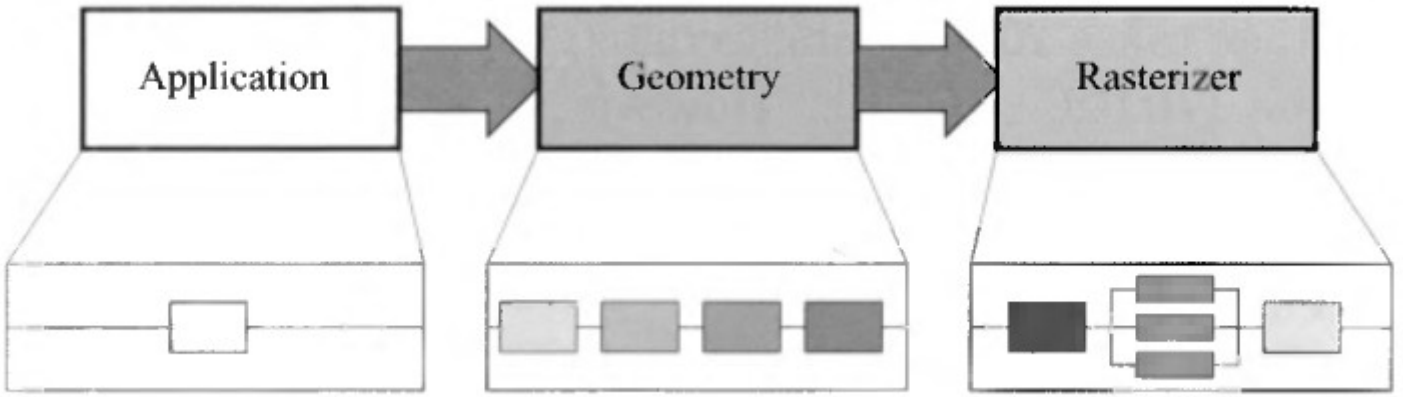
- Az elvégzendő feladatot részfeladatokra bontjuk
- Mindegyik részfeladatot más-más egység dolgozza fel (ideális esetben)
- Minden egység inputja, a pipeline-ban őt megelőző egység outputja

Pipeline

- Ha elkészült egy egység a részfeladatával, akkor továbbadja az eredményt → nem kell megvárnia, hogy befejeződjön a munka az egész munkadarabon
- → n fázis (stage) bevezetése optimális esetben n-szeres gyorsulást jelent (felfutás után)
- Nem csak GPU! Már a Pentium IV-nek is 20 pipeline stage-e volt (egy GeForce 3-nak 6-800)

Pipeline

- Vigyázzunk: ha egy részfeladat sokáig tart, akkor az előtte és utána lévők hiába készülnek el, nem tudnak tovább dolgozni
- → a pipeline sebességét a leglassabb komponense határozza meg



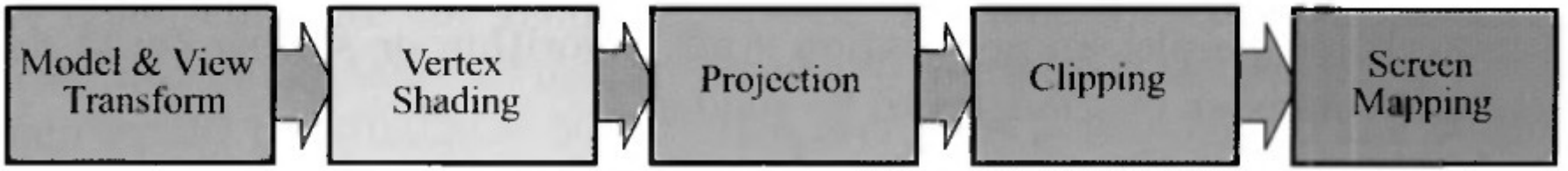
Pipeline – application stage

- A CPU-n futó programunkat jelképezi, általános hw-n
- Feladata
  - a kirajzolandó geometria előállítása (primitívek), és továbbadása a köv. fázisba.
  - Ütközésvizsgálat, gyorsítási adatszerkezetek és műveletek megvalósítása (pl. frustum culling)
  - a felhasználói input kezelése
  - transzformációk módosítása/animálása

Pipeline – geometry stage

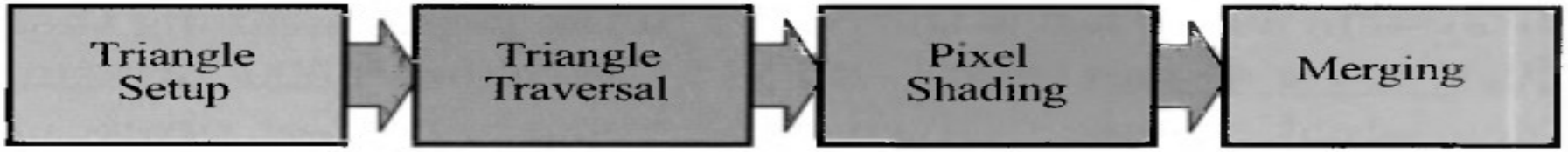
- A per-vertex és per-poligon műveletek itt zajlanak (és ma már per-patch is)
- Egy ideje már GPU-n
- Funkcionálisan a következő alfázisokra bontható:
  - Világ és nézeti transzformáció
  - Vertex és geometry shading/tesszeláció
  - Vetítés
  - Vágás
  - Képernyő leképezés (NDC → képernyő (egész) pixelkoordinátái – ff: DX9-ig hol a pixel közép...)

Pipeline – geometry stage



Pipeline – rasterizer stage

- Az előző fázis eredményének felhasználásával a feldolgozott primitív által lefedett pixelek színének kiszámítása
- ...azaz a fragmentjeinek kiszínezése
- Per-pixel/fragment számítások helye



Deriváltak stb. kiszámítása
Bedrótozva a hw-be

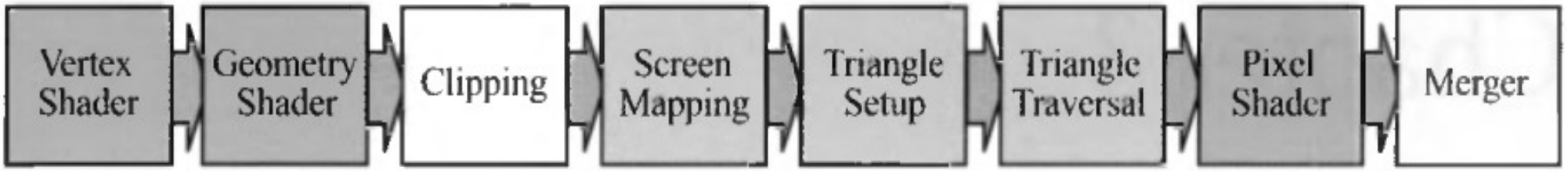
Mely pixelek középpontjait fedi le a háromszög és adatok interpolációja
- ez is bedrótozva a hw-be

Fragment színének kiszámítása, textúrázás, megvilágítás stb.
- programozható

Color buffer-be illesztése a fragment színének – konfigurálható, de nem programozható (még...)

GPU implementáció

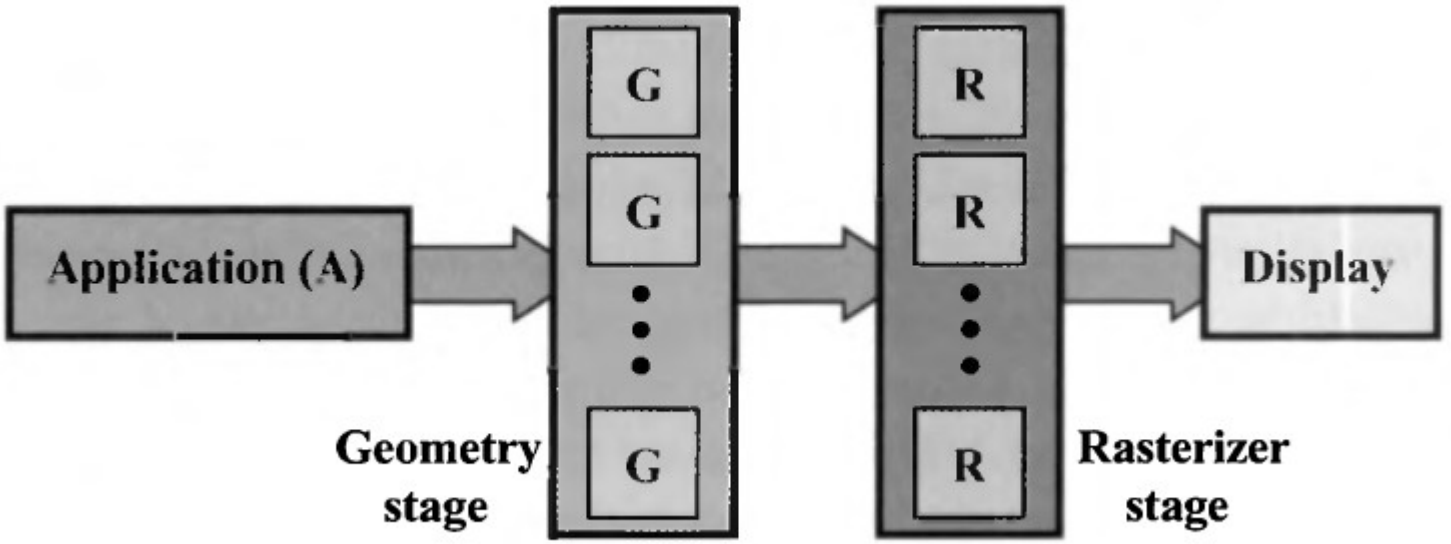
- Ahogy előző előadáson láttuk: bedrótozott → konfigurálható → programozható irányvonal mentén halad



GPU implementáció

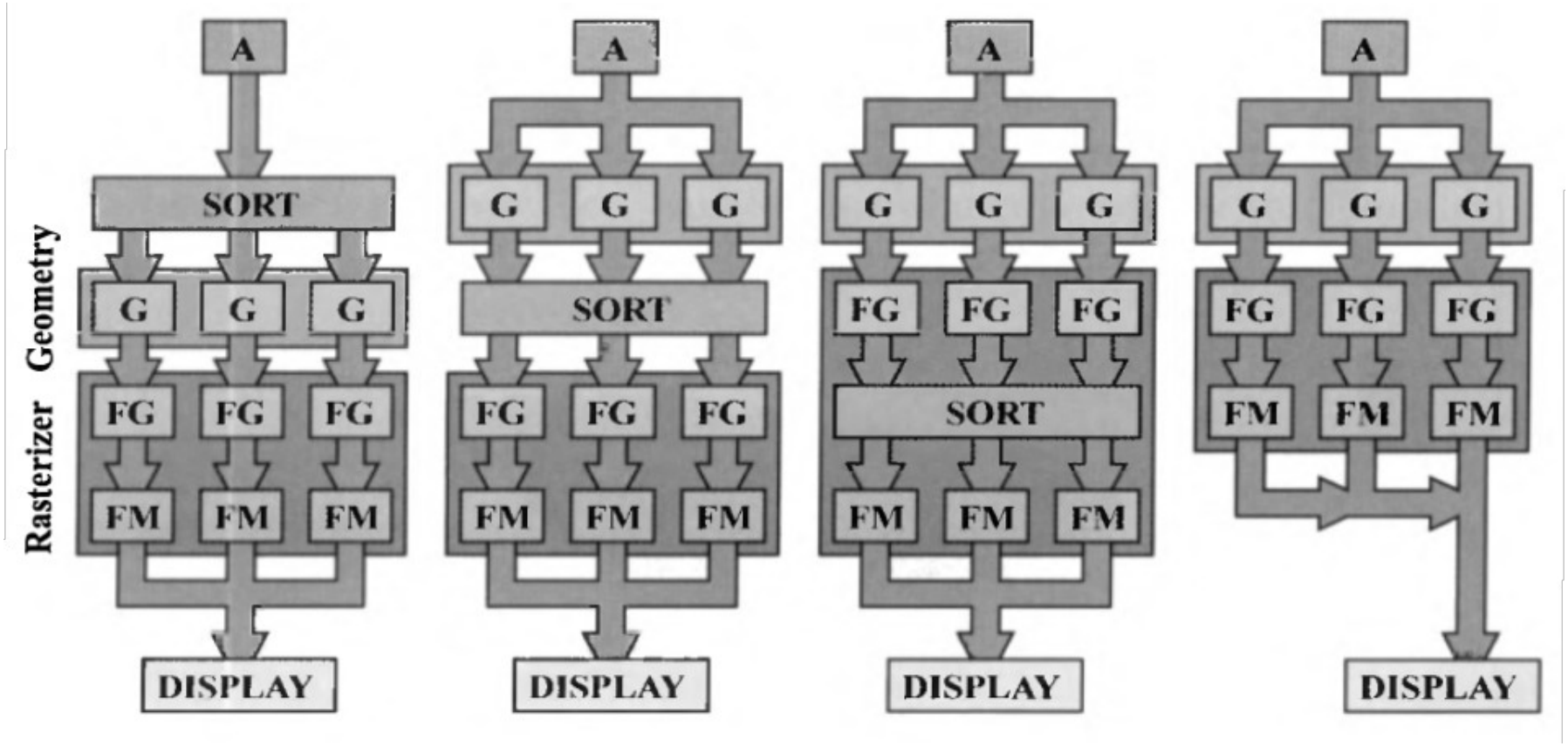
- Programozható fázisok (shaderek):
  - Vertex
  - Geometry
  - Tesselation
  - Fragment
- Unified shading model óta ugyanolyan gpu magokon futnak (desktopon)

GPU implementáció



GPU implementáció – párhuzamosítások

- FG = fragment generation
- FM = fragment merging



GPU implementáció - párhuzamosítások

- Sort first:
  - Cél: osszuk fel a képernyőt területekre, és az ezekbe eső primitívek egy olyan pipeline-ra kerülnek, „amelyiké” az adott terület
  - Primitívek rendezése, geometriai fázis egy részét el kell hozzá végezni
  - Ritka

GPU implementáció - párhuzamosítások

- Sort middle:
  - Ötlet: a raszterizálást csináljuk csak régiókra, *tile*-okra bontva
  - A primitív ahhoz/azokhoz a raszterizáló(k)hoz kerül(nek), aki(k)nek a tile-jába belemetsz
  - Globális post-process effektek, amelyek tetszőleges pixel-szomszédságot igényelnek (motion blur stb.) problémásak ilyen architektúrán a tile-ok közti kommunikáció igénye miatt
  - Pl.: Mali GPU-k

GPU implementáció - párhuzamosítások

- Sort last fragment:
  - FG után, de FM előtt jöjjön a rendezés → egy fragment egy FM-hez kerül, ez az optimum
  - Pl.: Playstation 3

GPU implementáció - párhuzamosítások

- Sort last image:
  - Tekinthető független pipeline-ok rendszerének is
  - FG és FM után jön csak a rendezés
  - A különböző objektumokat különböz processzorok dolgozzák fel

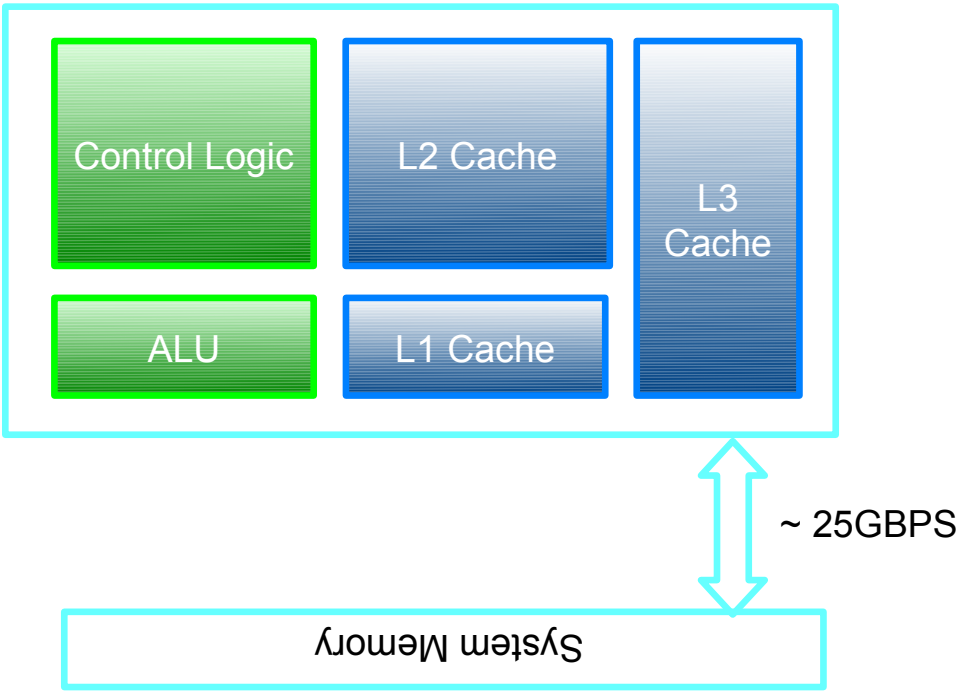
GPU implementáció - trükkök

- Z-culling:
  - Ha nincs „z” adat módosítás
  - Eldönteni, hogy a háromszög egy része az adott területen takarva lesz-e mélység alapján (közelítés háromszög csúcspontjainak mélysége stb. alapján)
- Early-Z:
  - Fragment mélységi értéke alapján tudjuk még a drága fragment shader lefuttatása előtt, hogy a fragment shaderbe beérkező fragment takarva van-e már
  - Pontos mélységi érték alapján eldobás, ha takarva van

Vashöz közelebb

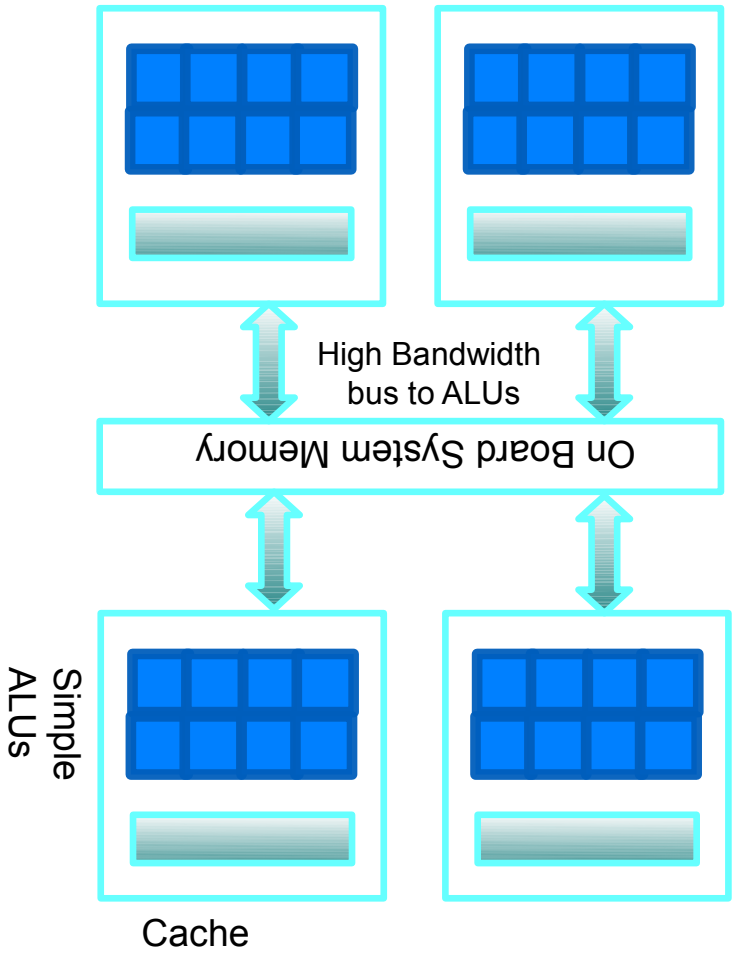
CPU felépítése

- Vezérlés és nem a nyers aritmetikai teljesítmény a hangsúlyosabb (ALU<32 ált.)
- A CPU-knál elsődleges egy-egy szál késleltetésének a minimalizálása



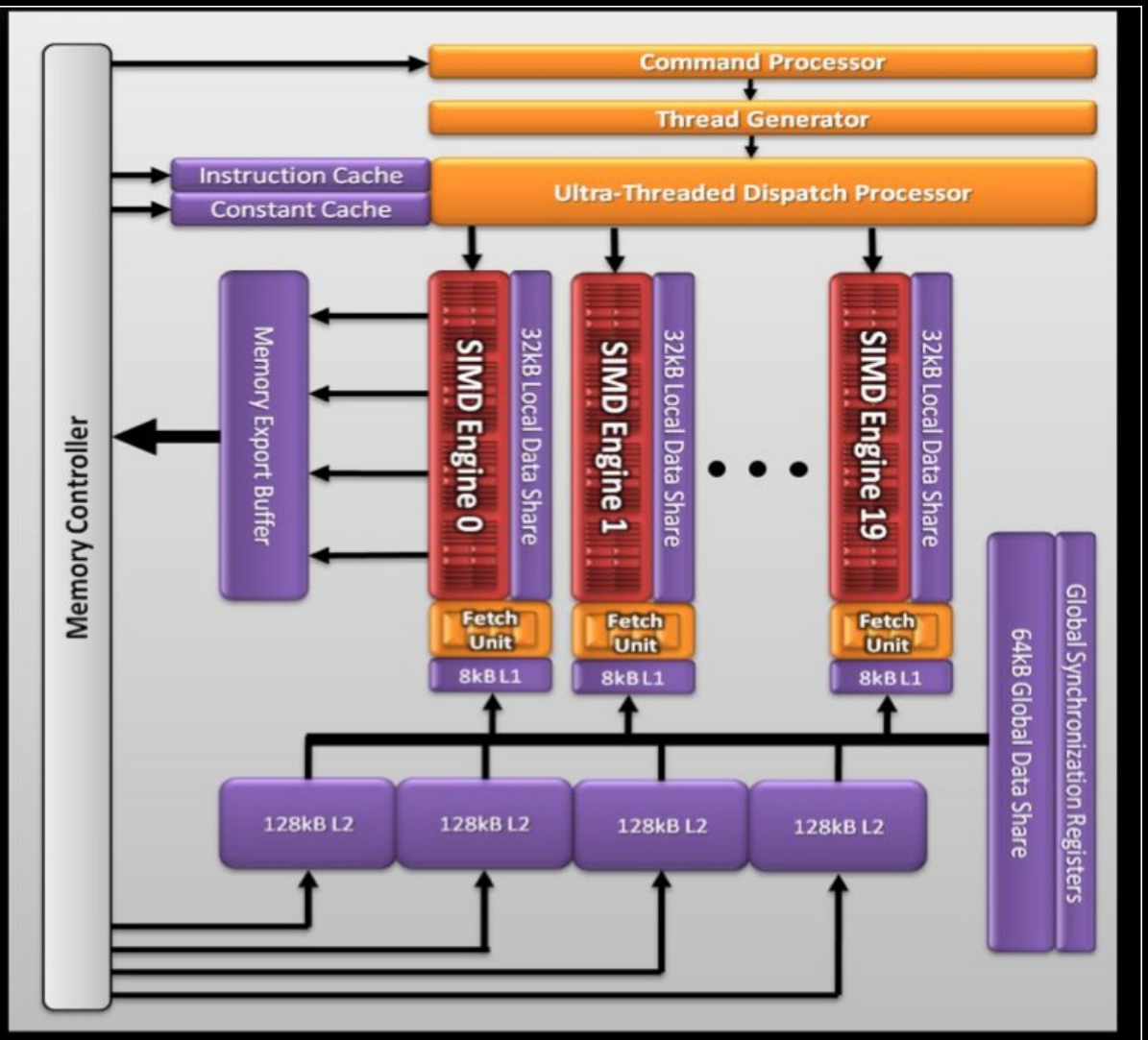
GPU felépítése

- Kevesebb vezérlés, több ALU „magonként”



ATI/AMD

AMD GPU Hardware Architecture



AMD 5870 – Cypress

20 SIMD engines

16 SIMD units per core

5 multiply-adds per functional unit (VLIW processing)

2.72 Teraflops Single Precision

544 Gigaflops Double Precision

Source: Introductory OpenCL *SAAHPC2010,* Benedict R. Gaster

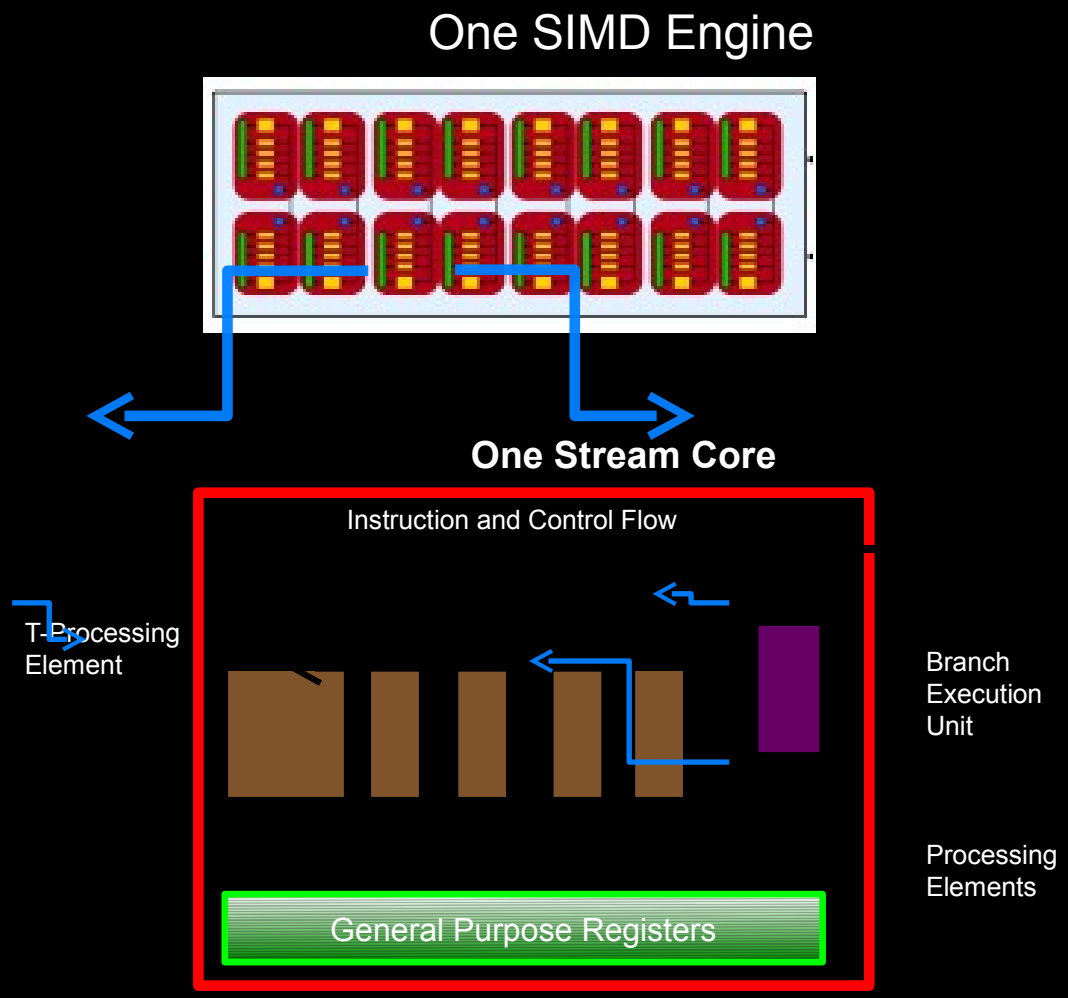
SIMD Engine

A SIMD engine consists of a set of “Stream Cores”

Stream cores arranged as a five way Very Long Instruction Word (VLIW) processor

- Up to five scalar operations can be issued in a VLIW instruction
- Scalar operations executed on each processing element

Stream cores within compute unit execute same VLIW instruction



Source: AMD Accelerated Parallel Processing OpenCL Programming Guide

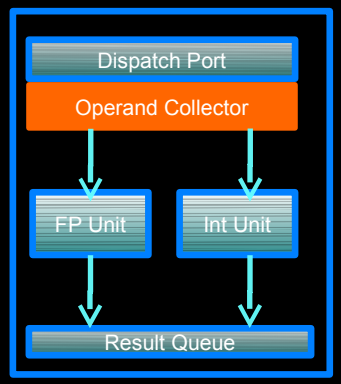
NVIDIA

Nvidia GPUs - Fermi Architecture

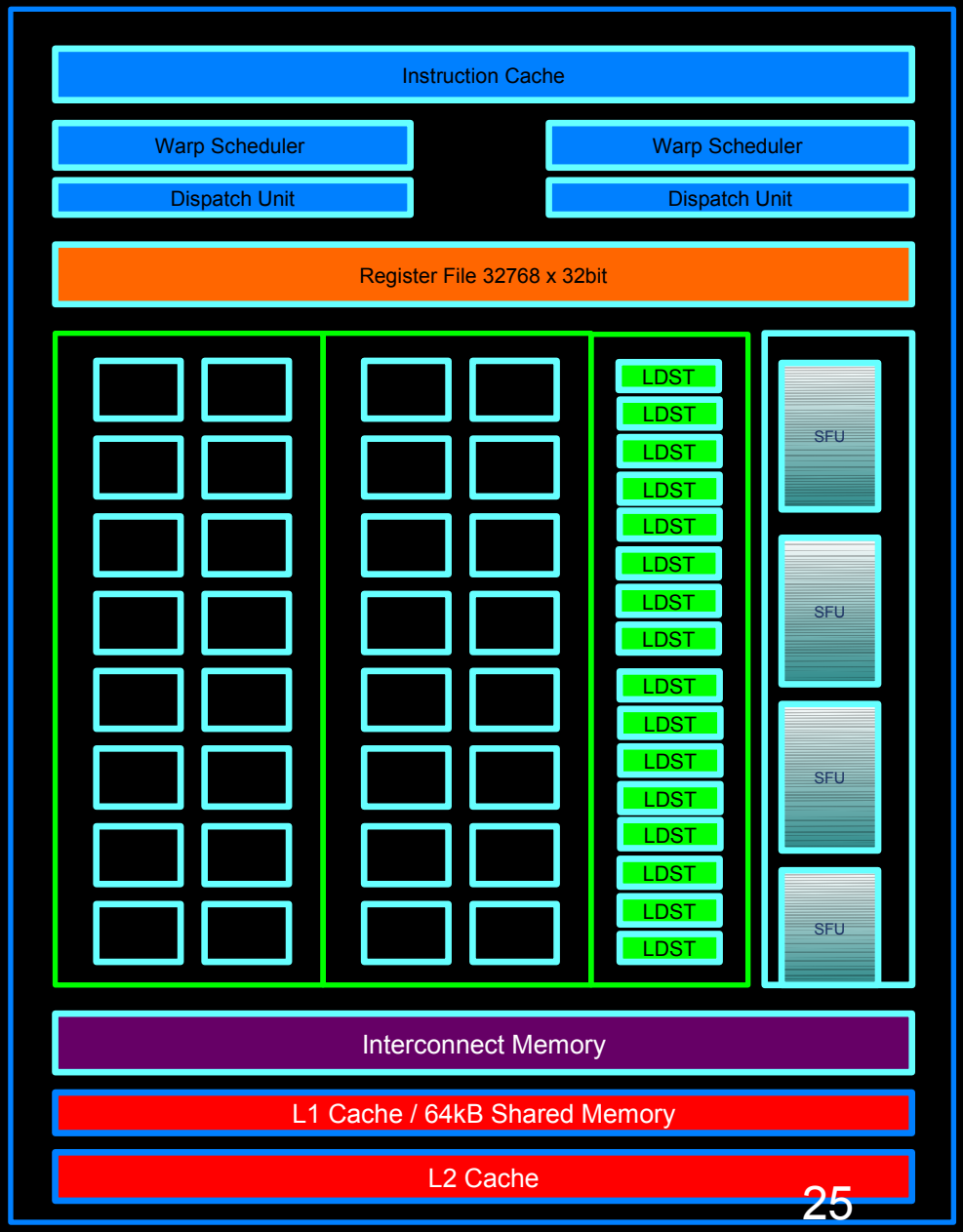
GTX 480 - Compute 2.0 capability

- 15 cores or Streaming Multiprocessors (SMs)
- Each SM features 32 CUDA processors
- 480 CUDA processors

Global memory with ECC



Source: NVIDIA's Next Generation CUDA Architecture Whitepaper



Perhaad Mistry & Dana Schaa, Northeastern Univ

Nvidia GPUs – Fermi Architecture

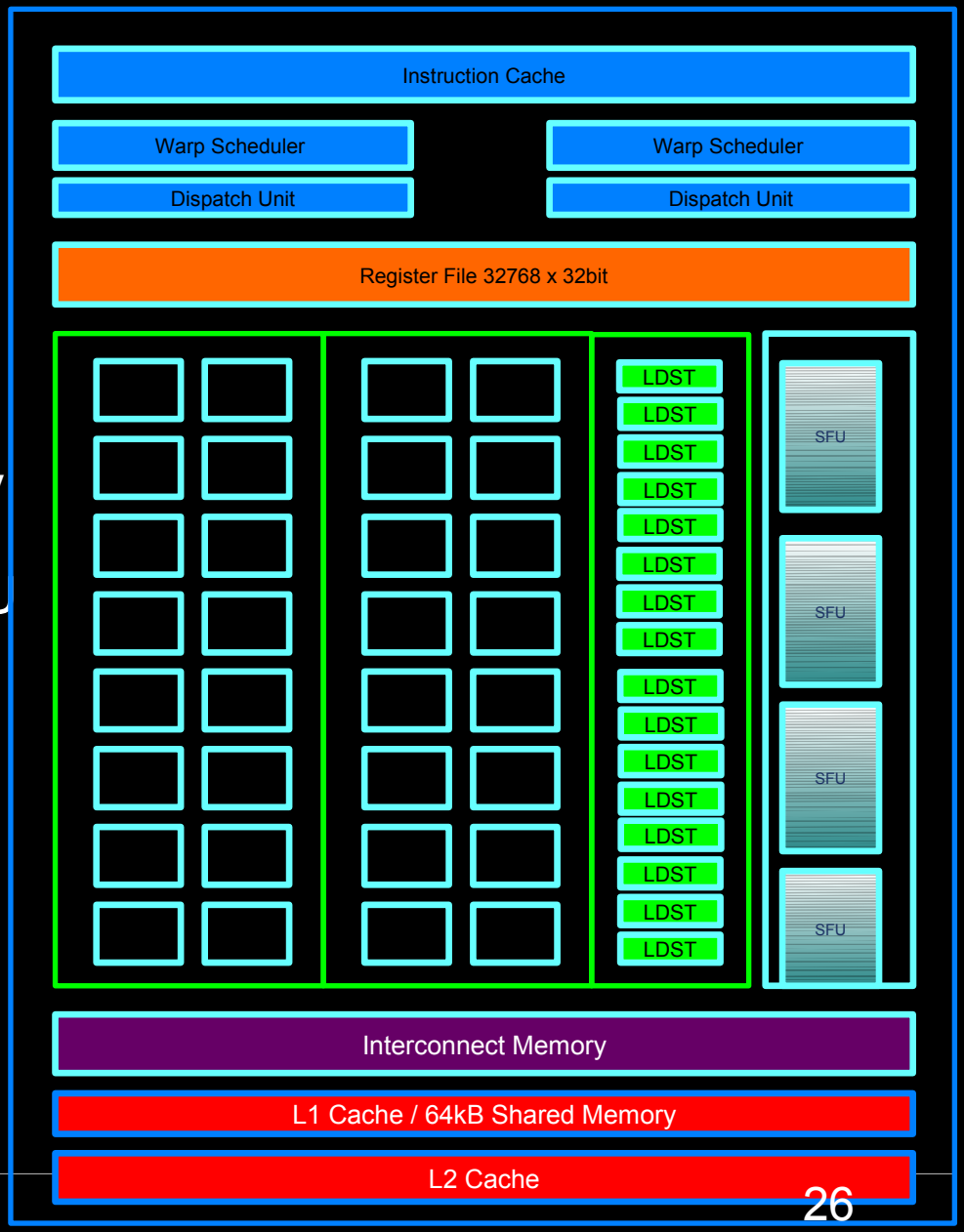
SM executes threads in groups of 32 called warps.

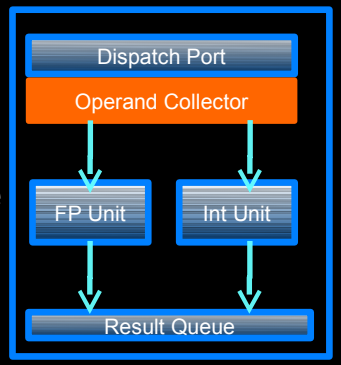
- Two warp issue units per SM

Concurrent kernel execution

- Execute multiple kernels simultaneously to improve efficiency

CUDA core consists of a single ALU and floating point unit FPU





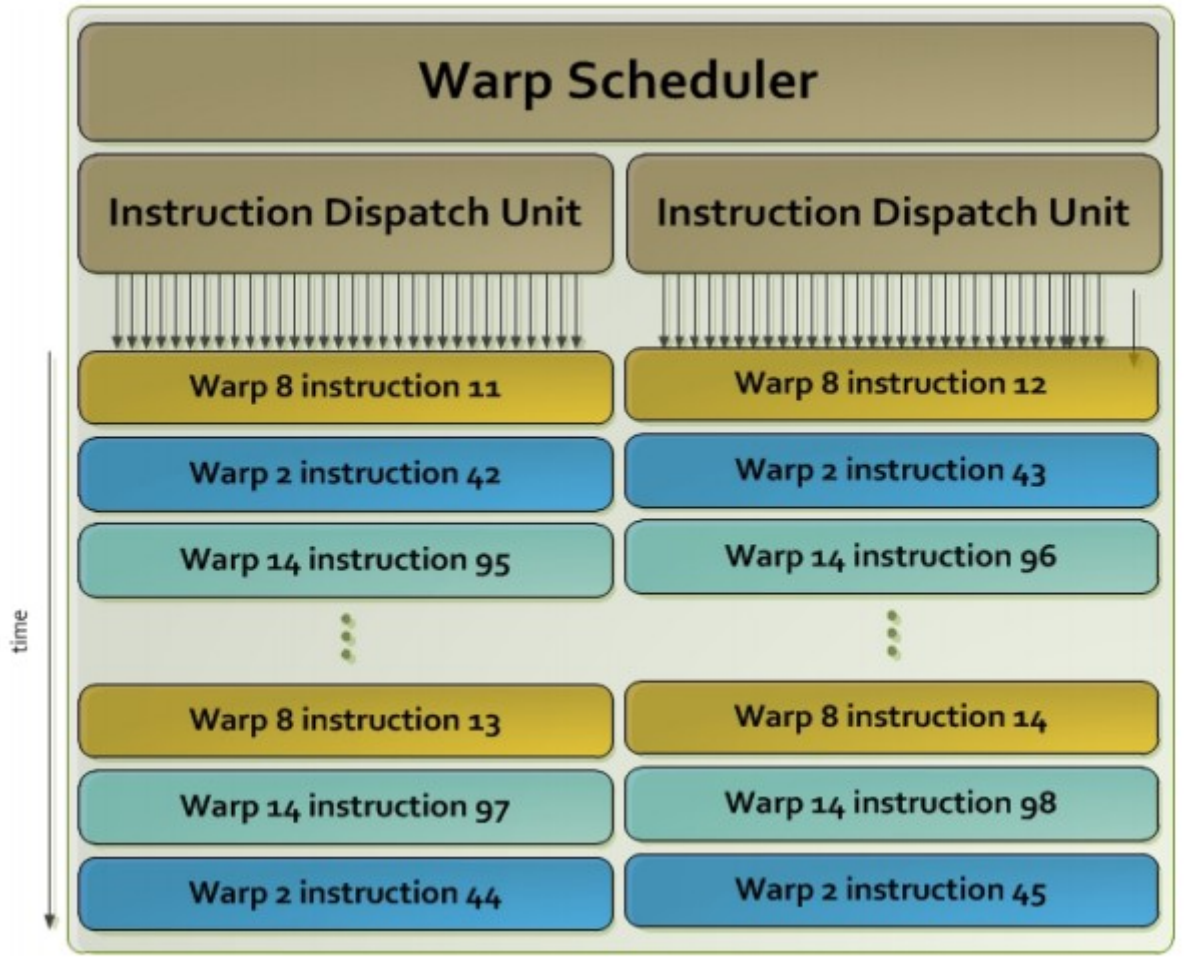
Source: NVIDIA's Next Generation CUDA Compute Architecture Whitepaper

NVIDIA Kepler

- Teljes Kepler 15 SMX egységből áll és hat 64

SMX egység

- SMX = Streaming Multiprocessor, megnövelt duplapontosságú teljesítménnyel
- Multiprocesszor: valahány streamprocessor csoportja (Tesla = 8, Fermi = 2x16)
- Kepler-ben: 192 egyszeres pontosságú CUDA core
- SFU: special function unit a különleges dolgokhoz (inverse square root, sin, cos, transzcend fv közelítések stb.)
- Warp: 32 szál


Warp Scheduler
Instruction Dispatch Unit
Instruction Dispatch Unit
Warp 8 instruction 11
Warp 8 instruction 12
Warp 2 instruction 42
Warp 2 instruction 43
Warp 14 instruction 95
Warp 14 instruction 96
Warp 8 instruction 13
Warp 8 instruction 14
Warp 14 instruction 97
Warp 14 instruction 98
Warp 2 instruction 44
Warp 2 instruction 45
time

SIMT

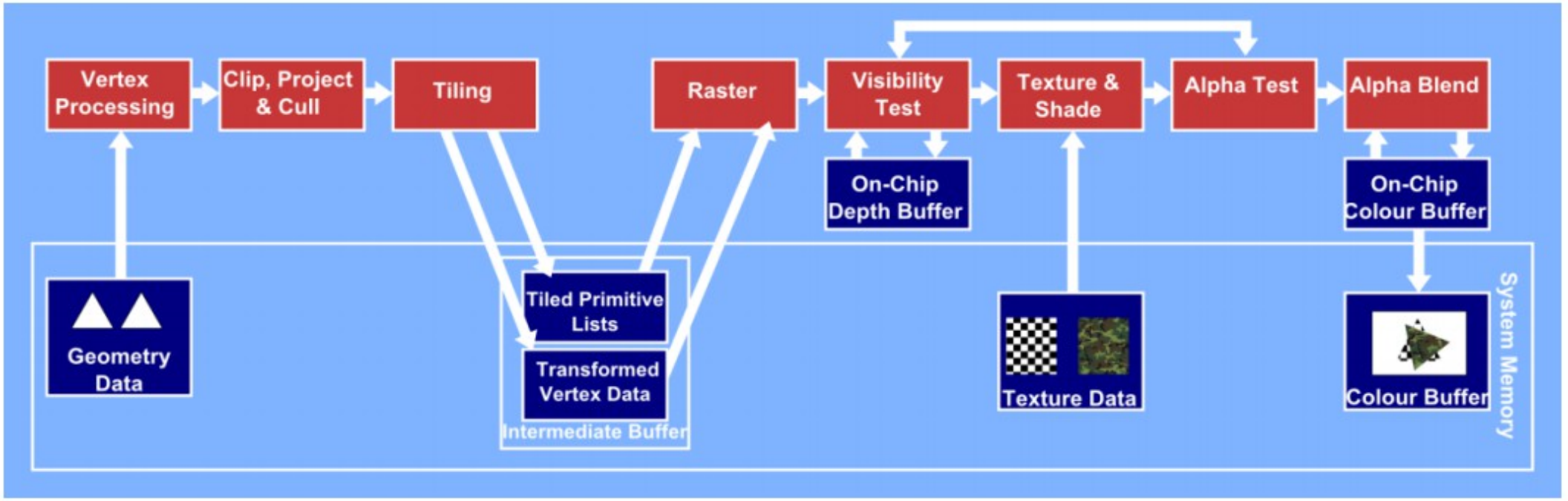
- Single Instruction, Multiple Thread
- Step lock-ban mennek az egy csoporthoz tartozó stream processorok, ugyanazt az utasítást hajtják végre, csak más adatokon

PowerVR

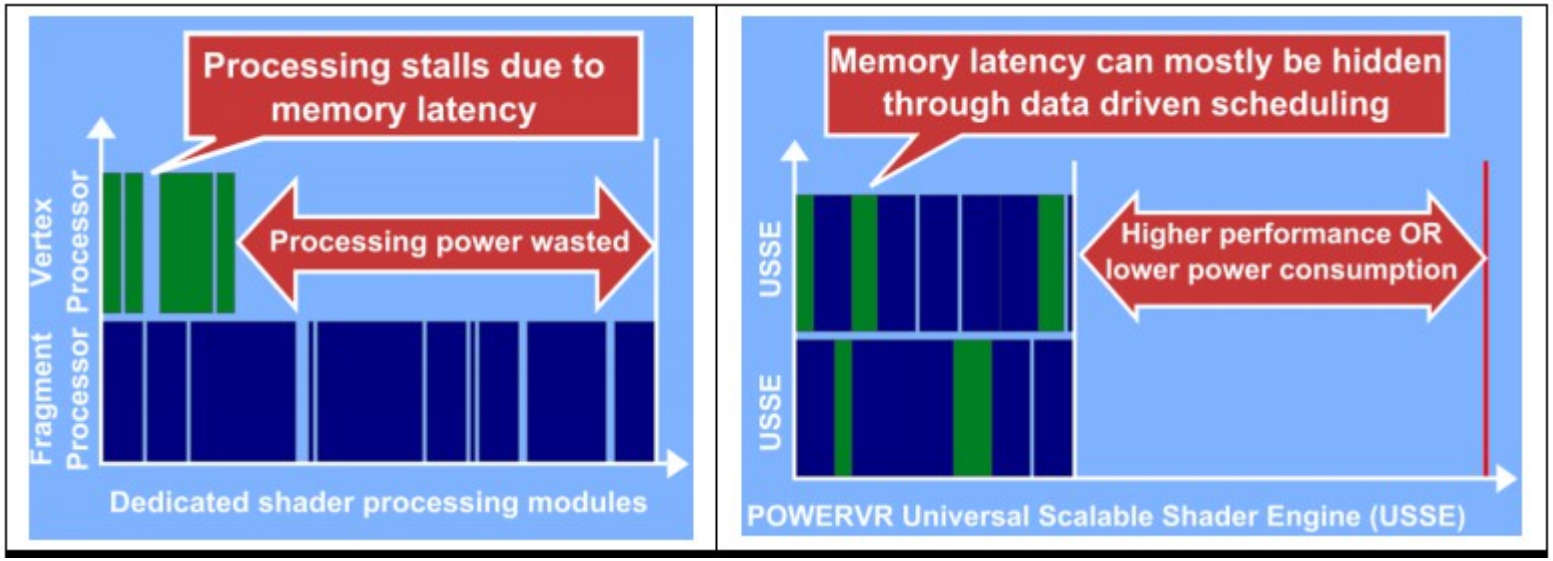
PowerVR

- Elsősorban mobileszközökhöz, de skálázható eszközök, GPGPU-t is támogatnak (CPU kisegítésére)
- Universal Scalable Shader Engine 1-2...
- 2-16 mag (és nő)
- 32x32-es tile-okra osztja a képernyőt
- Pl.: iPad, iPhone, iAkármi, Samsung Galaxy S, Tab, Wave I, II...
- Több infó: http://www.imgtec.com/powervr/insider/docs/POWE

Tile alapú rajzolás



USSE


Processing stalls due to memory latency
Vertex Processor
Fragment Processor
Processing power wasted
Dedicated shader processing modules
Memory latency can mostly be hidden through data driven scheduling
USSE
USSE
Higher performance OR lower power consumption
POWERVR Universal Scalable Shader Engine (USSE)

Tile Accelerator

- Vágás, vetítés, cull – hagyományos szerelőszalagnál a vertex shadert hajtja végre

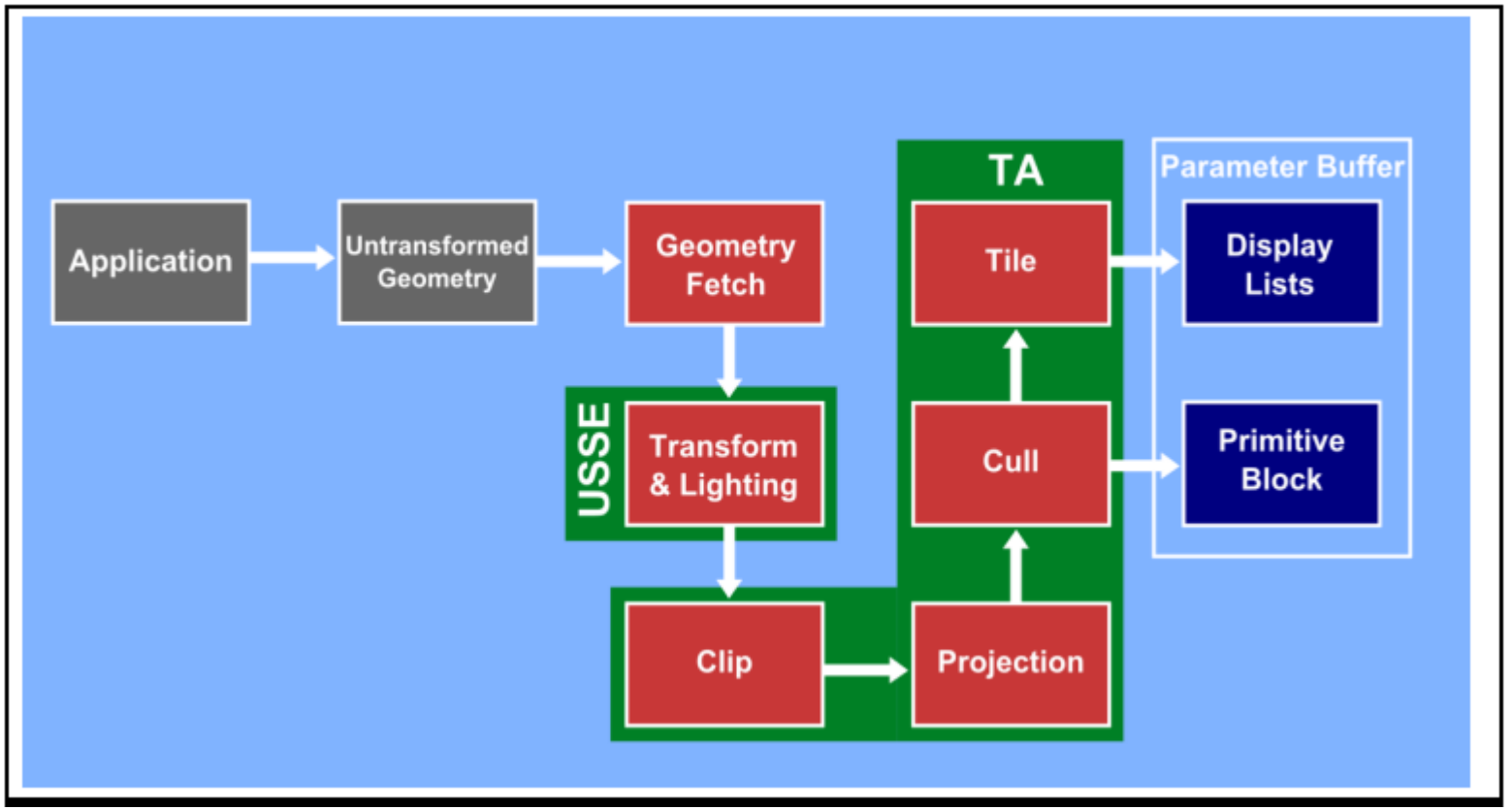
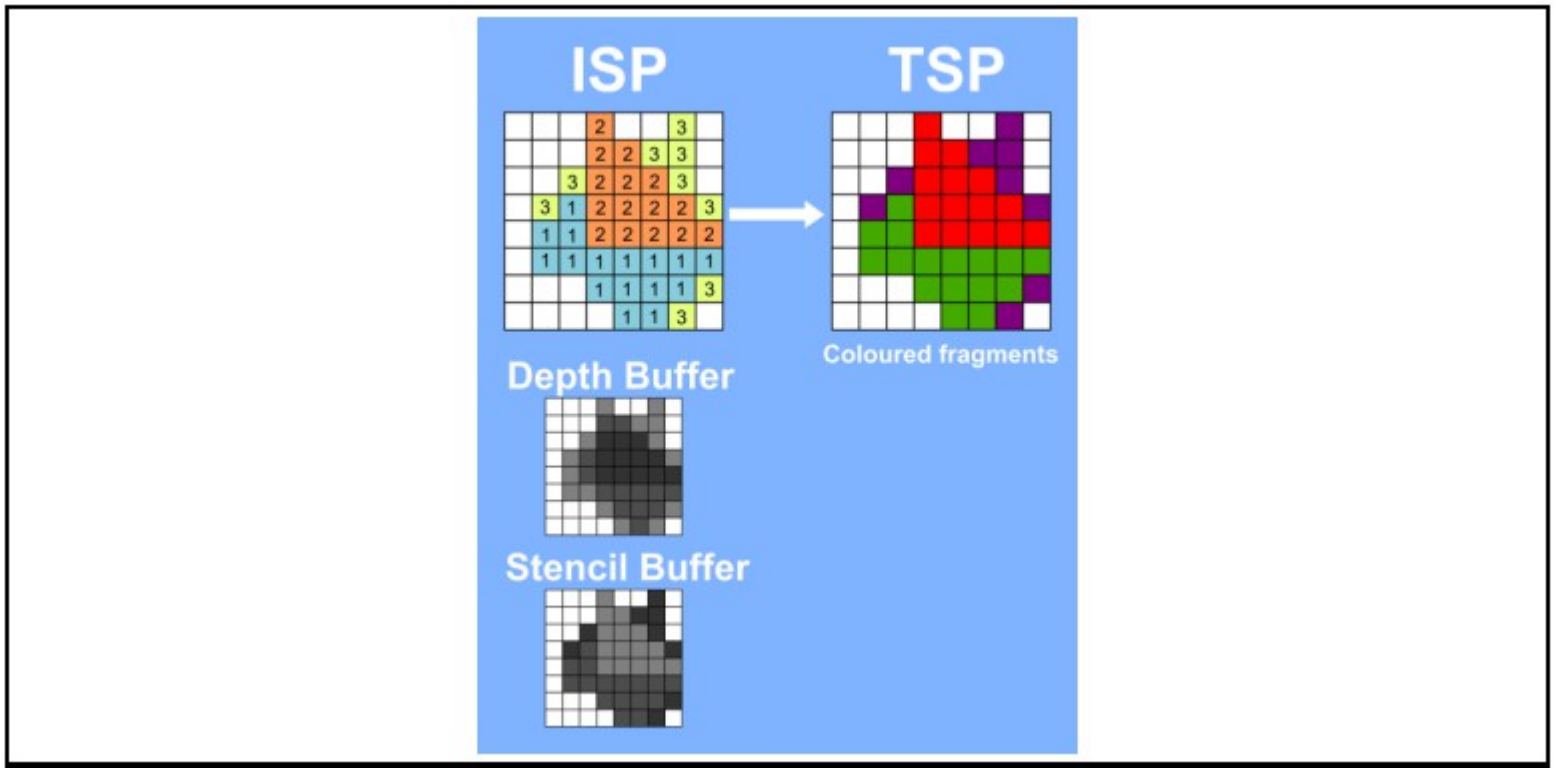
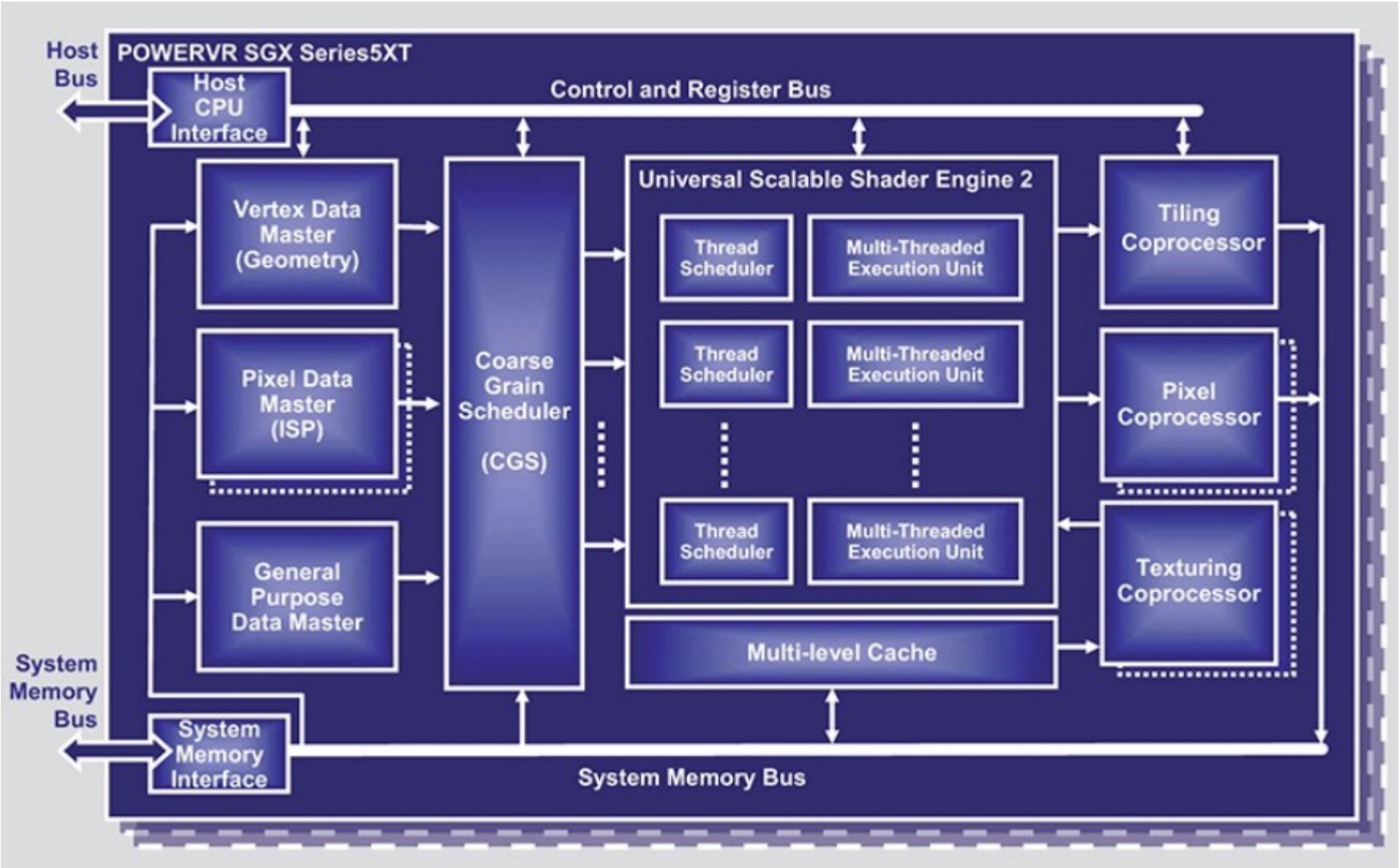


Image Synthesis Processor

- Tile-onkénti láthatóságvizsgálat (takarás)
- TSP = Texture and Shading Processor



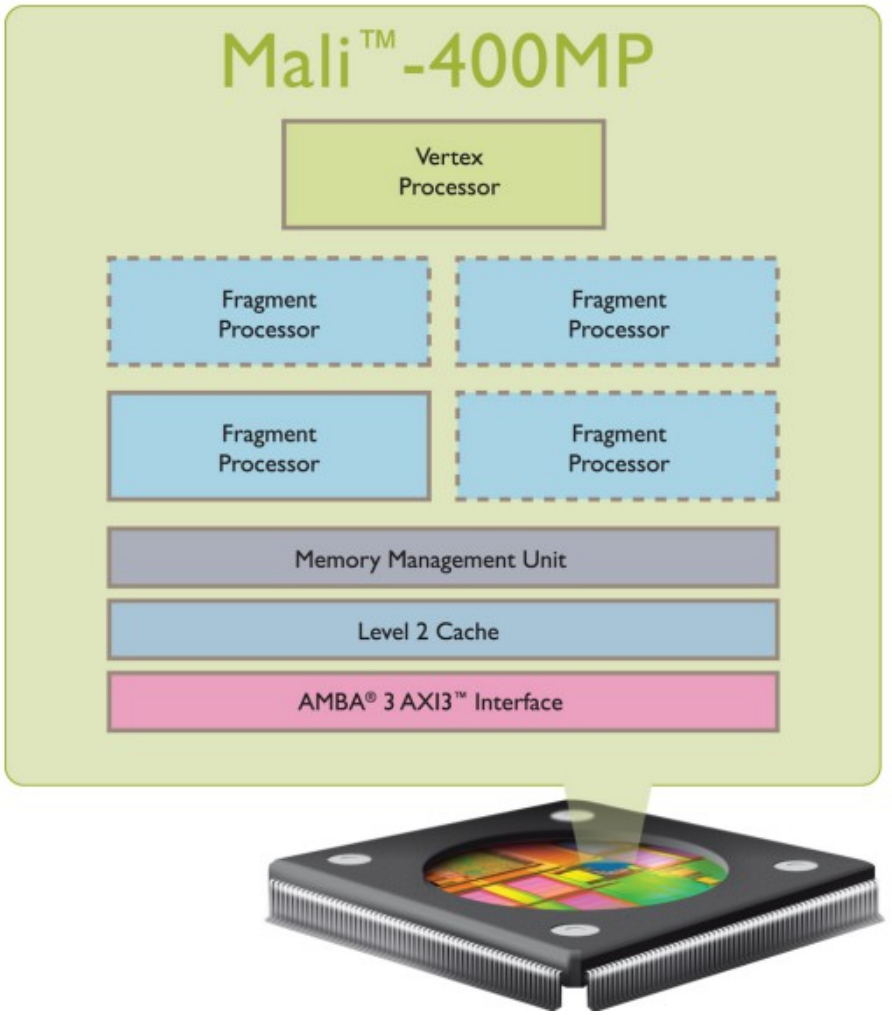
PowerVR SGX Series5XT



ARM Mali

ARM Mali

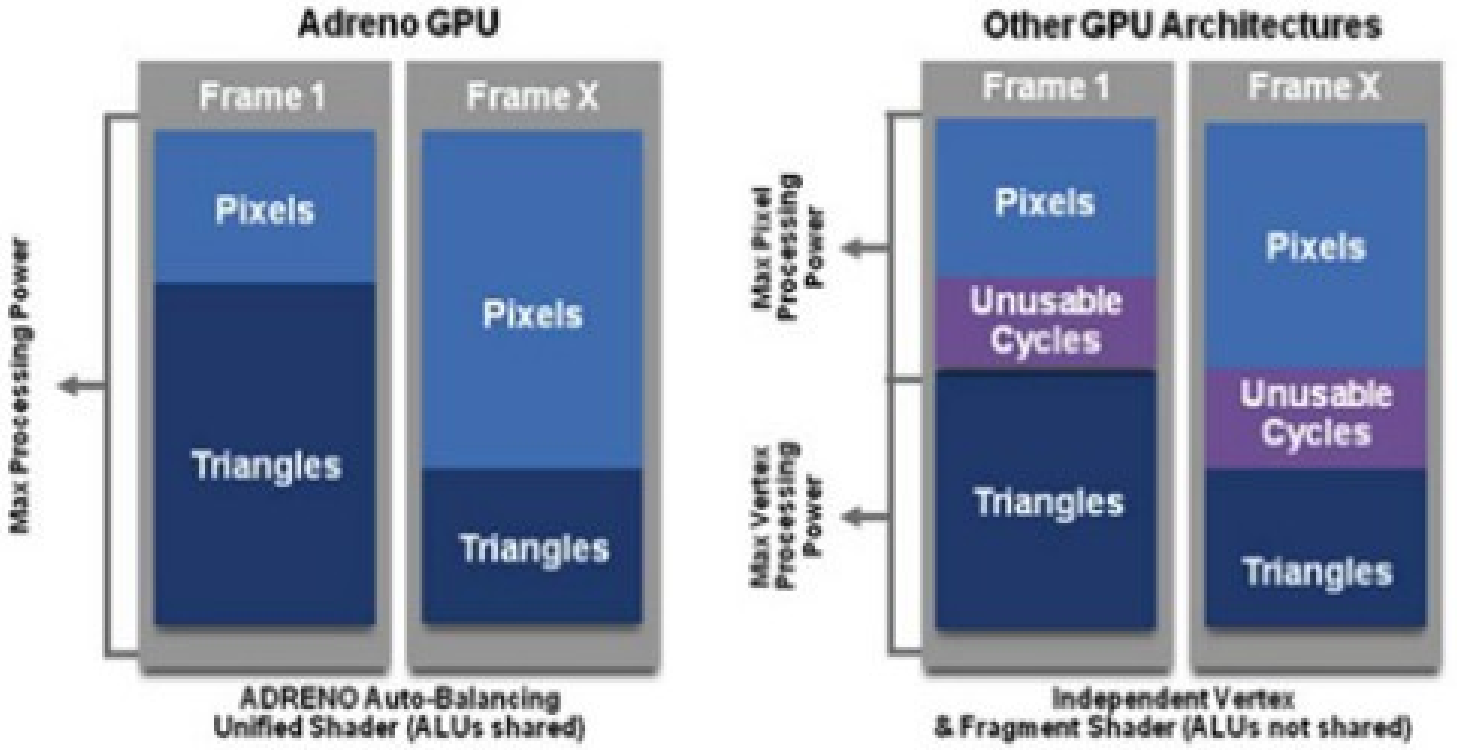
- Nem csak GPU-kat gyártanak és nem csak mobileszközökre
- Mali 400 MP 1-4 magos
- Más Mali eszközökkel integrálható
- Pl.: Samsung Galaxy S2



Qualcomm Adreno

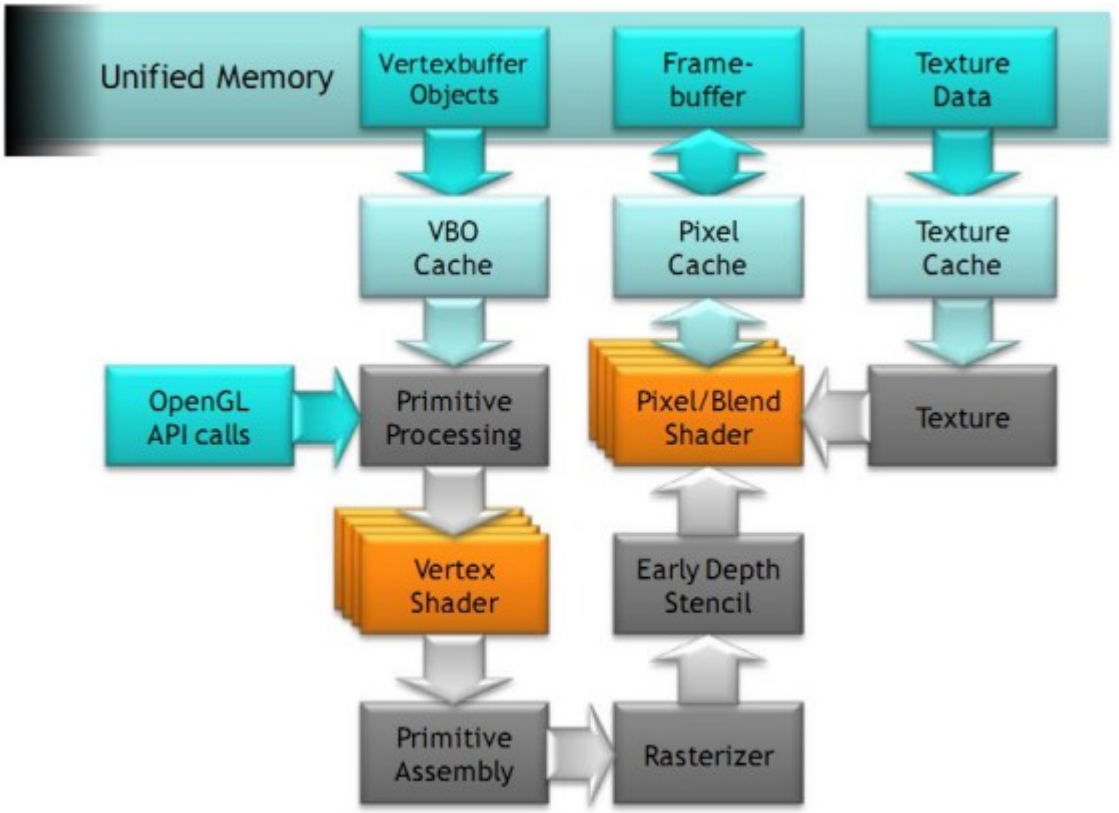
Qualcomm Adreno

- Pl.: HTC telefonok (Desire, Evo stb.)
- Ők is „forradalmiak”:



NVIDIA (megint)

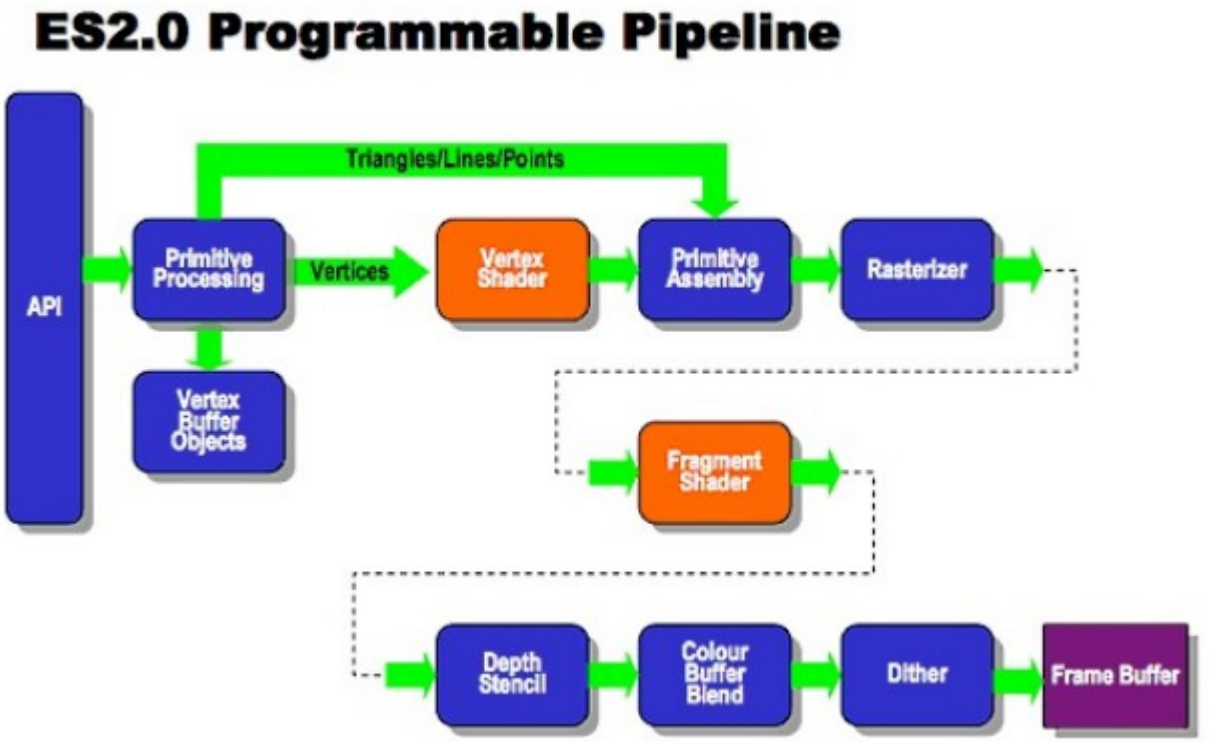
Ultra low power GeForce



API-k

API-k

- OpenGL ES 1.0-3.0



API-k

- OpenVG
  - Vektorgrafikákra
- Microsoft Direct3D Mobile