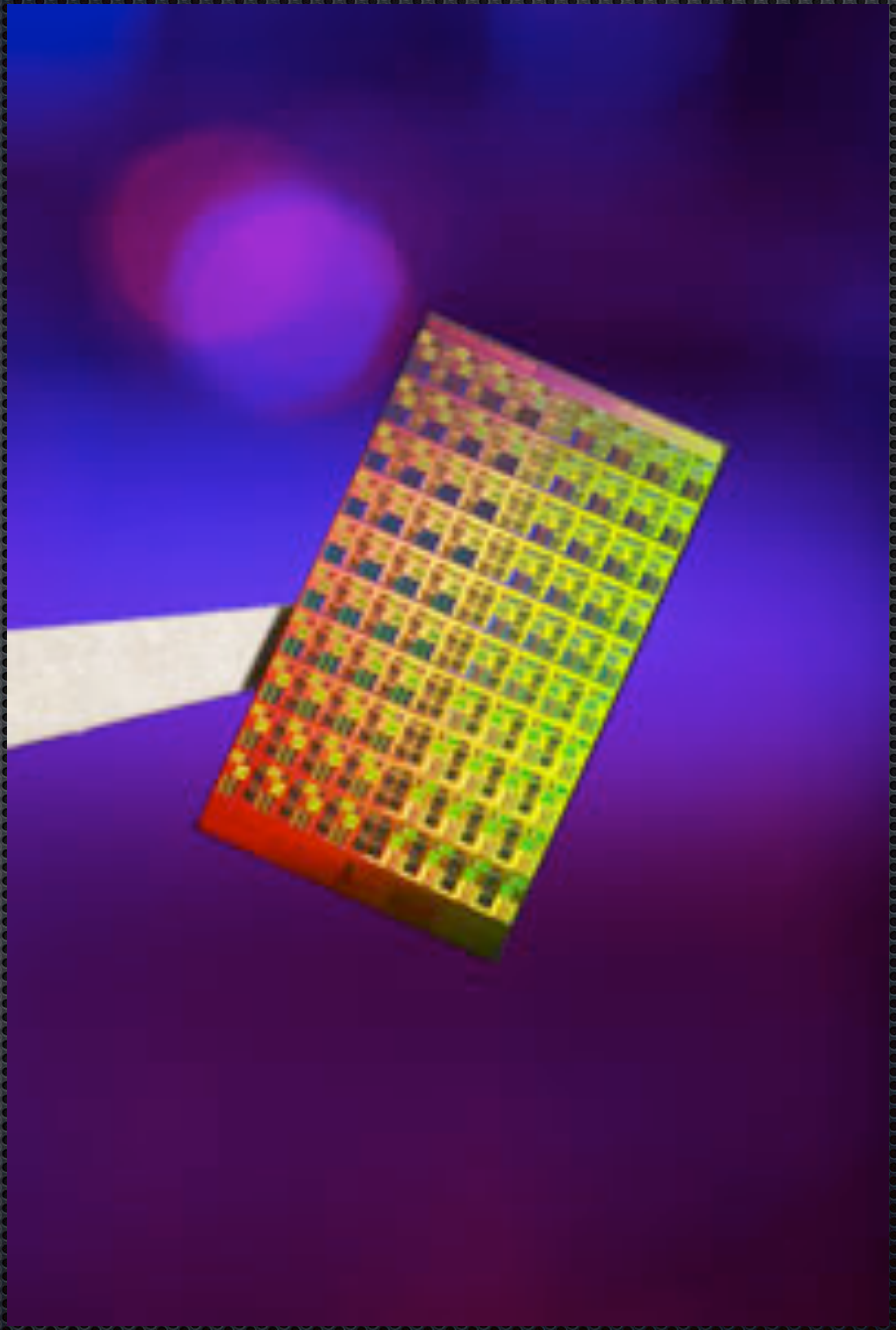
Vector Processing

Manycores, SIMDs, and Cells (Oh my!)

Sodani HC 2015

Knights Landing (KNL): 2nd Generation Intel Xeon Phi Processor

Intel 80-core Teraflops Research Chip



Intel Exec Quote

“We can now put 80 cores on a chip.
We just don’t have any idea of
what to do with them.”

Intel MIC (Xeon Phi)

- Many Integrated Cores - after Larrabee GPU & 80 core
- Knights Ferry prototype with 32 cores, 1024 bit ring
- Knights Corner product
- 50 x86 (P54C) cores, 4-thread, superscalar, in-order
- 512 bit SIMD registers
    - 16x32 bit vectors, gather/scatter/mask
- Programmed with OpenMP, OpenCL, Intel Cilk Plus
- Claim up to 1.2 TFLOPS @ 300W

Knight’s Landing Phi

- 72 Silvermont Atom cores in 36 tiles
  - 2-wide OoO issue, 4-way threading, mesh connect
- Two 512-bit vector units
- Each pair shares 1MB cache
  - MESIF, directory-based coherence with other tiles
- Up to 215 Watts

Memory

- 16GB MCDRAM on same carrier (a la Pentium Pro)
- 3-modes: All cache, all RAM, part cache part RAM
- Specially allocated in software for critical data
- Other data in slower DDR

Interconnect

- Mesh of rings with three modes
- All-to-all: slowest but most general
- Quadrant: Directories for groups of 9 tiles (faster)
- Sub-NUMA: Quadrants are separate NUMA domains that allow explicit software optimization for locality (fastest, most programming effort)
- 25GB/s Omni-Path external ports

Performance

- 5X peak DP floating performance of Knights Corner (implies about 6TFLOPS)
- At 200W, gives about 30GFLOPS/W
- Knights Mill variant introduced 2017 with optimizations for machine learning

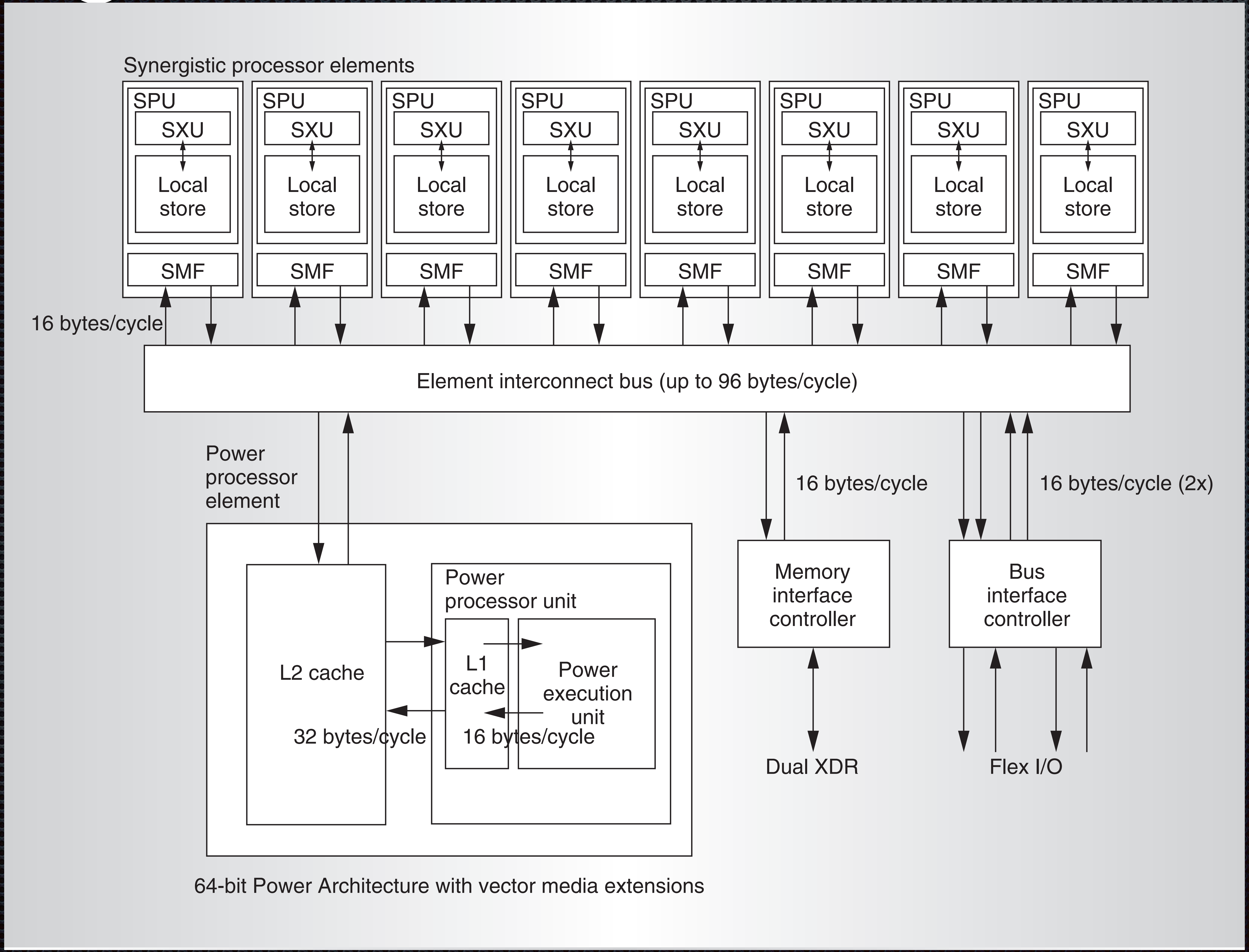
Synergistic Processing in Cell's Multicore Architecture

Gschwind, et. al.

Cell Broadband Engine Overview

- Heteogeneous Multicore Processor
- One Power-PC-based control processor (PPE)
- Eight “Synergistic” Vector-only Processors (SPE)
- Interconnect Bus (EIB)
- Common address space

CBE Diagram


Synergistic processor elements
SPU
SXU
Local store
SMF
16 bytes/cycle
Element interconnect bus (up to 96 bytes/cycle)
Power processor element
L2 cache
Power processor unit
L1 cache
Power execution unit
32 bytes/cycle
16 bytes/cycle
16 bytes/cycle
16 bytes/cycle (2x)
Memory interface controller
Bus interface controller
Dual XDR
Flex I/O
64-bit Power Architecture with vector media extensions

PPE Functionality

- Runs Operating System
- General Purpose (scalar heavy) computation
- Organizes structure of global address space
- Coordinates distribution and collection of data

SPE Functionality

- Vector/SIMD Instruction Set
- 128 128-bit Registers
    - Register Contents are polymorphic
- 256 KB Local Store
- 2-way Specialized Pipeline

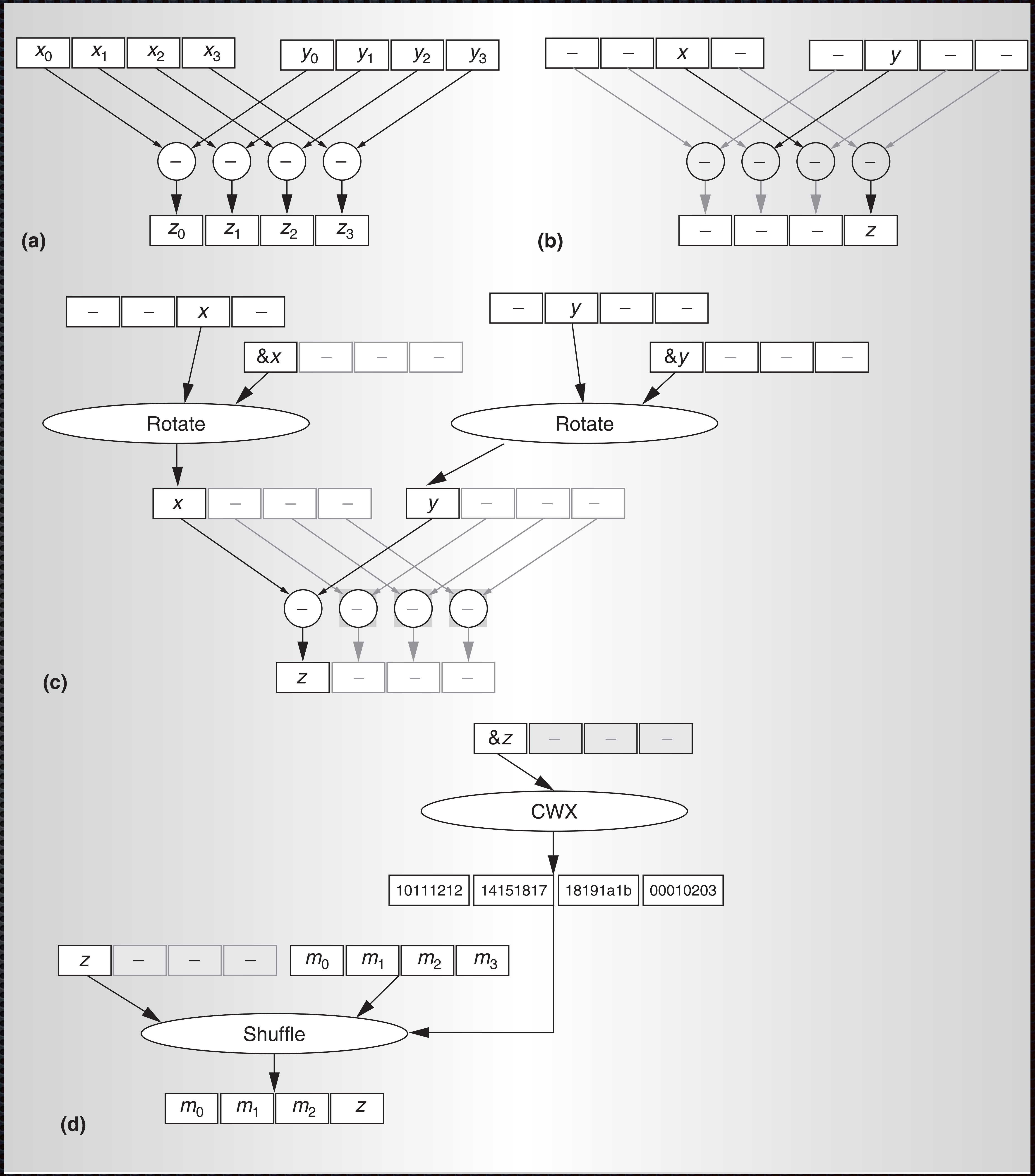
SPE Programming (I)

- SPE has no native scalar processing
  - No scalar registers
  - Registers can hold vectors of ints and floats
- Instead, SPE Scalar processing is folded into vector processing
  - Manual alignment via software
  - Aim is for compiler to pick appropriate alignment

SPE Programming (II)

- Scalar Layering
  - Sequential scalar operations on a vector machine
  - Large register file can help (more scratch space)
- Data-parallel conditional execution
  - Branches expensive in SPEs
  - Minimal Branch Prediction (Hints)
  - Convert if-then to vector select

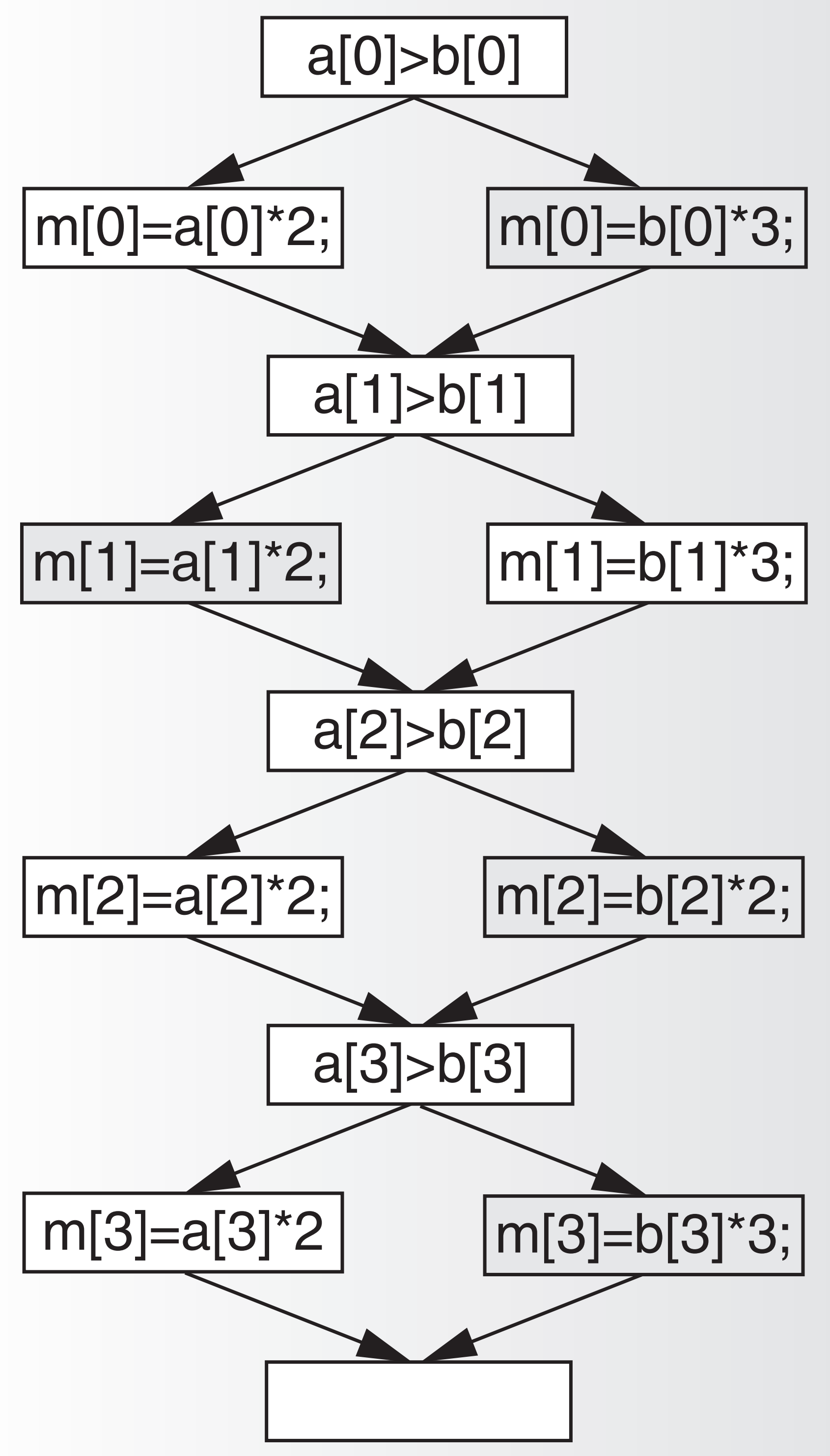
Scalar Layering



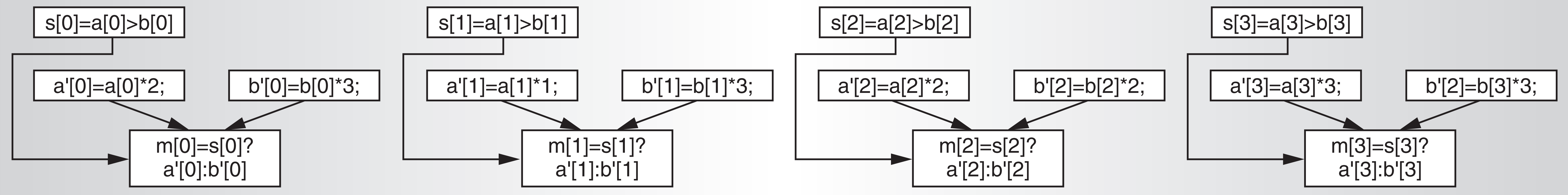
Data-parallel Selection

```
for (i = 0; i< VL; i++)
  if (a[i]>b[i])
   m[i] = a[i]*2;
 else
   m[i] = b[i]*3;
```

(a)



(b)



(c)

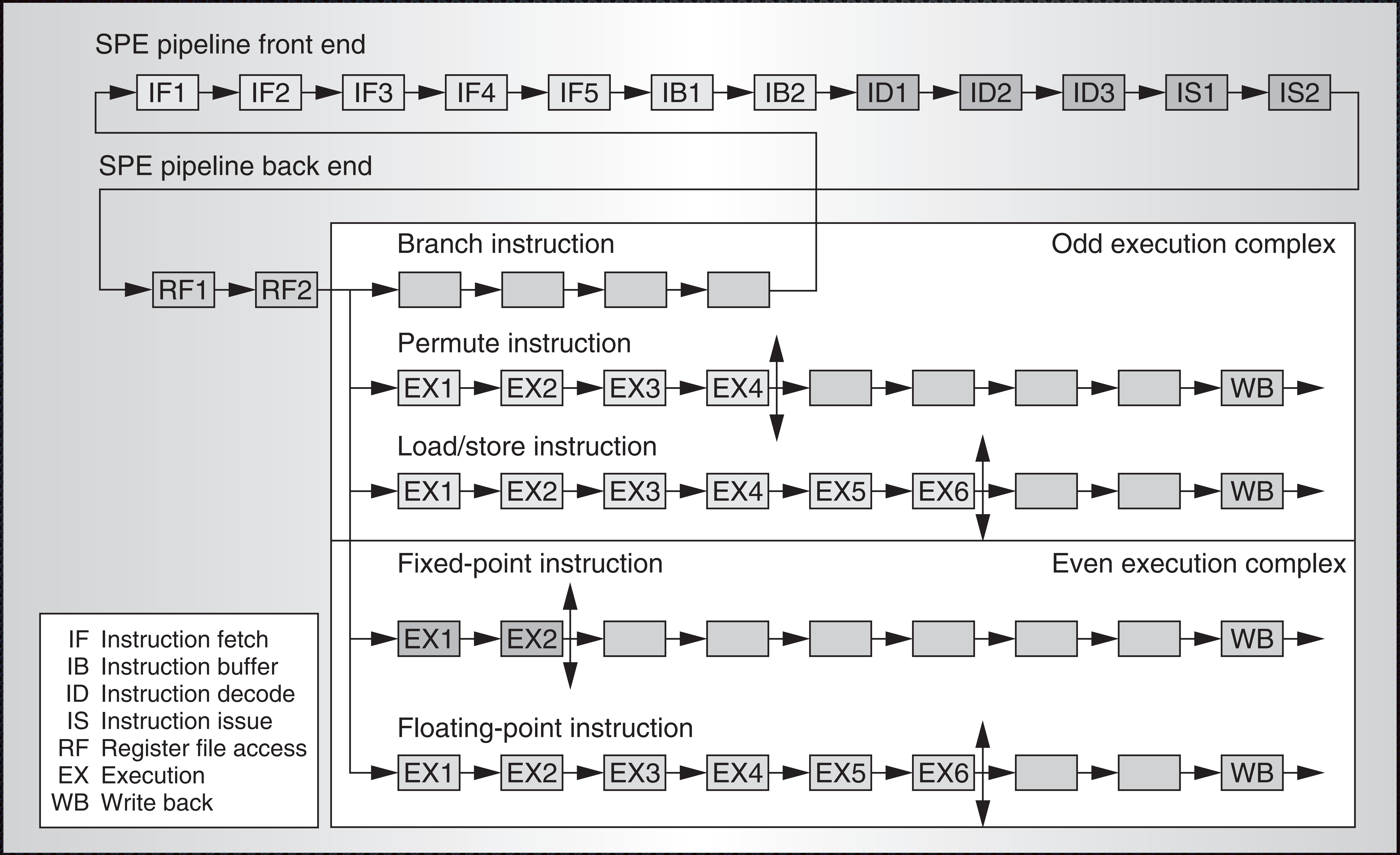
SPE Arithmetic and Local Store

- Emphasis on Non-saturating integer and single-precision FP
- Local Store is NOT a cache
  - Simpler Hardware
  - Deterministic Timing
  - YOU have to perform the DMAs yourself (explicit prefetch for next thread)
- Local Store holds code, too!

SPE Pipelines

- Dual-pipelines, statically scheduled
  - Even = Integer and FP operations
  - Odd = Memory, Branch, and Data Formatting
- Explicit branch prefetcher instruction

SPE Pipelines



IF	Instruction fetch
IB	Instruction buffer
ID	Instruction decode
IS	Instruction issue
RF	Register file access
EX	Execution
WB	Write back

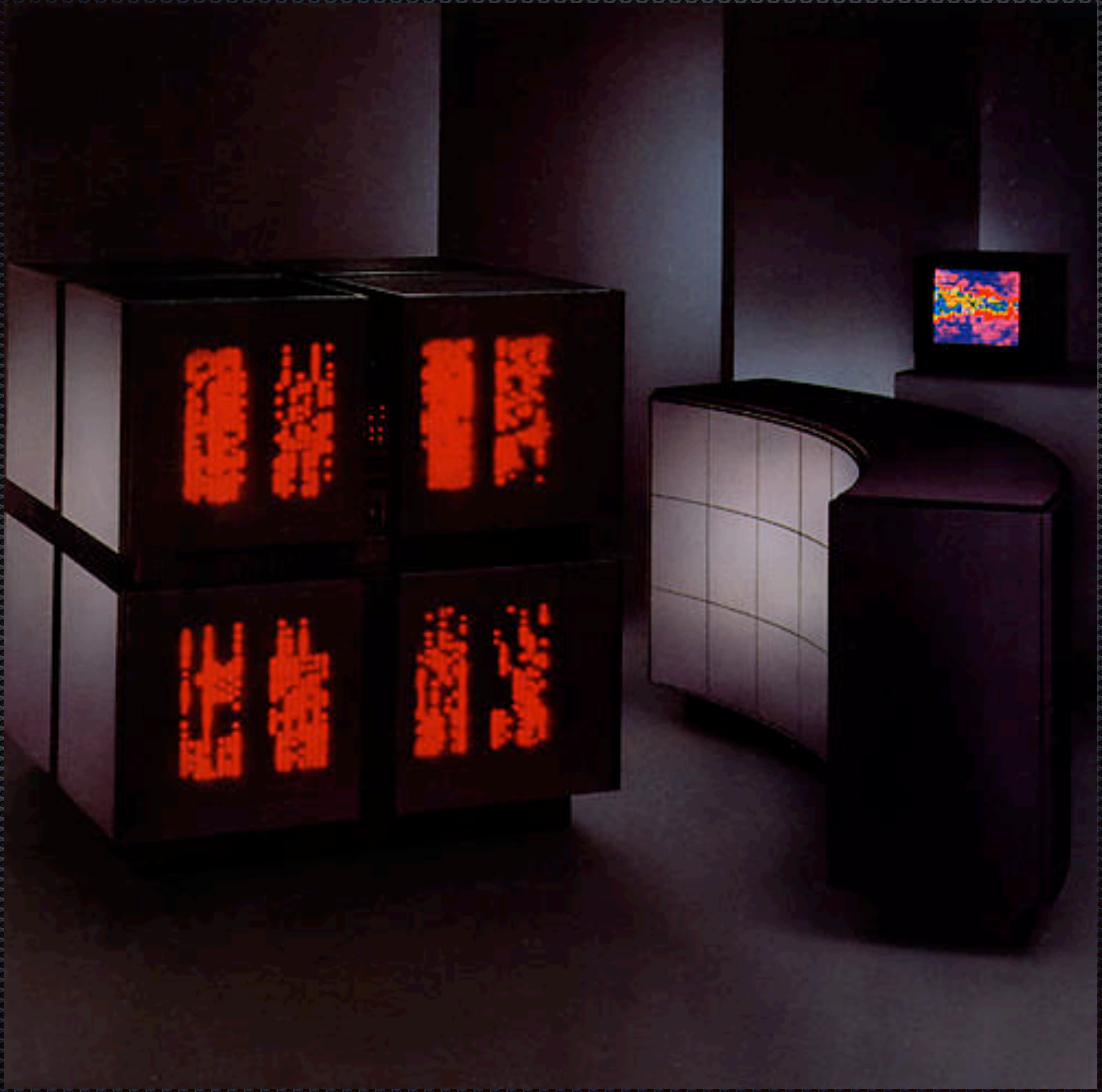
Summary of Techniques

- Turn scalar operations into shuffles
- Turn branches into selects (when possible)
- Plan local store resources carefully
- Balance your pipeline allocations carefully
- The point:
    - Hardware is simpler, but
    - Much more is exposed to the software

Single Instruction Multiple Data

Obvious and simple rarely is either

Bit-Serial SIMD: CM-1 (&2), late 1980’s



Sort of SIMD: CM-5 (SPMD), early 1990's

Maspar (late 1980s)

- Grew out of project at Digital Equipment Corp
- 32 4-bit PEs per chip
- 8-way grid interconnect
- Up to 16K processors

Many others of that ilk

- Goodyear Staran, MPP, ASPRO (bit-serial SIMD)
- CMU/Intel Warp and iWarp systolic arrays
- nCUBE (SPMD)
- Inmos Transputer (CSP array)
- UMass/Hughes IUA
    - heterogeneous bit-serial SIMD, 32-bit SPMD, 32-bit CC-NUMA SMP

Why SIMD?

- Conceptually simple form of parallelism
  - Parallel vector operations common in mathematics
  - Also common in image processing, signal processing, database, etc.
- Serial program with parallel data type and operations
- Efficient silicon implementation (many ALUs under a single control unit)

Why not SIMD?

- Size of problem almost never matches hardware
- If smaller, then a fraction of the array is idle
- If larger, then the array has to be virtualized
    - Virtualization has to swap contexts
    - Handle communication across virtual tiles boundaries
    - More fractional arrays at edge of virtual array

Why not SIMD?

- Branches have to serialize — times are additive
- IF (A < B) implies elements that meet the condition take the branch, those that fail take the ELSE clause
- Only one source of instructions for all elements
  - Select A<B, issue instructions
  - Select A>=B, issue instructions
- Multiway branches further divide the elements

Why not SIMD?

- Instruction distribution is hard to scale up
- Assumes a globally synchronous clock
- Broadcast has to be balanced for simultaneous arrival
- Instruction generation takes multiple operations, so has to run faster than consumption, which is simpler
  - Makes clock scaling difficult
- Scaling in size consumes much more power

Why not SIMD?

- Collective operations can be slow
- Need feedback from computations for global branches
- Fan-in from thousands to millions of elements is a multi-stage process (can be pipelined but doesn’t hide latency)
- Array either sits idle during collective, or control is much more complex

Why not SIMD?

- Context switching is expensive
- Elements typically lack enough space to hold multiple contexts
- Entire array has massive amount of data
- Switching moves whole context out, new context in
- Array is idle for long period during context switch

Why not SIMD?

- Can only address some issues by multithreading
  - Hide collective delay, hide issue latency
- Local expansion of instructions
  - Allows asynchronous clocking, communication
  - Needs rate buffers between asynchronous sections
  - More complex hardware, fewer elements
- Optimal use exposes threading in programming model

GPU Architecture

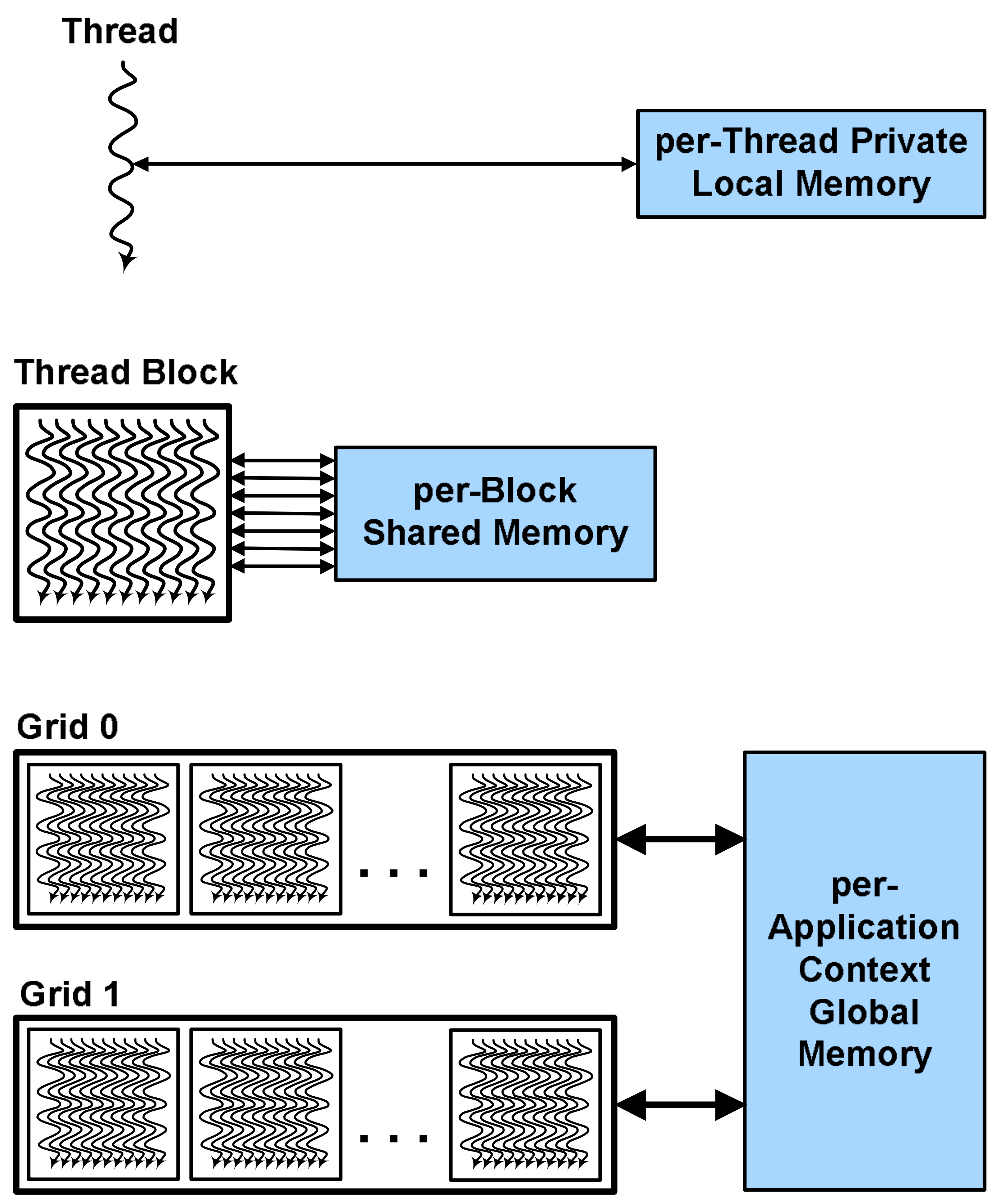
Not just for graphics any more

Fermi: NVIDIA’s Next Generation CUDA Compute Architecture

NVIDIA 2009

CUDA Programs

- Kernel = Parallel CUDA Program
- Thread = basic unit of processing
  - operates on Kernels
  - Private Local Memory
- Thread Block
  - Set of threads
  - Shared Memory between threads
- Grid
  - Array of Thread Blocks
  - Shared Global Memory


Thread
per-Thread Private Local Memory
Thread Block
per-Block Shared Memory
Grid 0
Grid 1
per-Application Context Global Memory

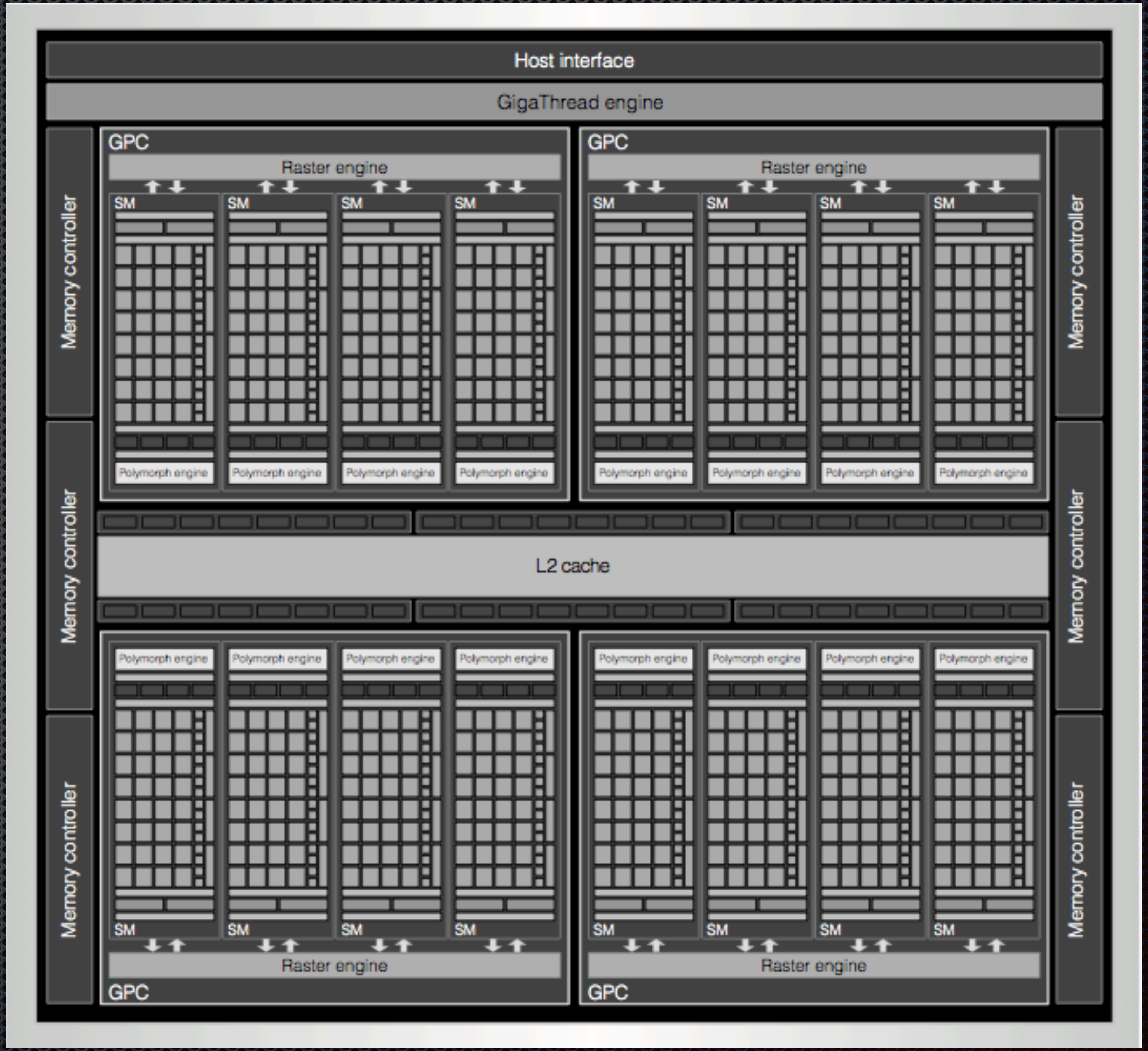
CUDA Hierarchy of threads, blocks, and grids, with corresponding per-thread private, per-block shared, and per-application global memory spaces.

The Fermi Architecture

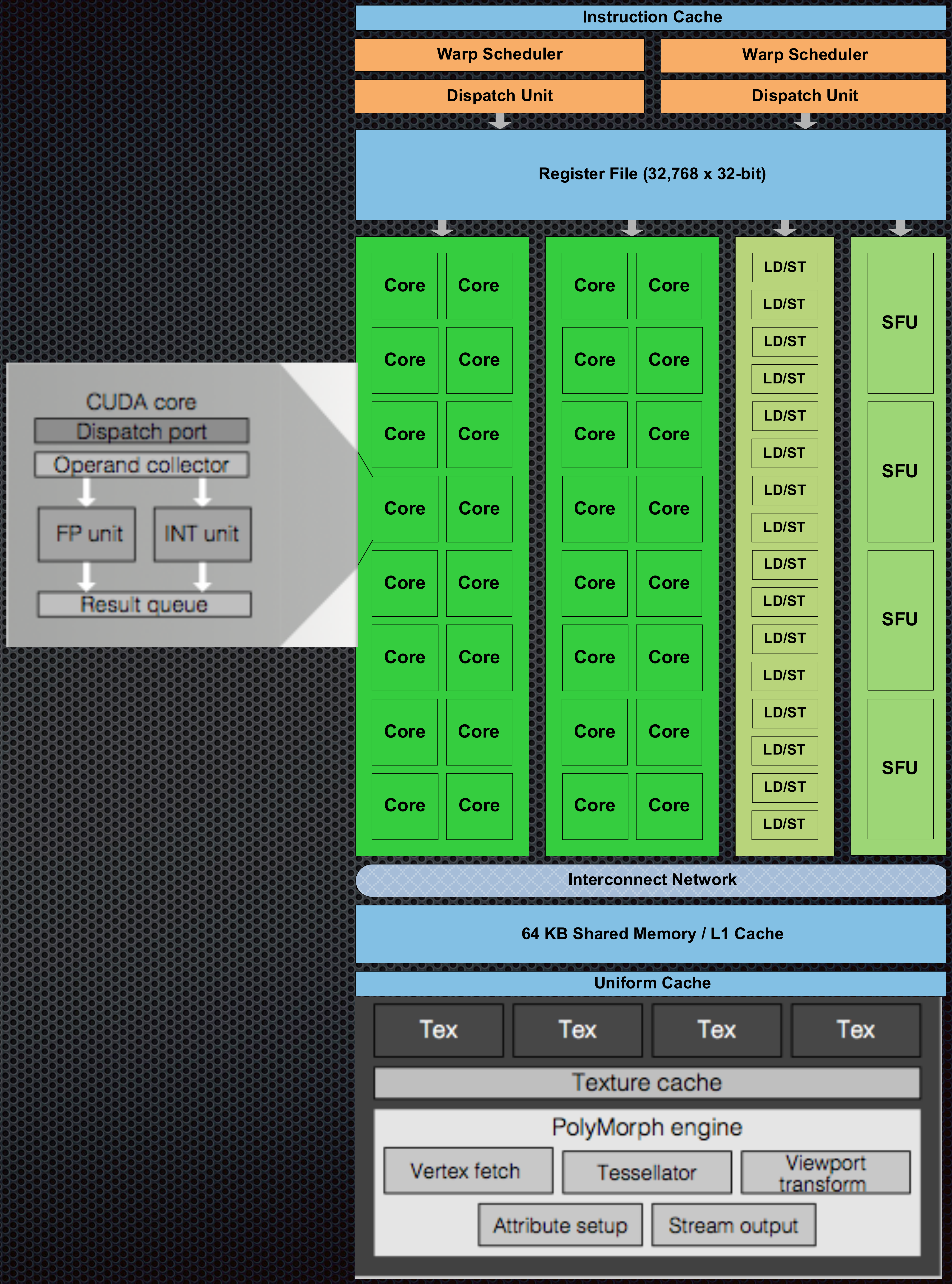
- GPU = executes grids (16 SMs in total)
- SM = executes thread blocks
  - Group of 32 threads = warp
  - Each SM has 32 CUDA cores (512 total)
  - 16KB L1 plus 48KB shared or 48KB L1 plus 16KB
  - 16 Load/Store Units
  - 4 Special Function Units

Overall Architecture

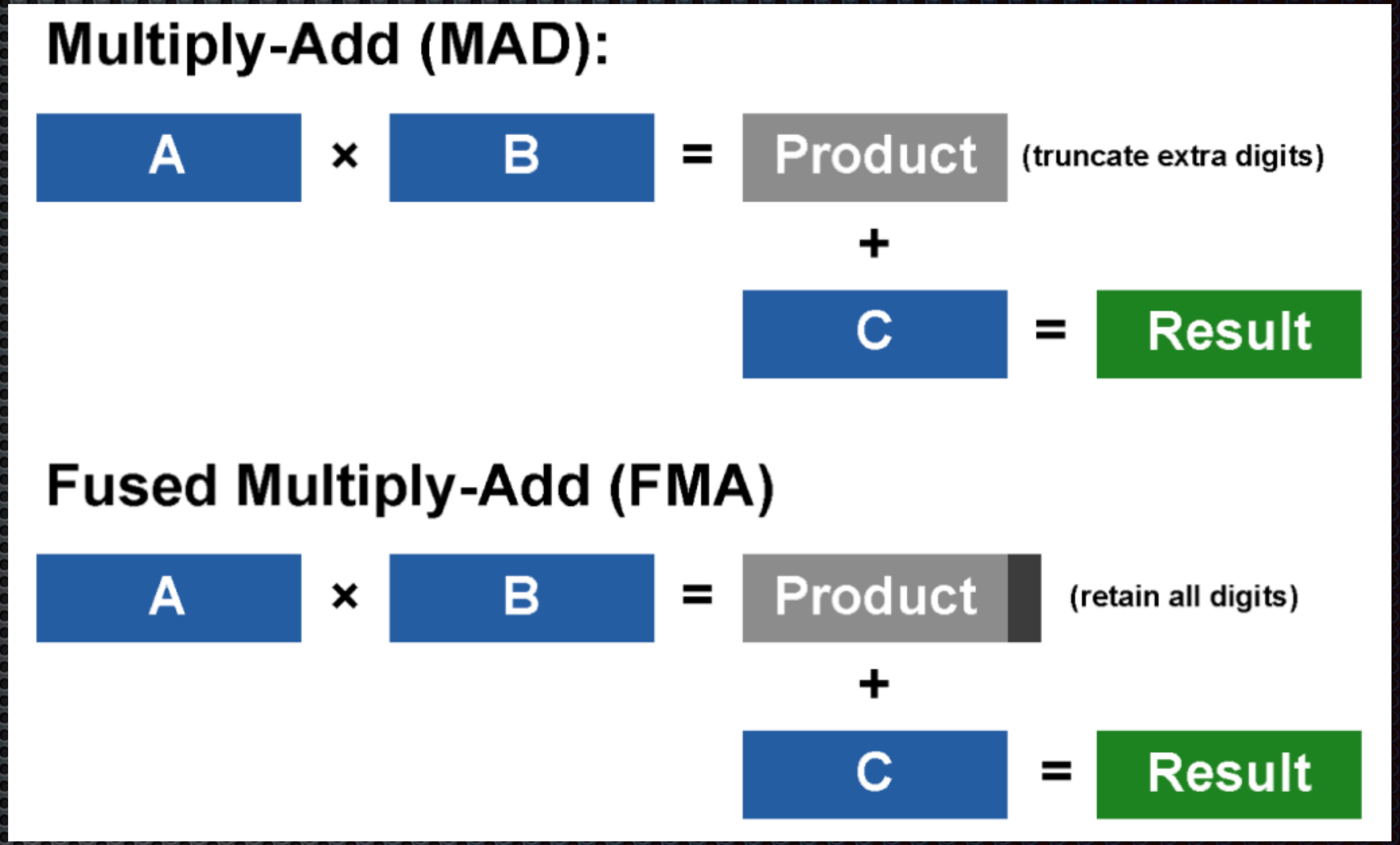
- 4 GPCs
- 4 SMs per GPC
- L2 Cache
- Thread engine
- Host and memory interfaces



SM Structure



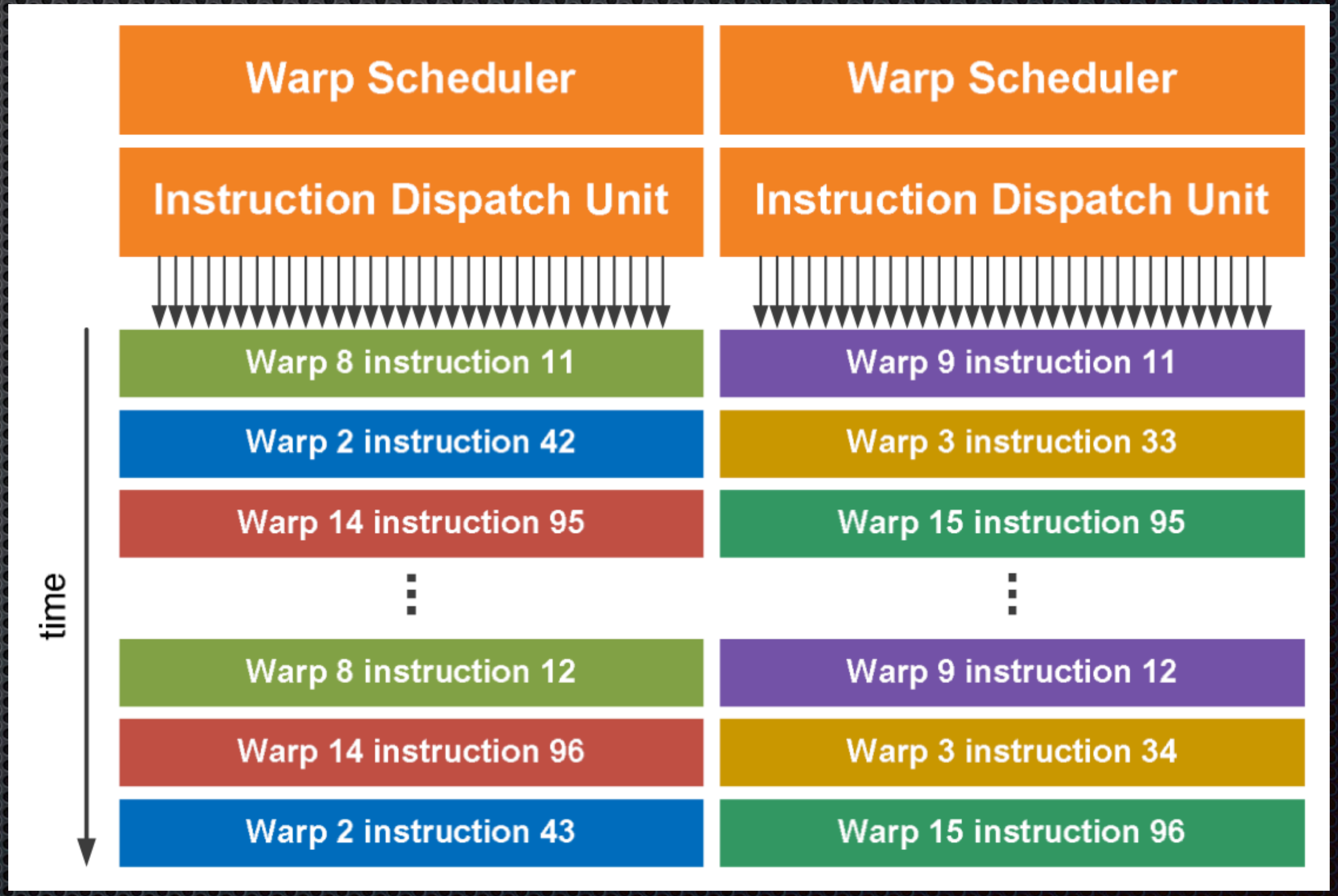
New Features of Fermi



- Double Precision Support, Fused Multiply-Add (FMA)
  - Better Scientific Precision, 8x faster than GT200
- 32-bit Integer Support (formerly 24 bit via FP mantissa unit)
- Configurable 64KB Shared Memory/L1 Cache, 768KB shared read/write L2 with writeback (GT200 cache was read-only)
- Additional ISA support for C++

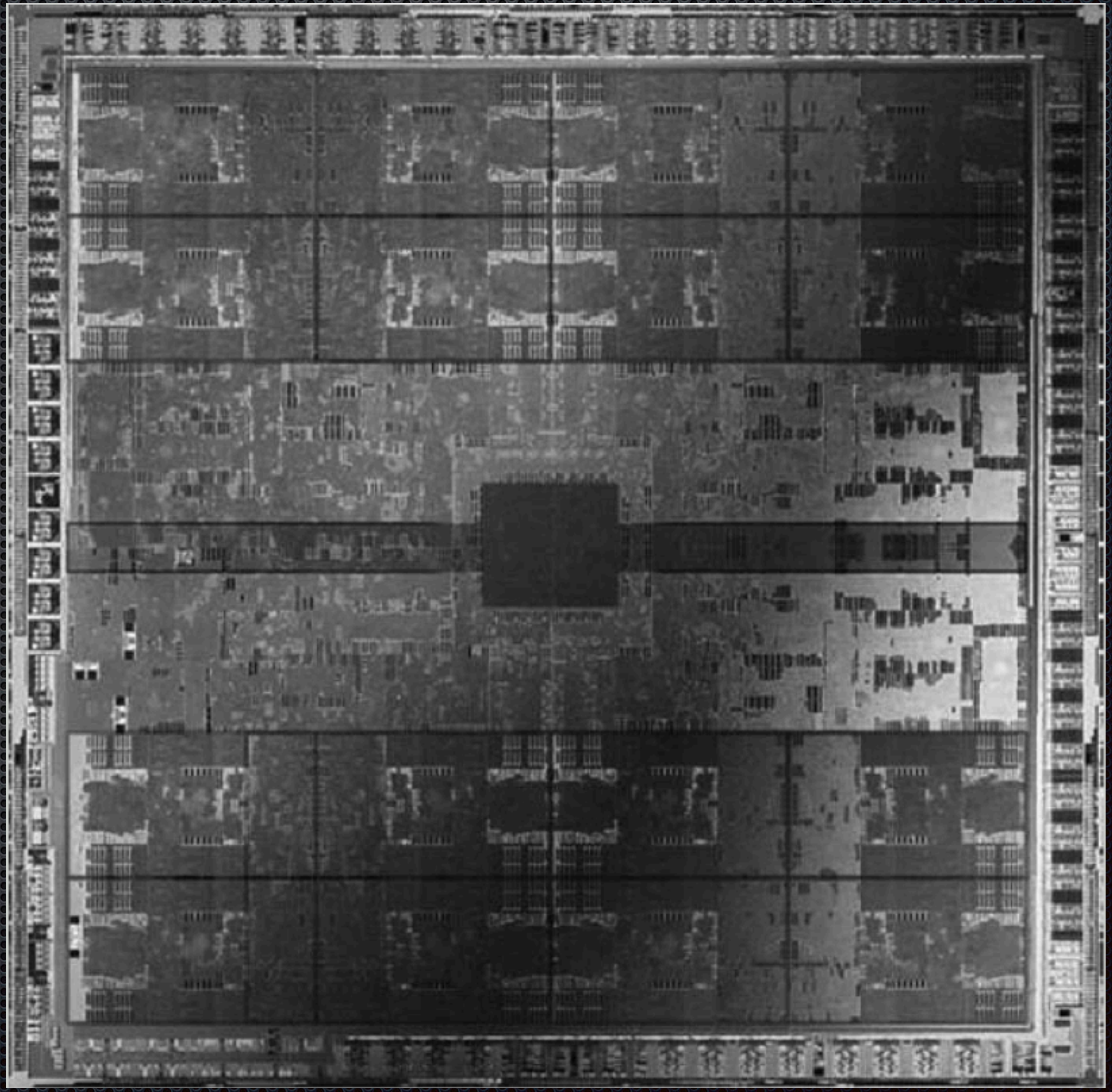
New Features of Fermi (II)

- Unified (40 bit) Addressing Space
  - Uniform Pointer Manipulation, multiple page sizes
- ECC Support
  - Reliability
- Fast Atomic Operations
  - Shared Data Structures
- Improved Scheduling
  - Dual Warp
  - Heterogeneous Kernels

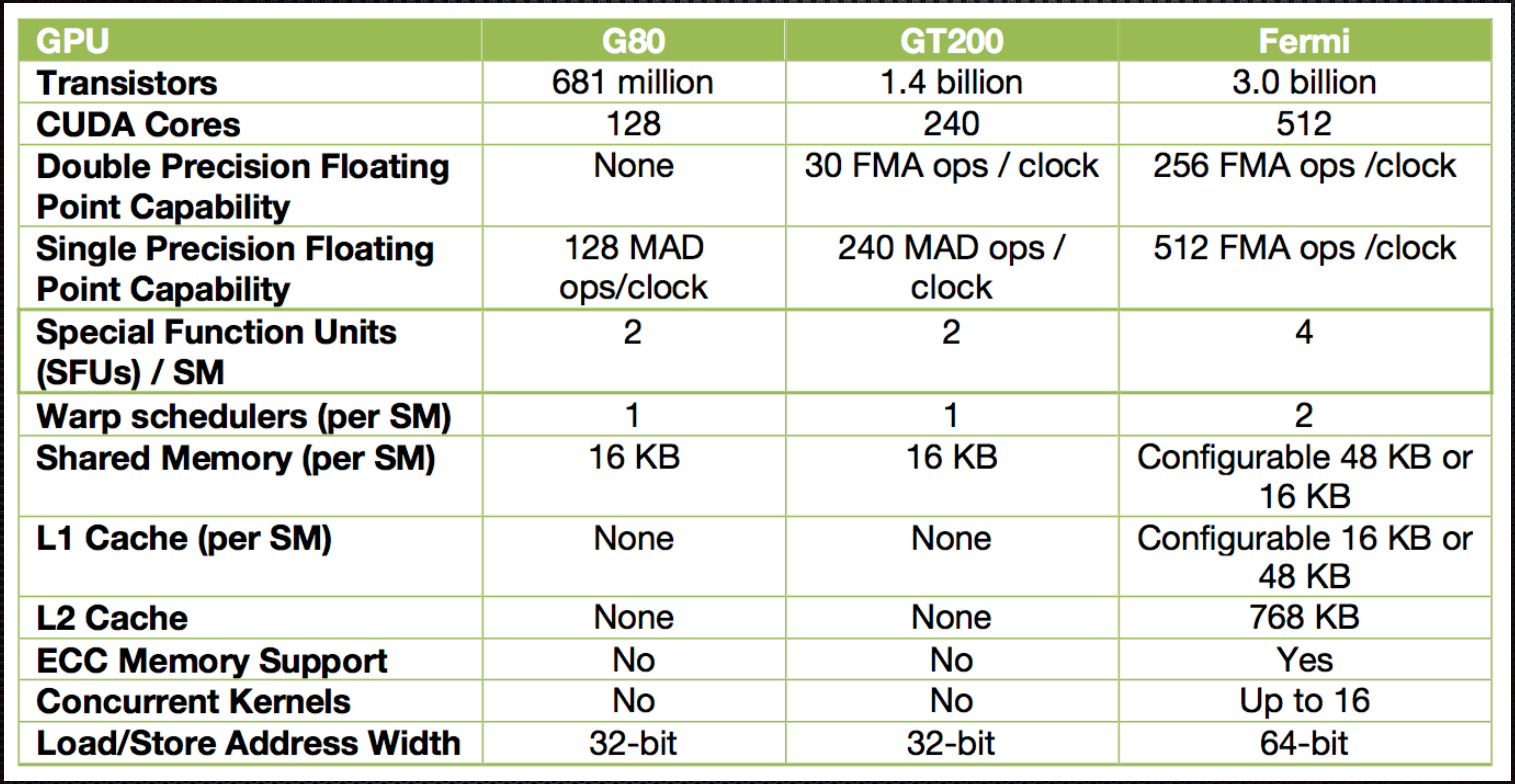


GF100 Chip

512 Cores
3B transistors
40nm process
GDDR5 memory
6 memory controllers
4 GPU clusters
“Gigathread” engine
Shared L2 cache



Comparison to Prior GPUs



GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Special Function Units (SFUs) / SM	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

Functionality

- At least one SM disabled on all models
- Graded, less expensive, parts with various numbers of SMs and memory controllers disabled
- Higher end Tesla versions have ECC memory enabled, double precision floating point, limited video out
- GF110 (GTX580) revision reduces heat and enables all SMs to operate

GP Clusters

- Rasterizer in each GPC
- Four streaming multiprocessors (SMs)
- 16 PolyMorph engines per SM (64 per GPC)
  - Fixed and configurable logic for tessellation support
- Fermi adds tessellation (DX11) support, z-compare and blend raster, better physics processing, distributed rasterization

Performance Growth of Shader vs. Geometry

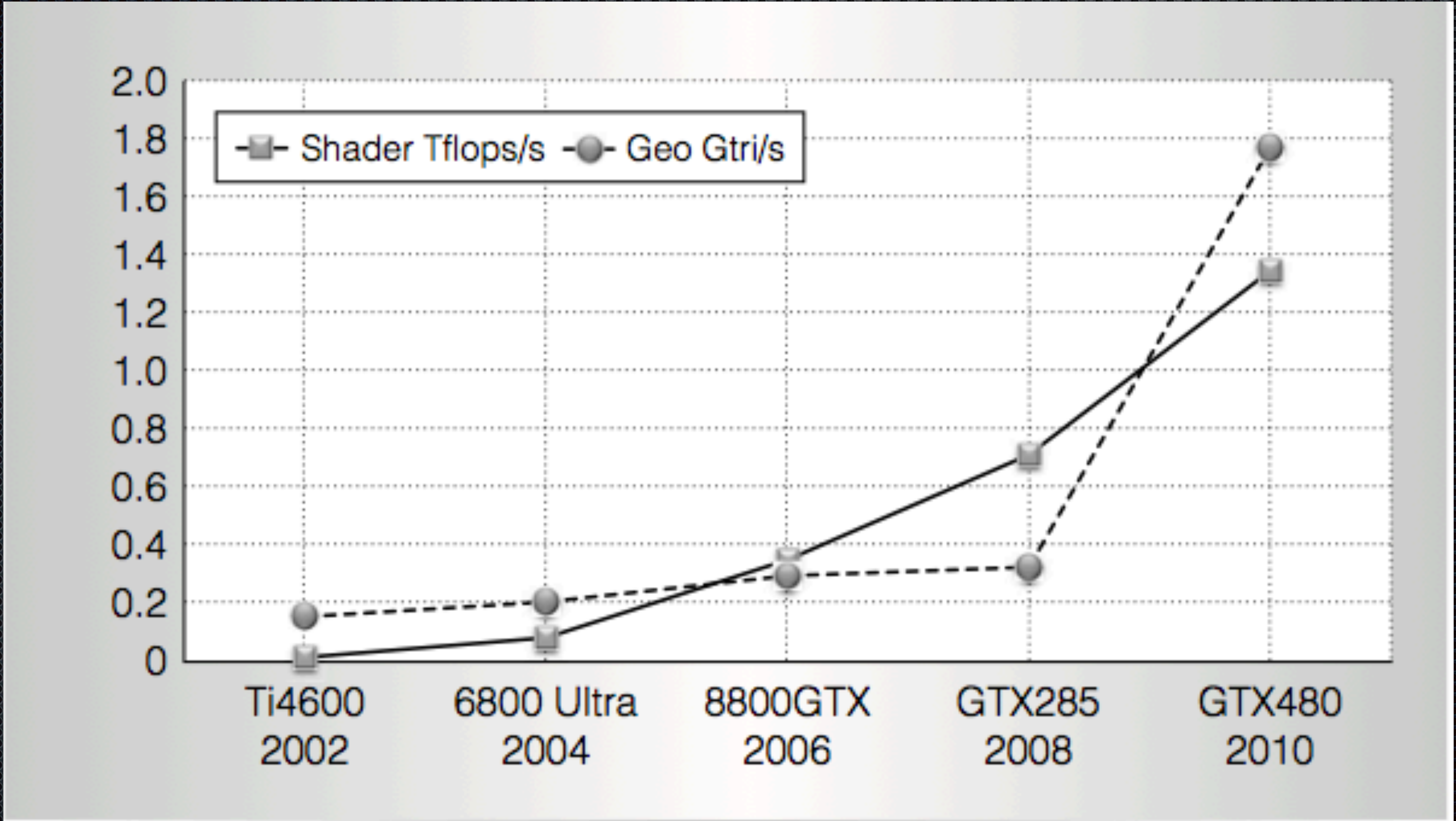


Figure 7. GPU generations showing shader horsepower in teraflops per second and geometry horsepower in giga-triangles per second.

Scalability

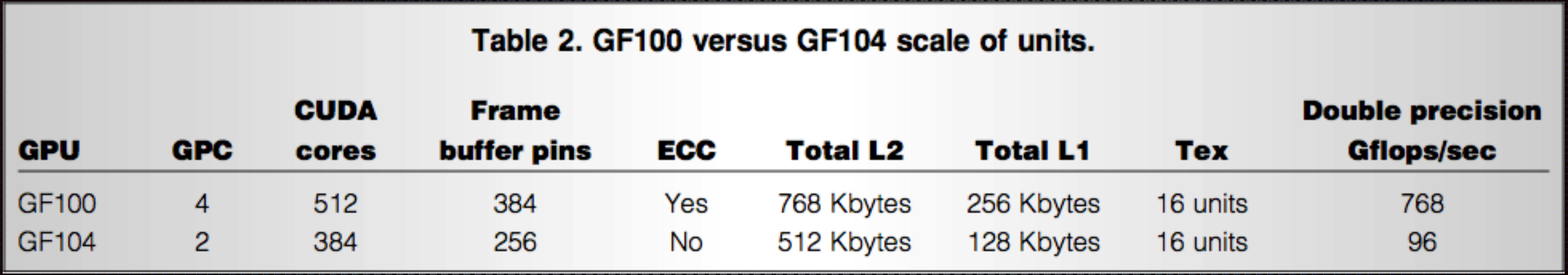


Table 2. GF100 versus GF104 scale of units.

GPU	GPC	CUDA cores	Frame buffer pins	ECC	Total L2	Total L1	Tex	Double precision Gflops/sec
GF100	4	512	384	Yes	768 Kbytes	256 Kbytes	16 units	768
GF104	2	384	256	No	512 Kbytes	128 Kbytes	16 units	96

- Distributed rasterization, L2 shared cache, more compartmentalization enable easy scaling of processor configurations. GF104 is a consumer-oriented version of GF100

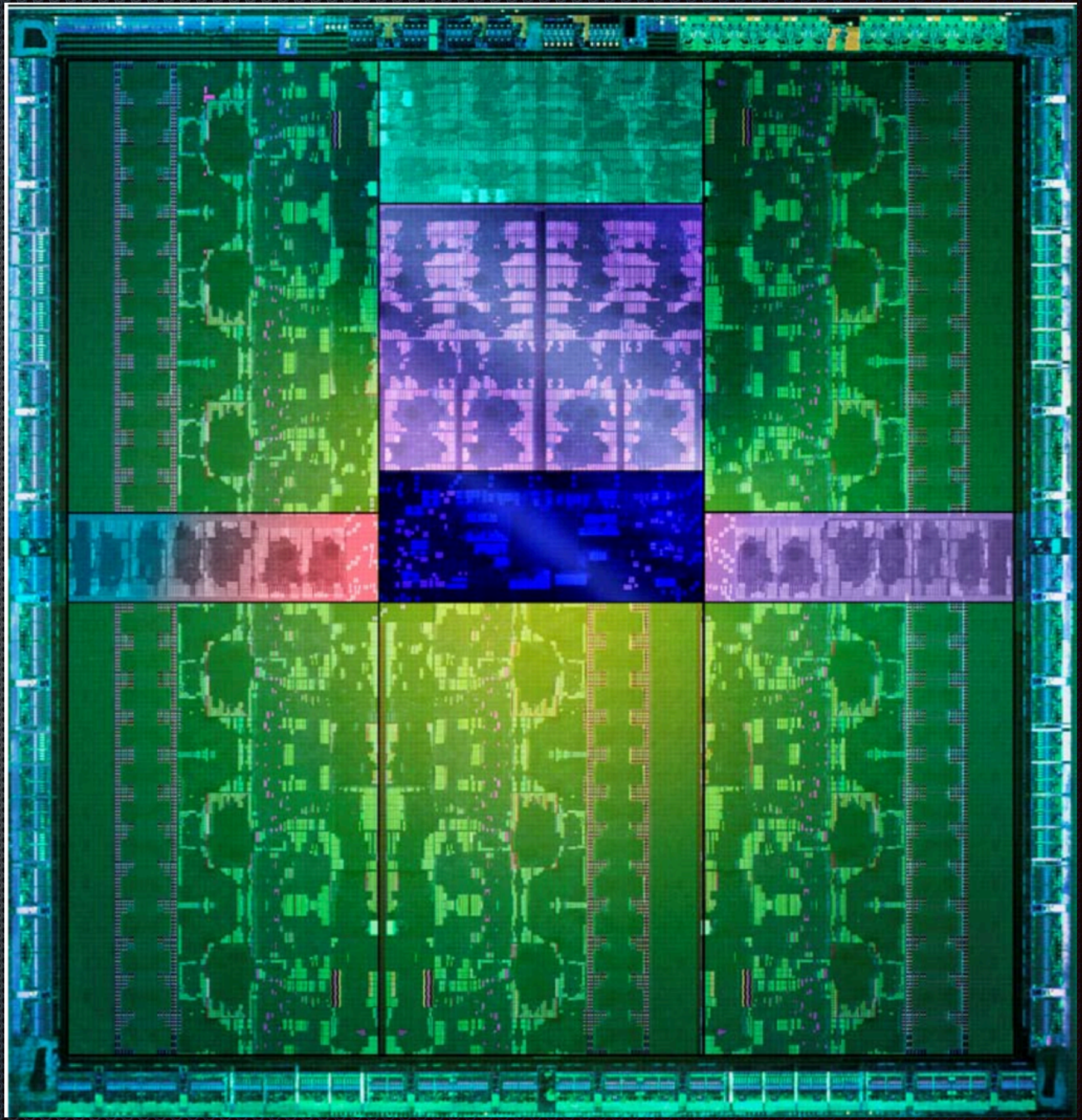
Discussion

Kepler: NVIDIAs Next Generation CUDA Compute Architecture

NVIDIA 2012

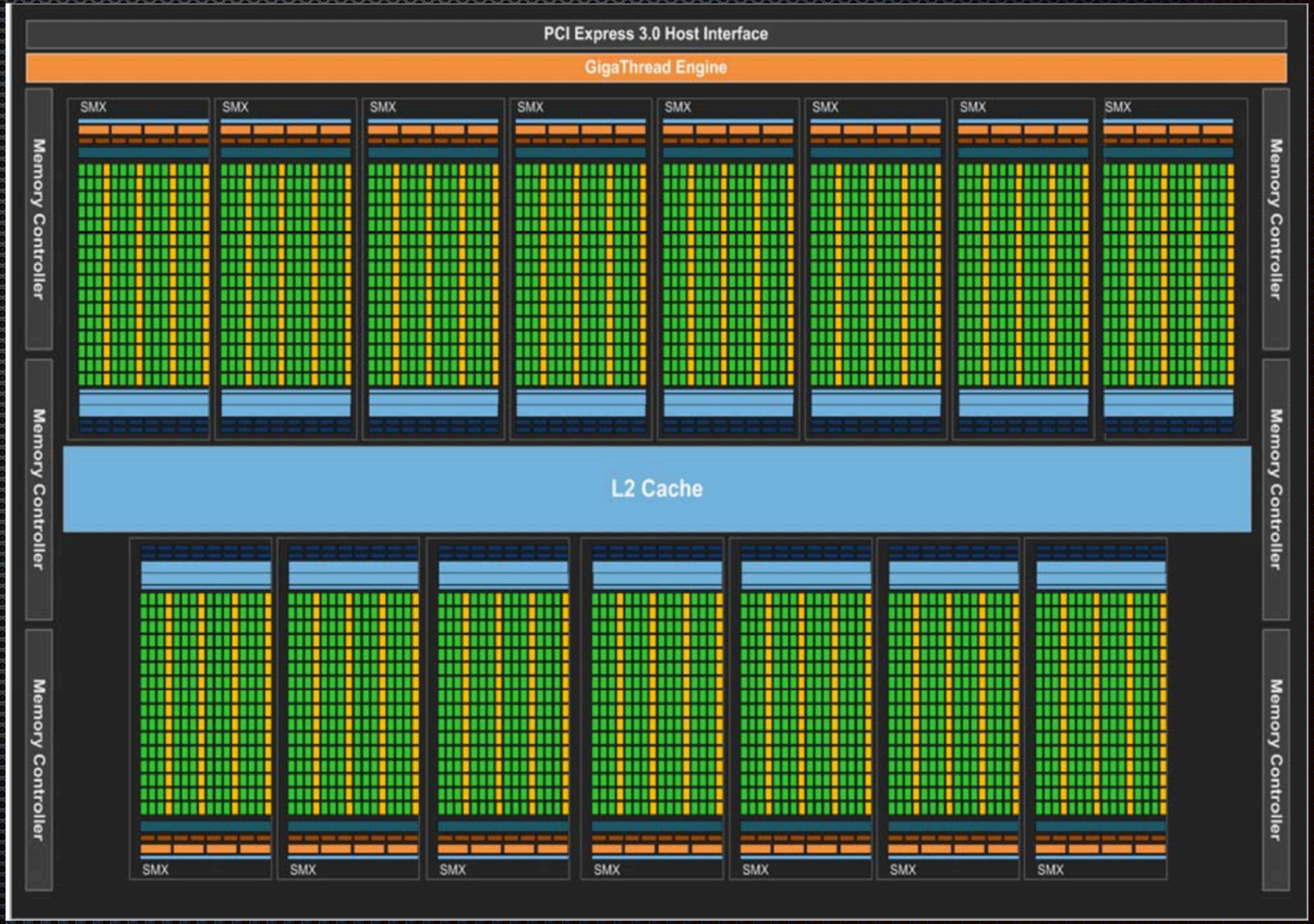
GK110 Chip

- 7.1B transistors
- 1TFLOPS IEEE 64bit
- 225W power
    - 3X more efficient
- 28nm process
- Designed for Tesla (GPGPU cards)

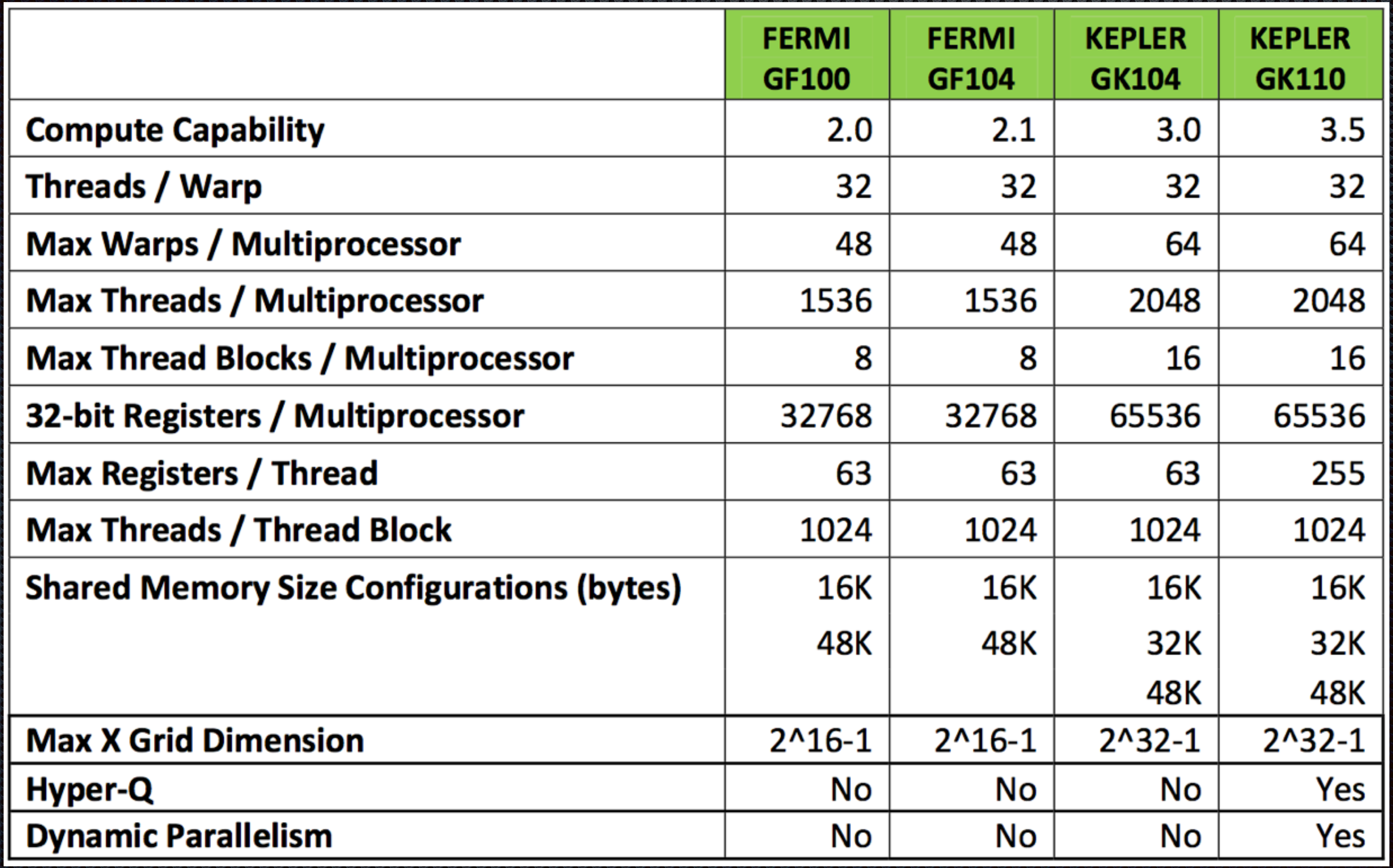


Chip Architecture

- More warps (64 vs 48)
- SMX architecture
- 64K registers/SMX (vs 32k), 15 SMX units
- Up to 255 regs/thread
- Same 48K shared memory/SMX
- Shifts memory model



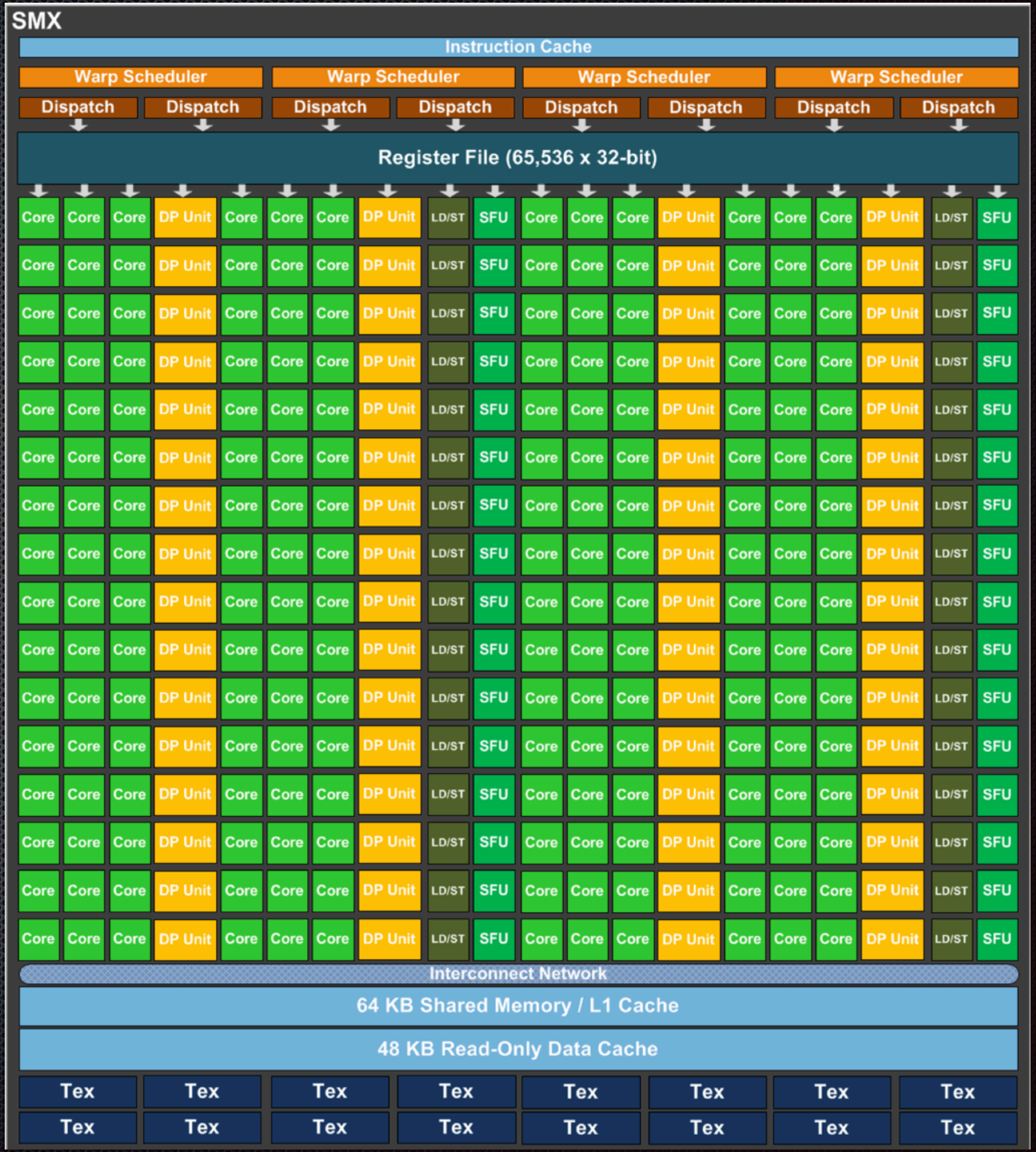
Comparison to Prior GPUs



	FERMI GF100	FERMI GF104	KEPLER GK104	KEPLER GK110
Compute Capability	2.0	2.1	3.0	3.5
Threads / Warp	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16
32-bit Registers / Multiprocessor	32768	32768	65536	65536
Max Registers / Thread	63	63	63	255
Max Threads / Thread Block	1024	1024	1024	1024
Shared Memory Size Configurations (bytes)	16K 48K	16K 48K	16K 32K 48K	16K 32K 48K
Max X Grid Dimension	2^16-1	2^16-1	2^32-1	2^32-1
Hyper-Q	No	No	No	Yes
Dynamic Parallelism	No	No	No	Yes

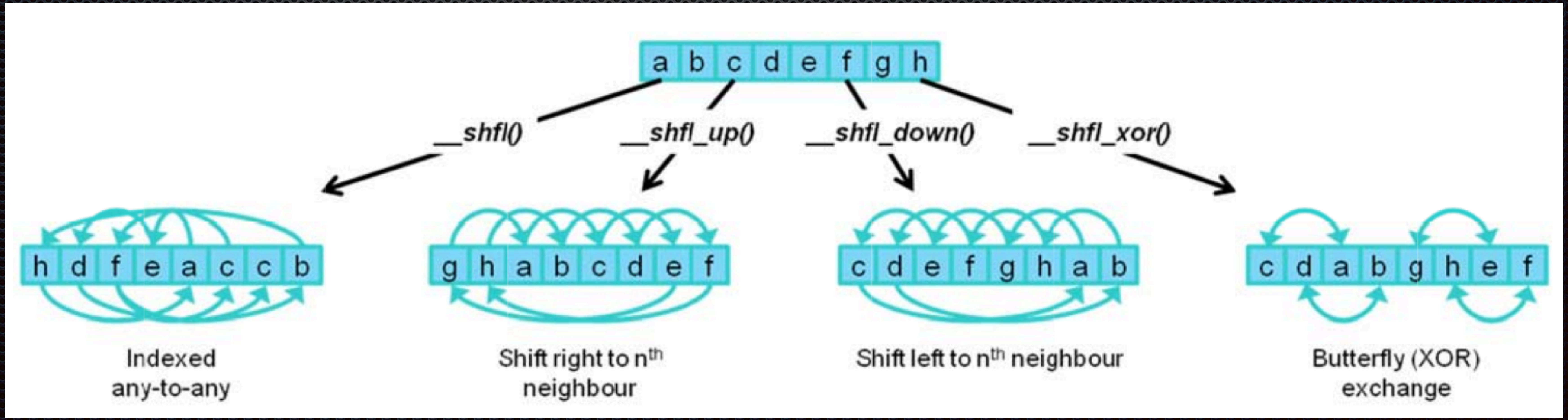
SMX Architecture

- 4 dual-warp schedulers
- Double Precision unit per three SP cores
- Load/Store and SFU per six cores
- Lower clock rate
- DP can now issue with other instructions

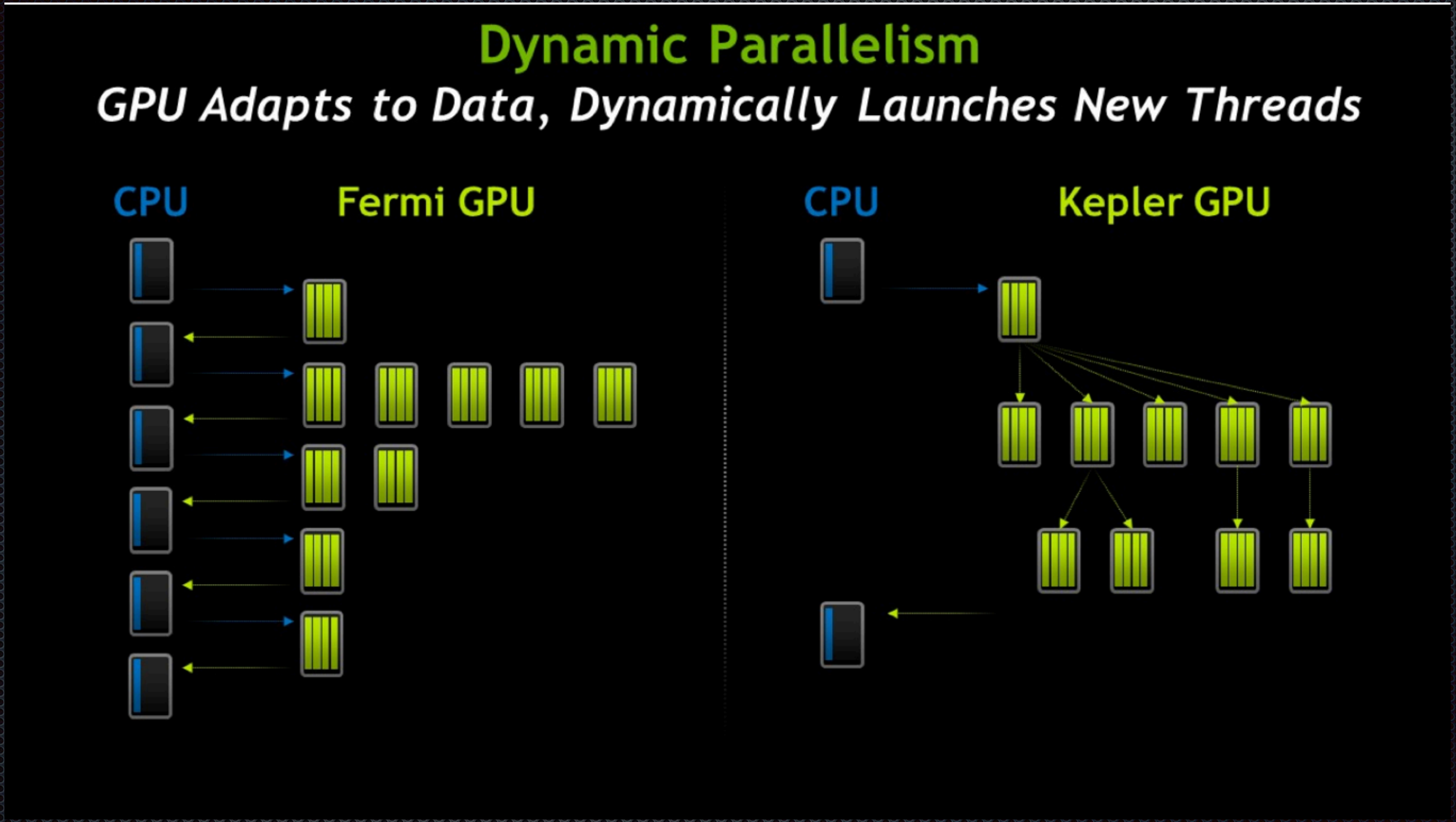

SMX
Instruction Cache
Warp Scheduler
Dispatch
Register File (65,536 x 32-bit)
Core
DP Unit
LD/ST
SFU
Interconnect Network
64 KB Shared Memory / L1 Cache
48 KB Read-Only Data Cache
Tex

New ISA Elements

- 255 regs per thread
- More atomic ops
- Dynamic parallelism, Hyper-Q, GPU Direct
- Shuffle instruction (faster transpose, FFT, etc.)

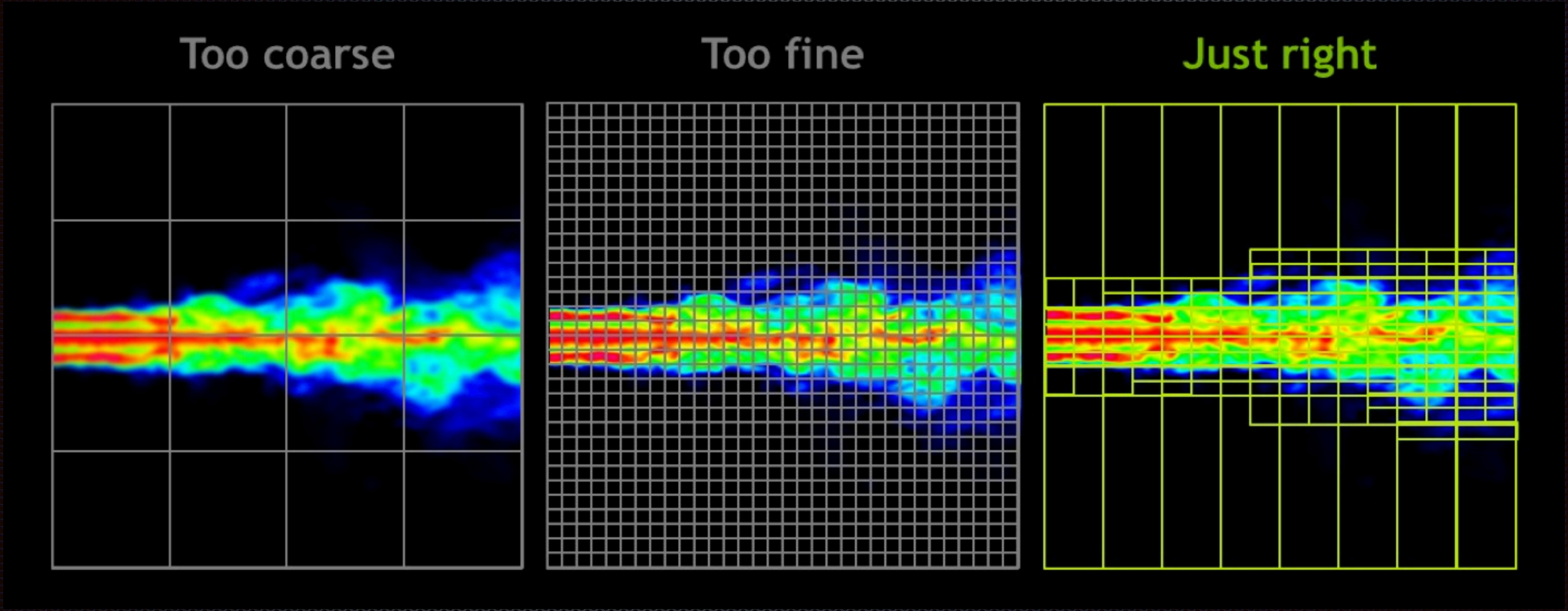


Dynamic Parallelism

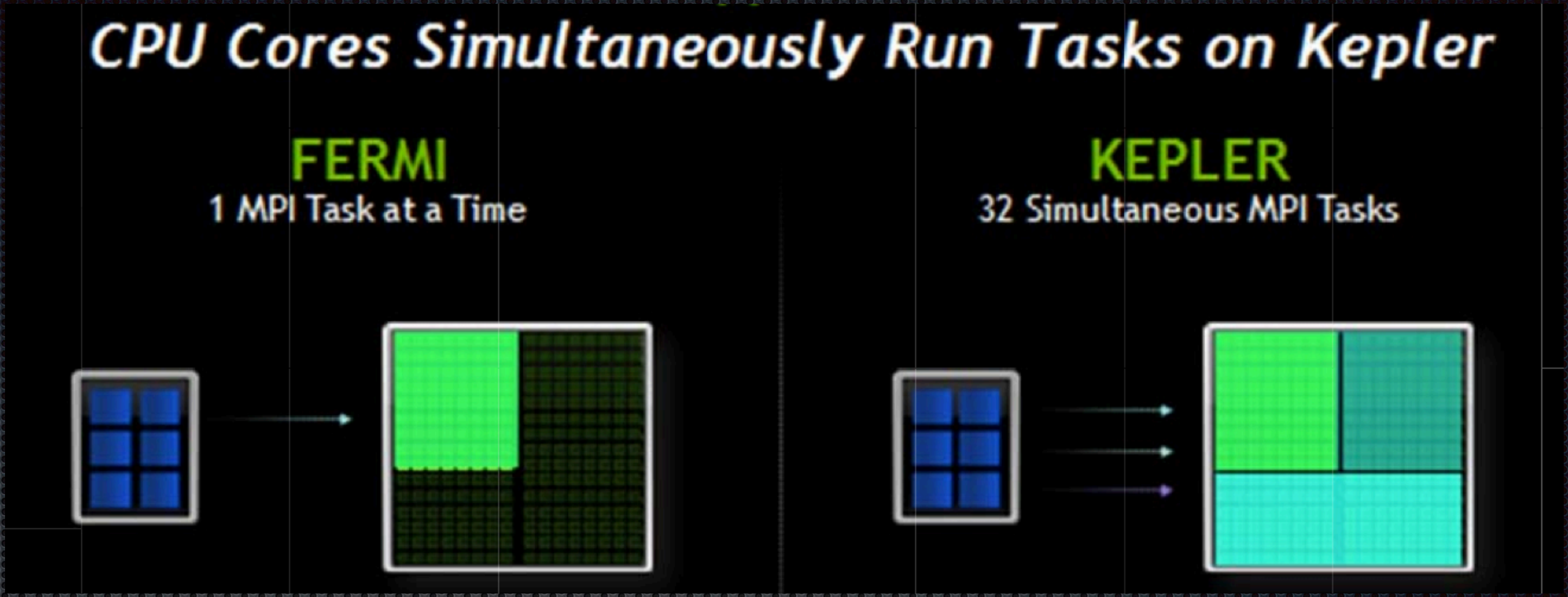


Kernels can now launch other kernels without CPU interaction

Fluid Flow Multigrid Example

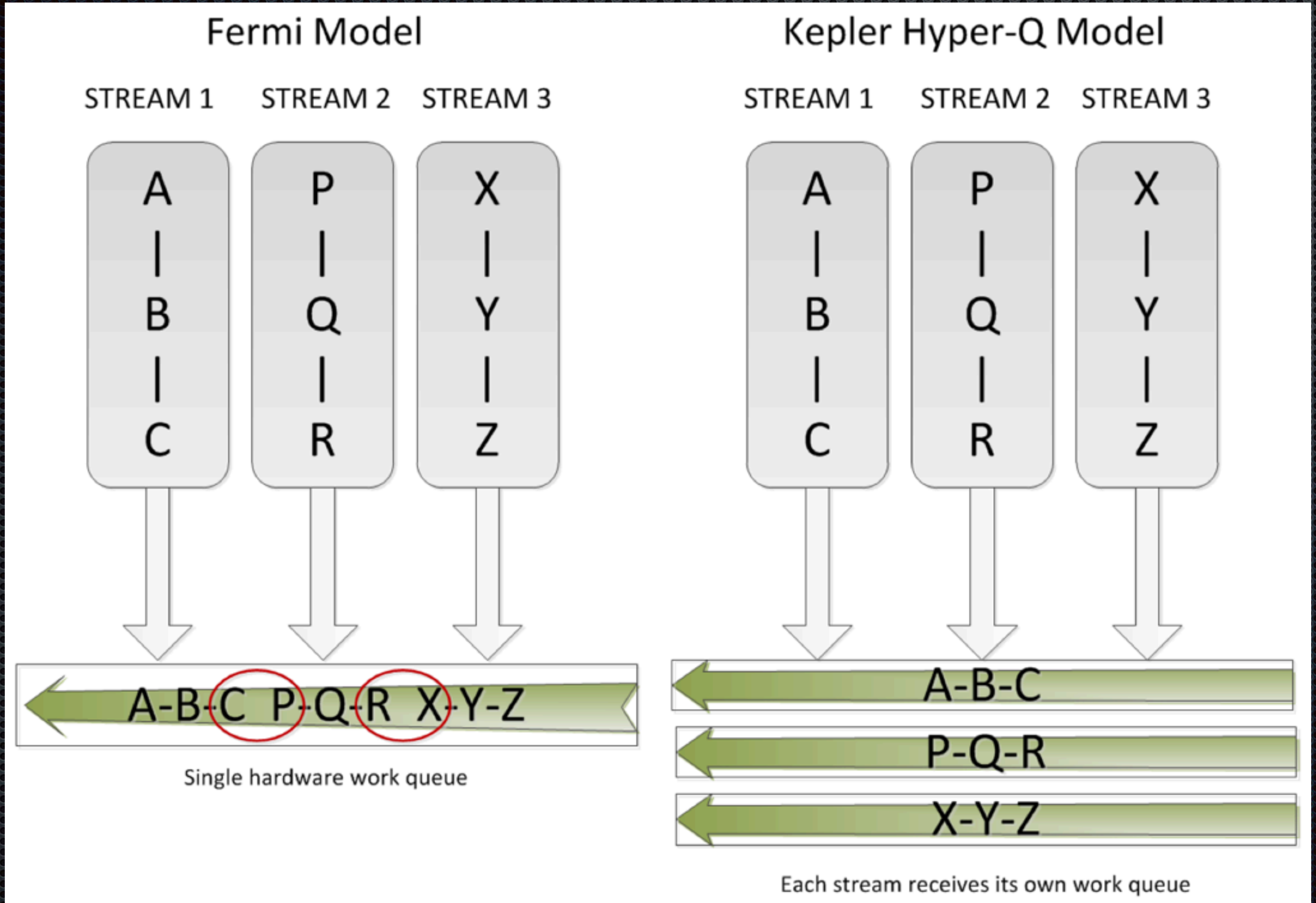


Hyper-Q


CPU Cores Simultaneously Run Tasks on Kepler
FERMI
1 MPI Task at a Time
KEPLER
32 Simultaneous MPI Tasks

- Supports up to 32 simultaneous CPU connections
- Designed for multi-core shared use of GPU
- Previously there was just one work queue

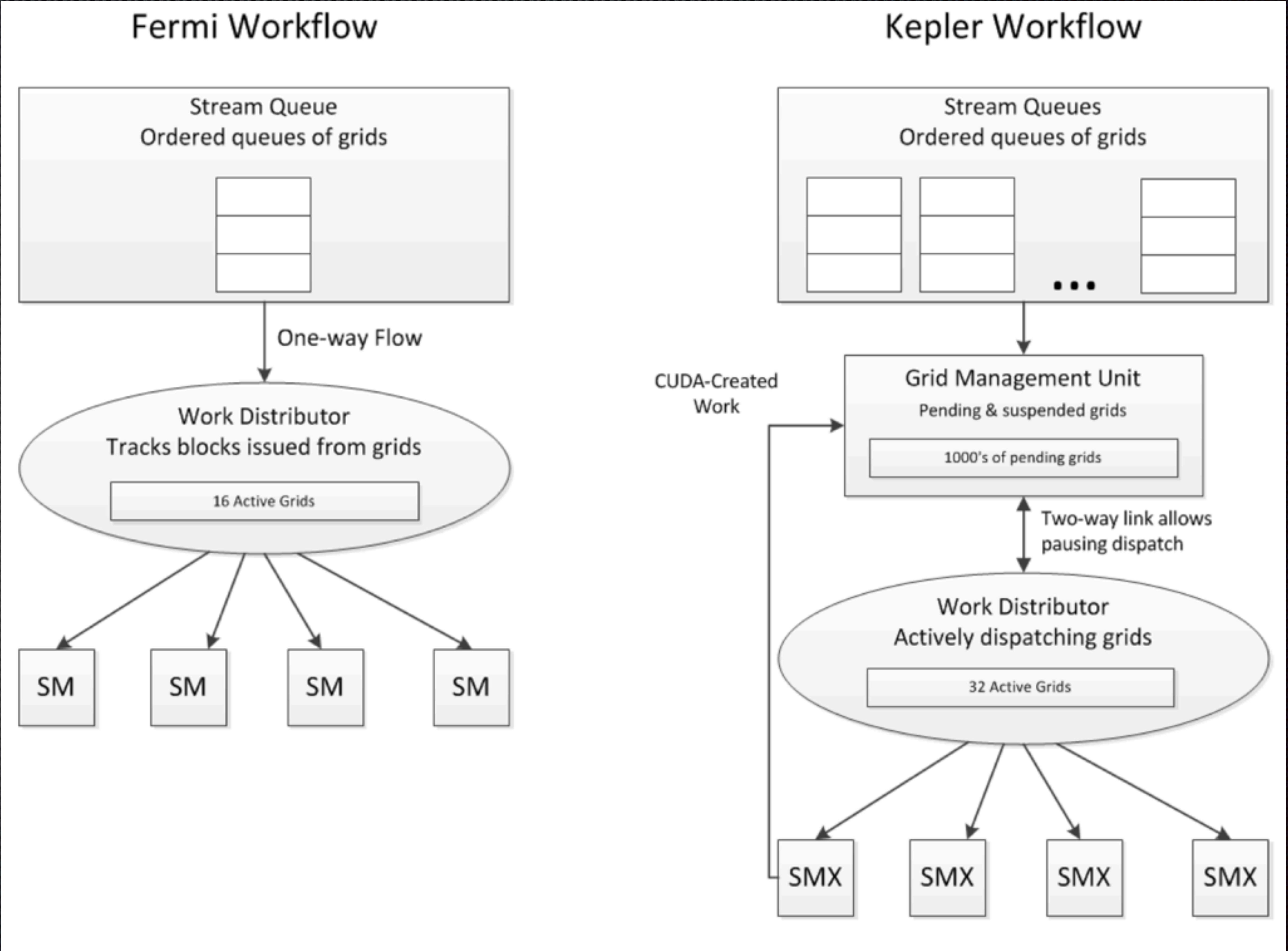
Kepler vs Fermi Work Queue



Hyper-Q working with CUDA Streams: In the Fermi model shown on the left, only (C,P) & (R,X) can run concurrently due to intra-stream dependencies caused by the single hardware work queue. The Kepler Hyper-Q model allows all streams to run concurrently using separate work queues.

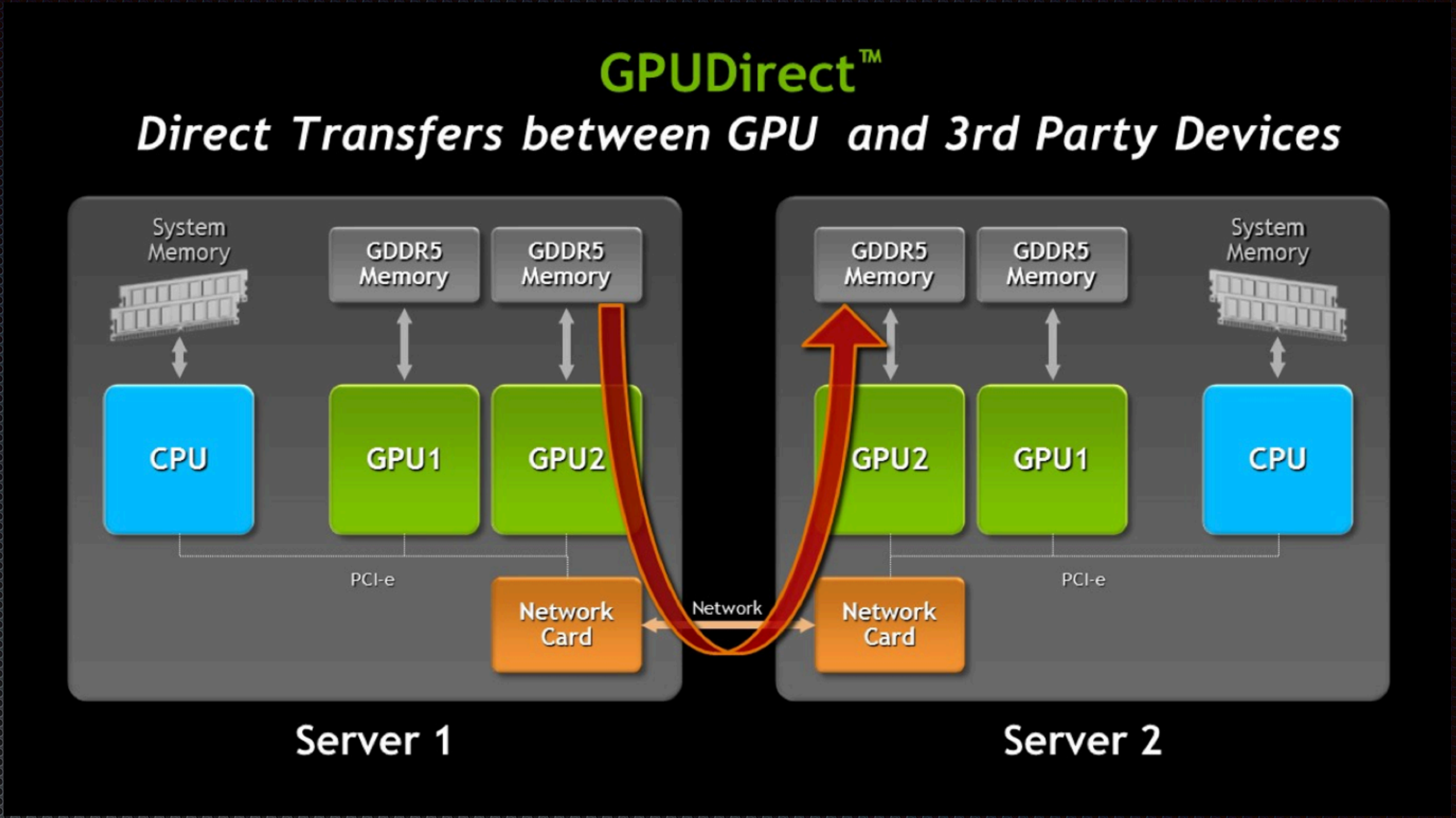
Grid Management Unit

- Dynamic parallelism means kernels can be launched by the GPU and the CPU
- Requires more complex dispatch control with pausing



The redesigned Kepler HOST to GPU workflow shows the new Grid Management Unit, which allows it to manage the actively dispatching grids, pause dispatch, and hold pending and suspended grids.

GPU Direct



- Inter-GPU communication previously went through CPU
- GPU-enhanced clusters need direct link
- Also support for PCI-e GPU-GPU communication

Discussion

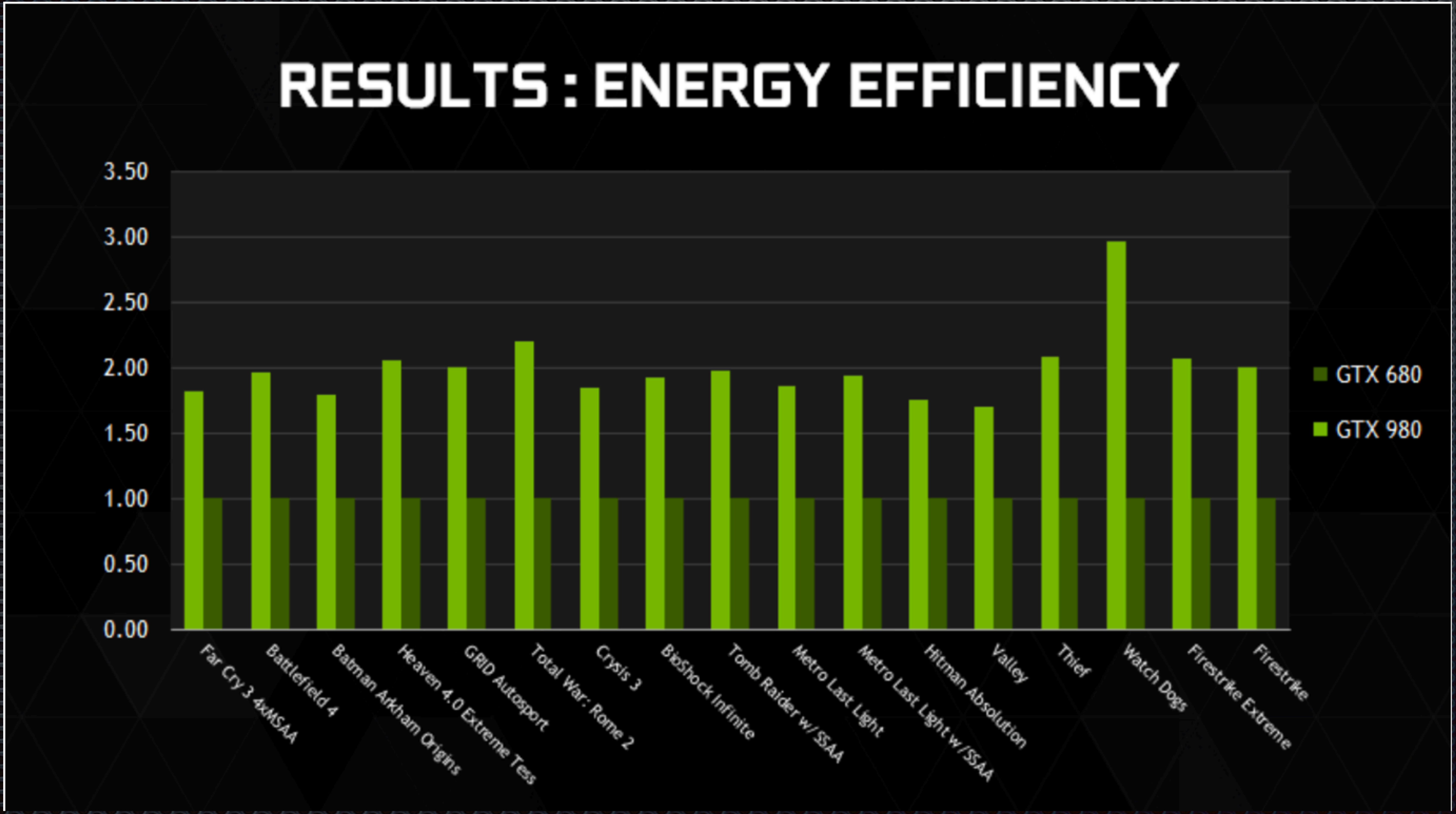
NVIDIA GeForce GTX980

White Paper 2016

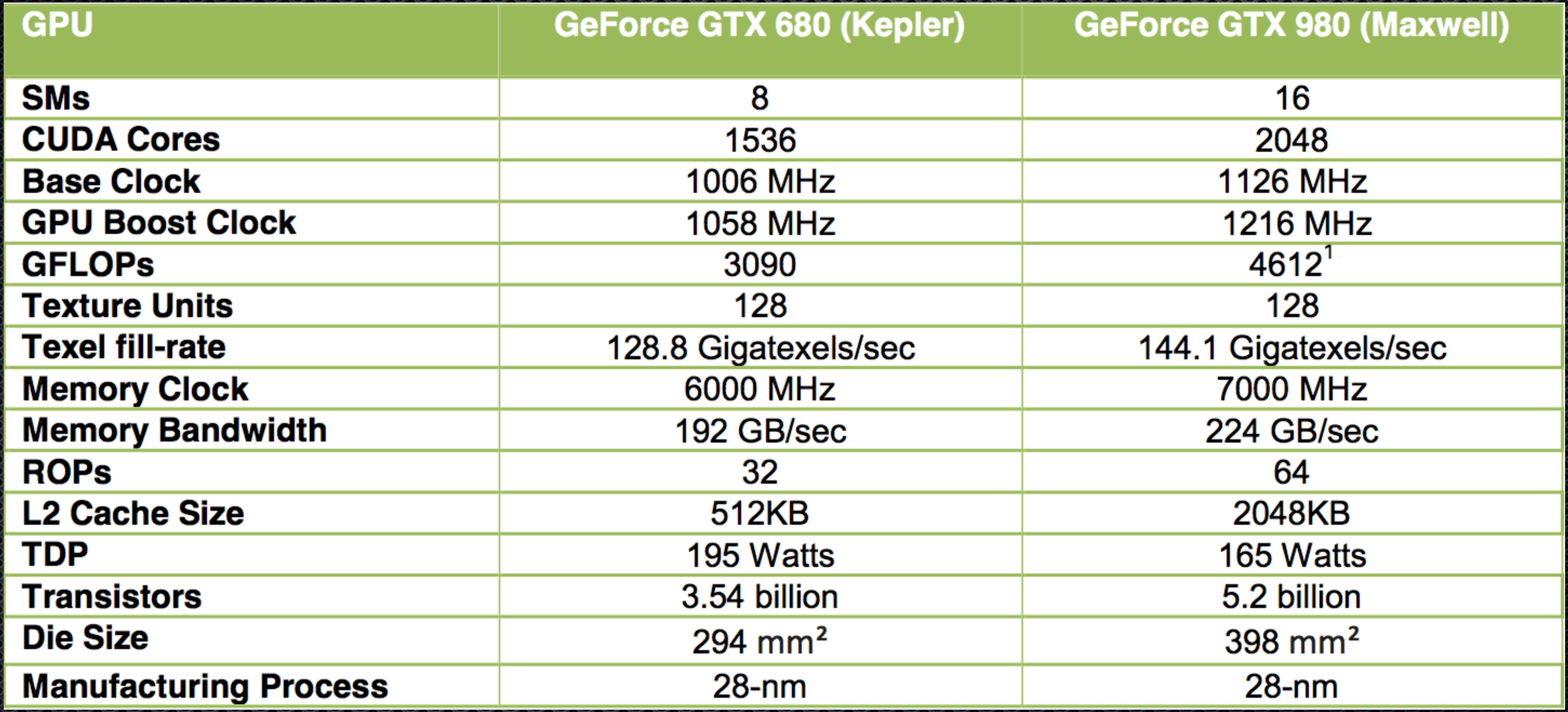
Reduced Power Goal

- SMM Similar to Kepler SMX, but more shared memory (64KB vs 48KB), larger L2 cache (2MB vs 256KB - shifts memory model again), more active blocks (32 vs 16)
- Finer grained power-down control for idle thread engines
- Eliminates shared units that bottlenecked scheduler
- 25% greater die area per SM, 35% performance increase, slightly slower clock (4%), 2X power efficiency

Power Benchmarks



Compared to GK104

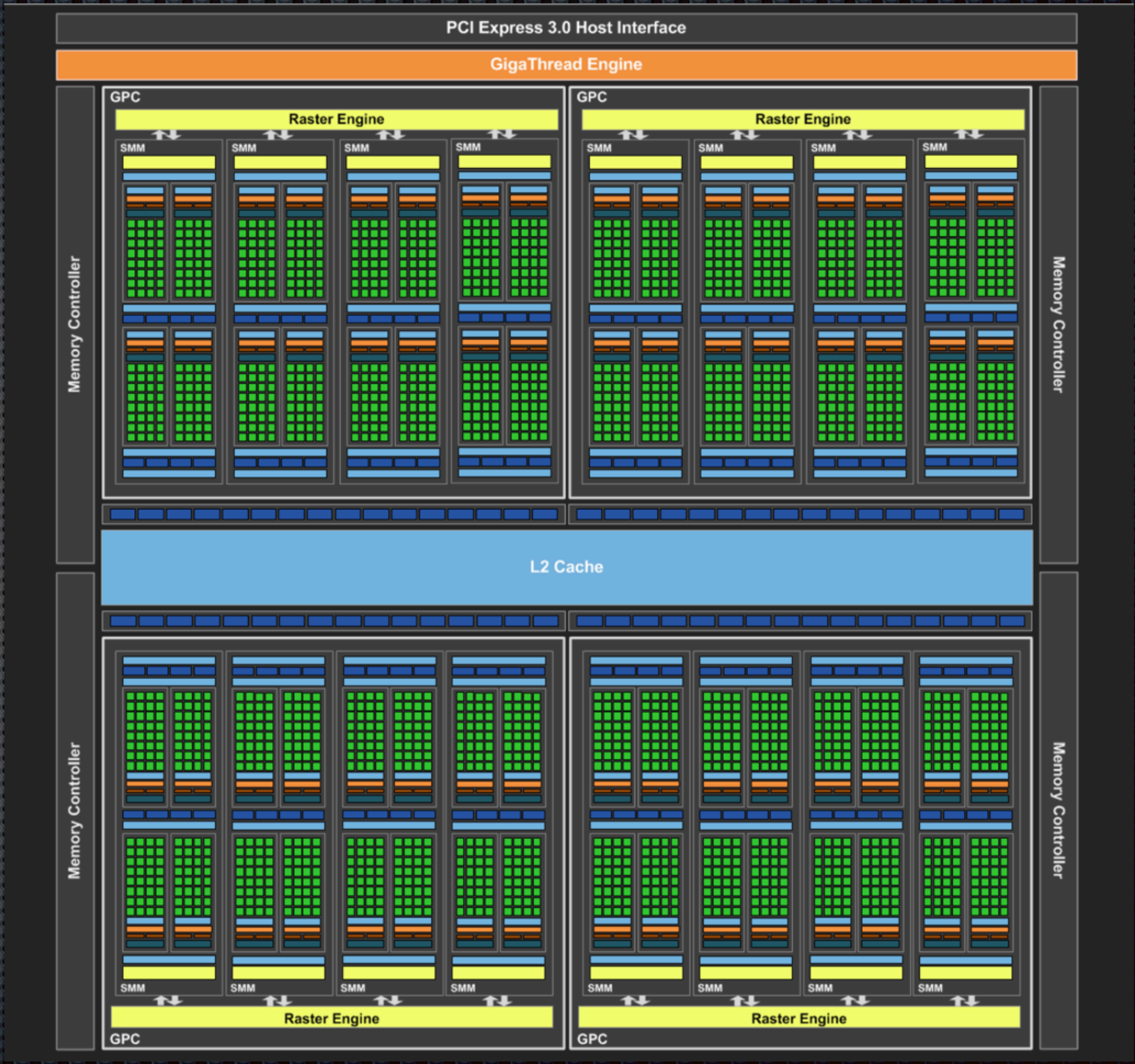


GPU	GeForce GTX 680 (Kepler)	GeForce GTX 980 (Maxwell)
SMs	8	16
CUDA Cores	1536	2048
Base Clock	1006 MHz	1126 MHz
GPU Boost Clock	1058 MHz	1216 MHz
GFLOPs	3090	4612[1]
Texture Units	128	128
Texel fill-rate	128.8 Gigatexels/sec	144.1 Gigatexels/sec
Memory Clock	6000 MHz	7000 MHz
Memory Bandwidth	192 GB/sec	224 GB/sec
ROPs	32	64
L2 Cache Size	512KB	2048KB
TDP	195 Watts	165 Watts
Transistors	3.54 billion	5.2 billion
Die Size	294 mm^2	398 mm^2
Manufacturing Process	28-nm	28-nm

Note: Previous paper was for GK110 Tesla

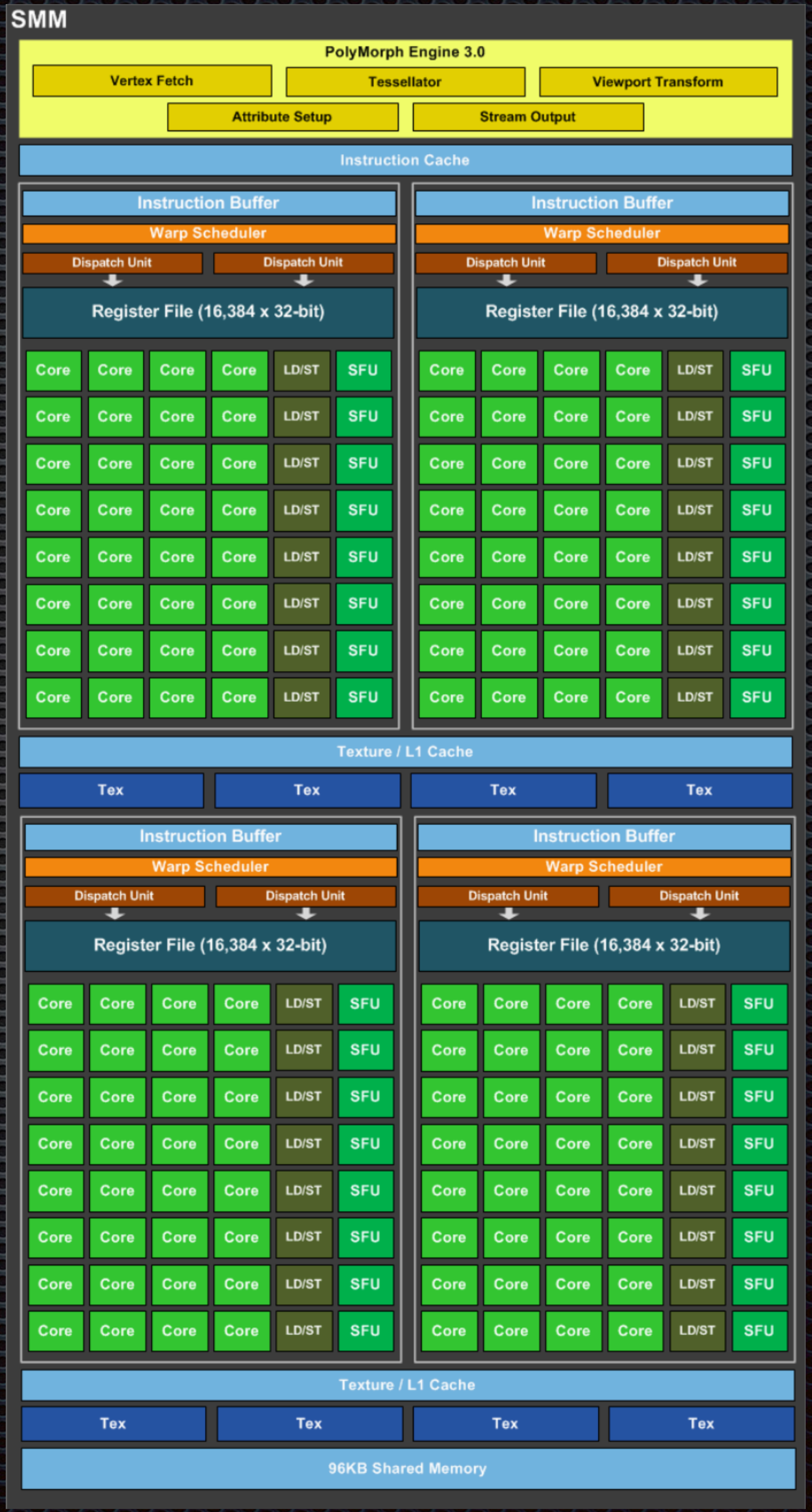
Chip Architecture

4 GPCs
Each w/ 8 SMMs = 32
Each w/128 cores = 2048 total
2MB shared L2



SMM

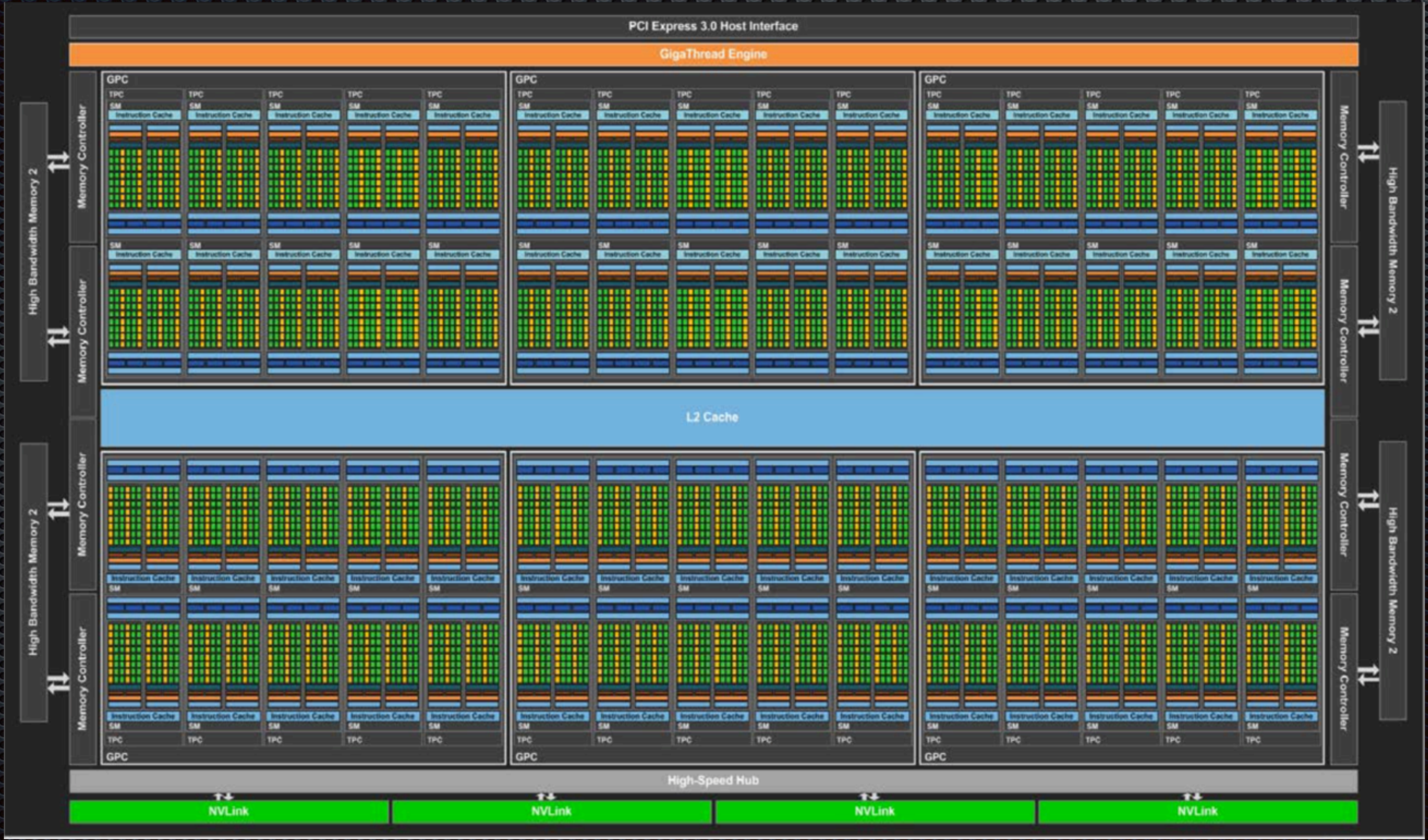
- 64K registers (2M on chip)
  - 256KB
- 96KB Shared memory
- 128 cores
- 4 dual issue schedulers
- 32 load/store units
- iCache



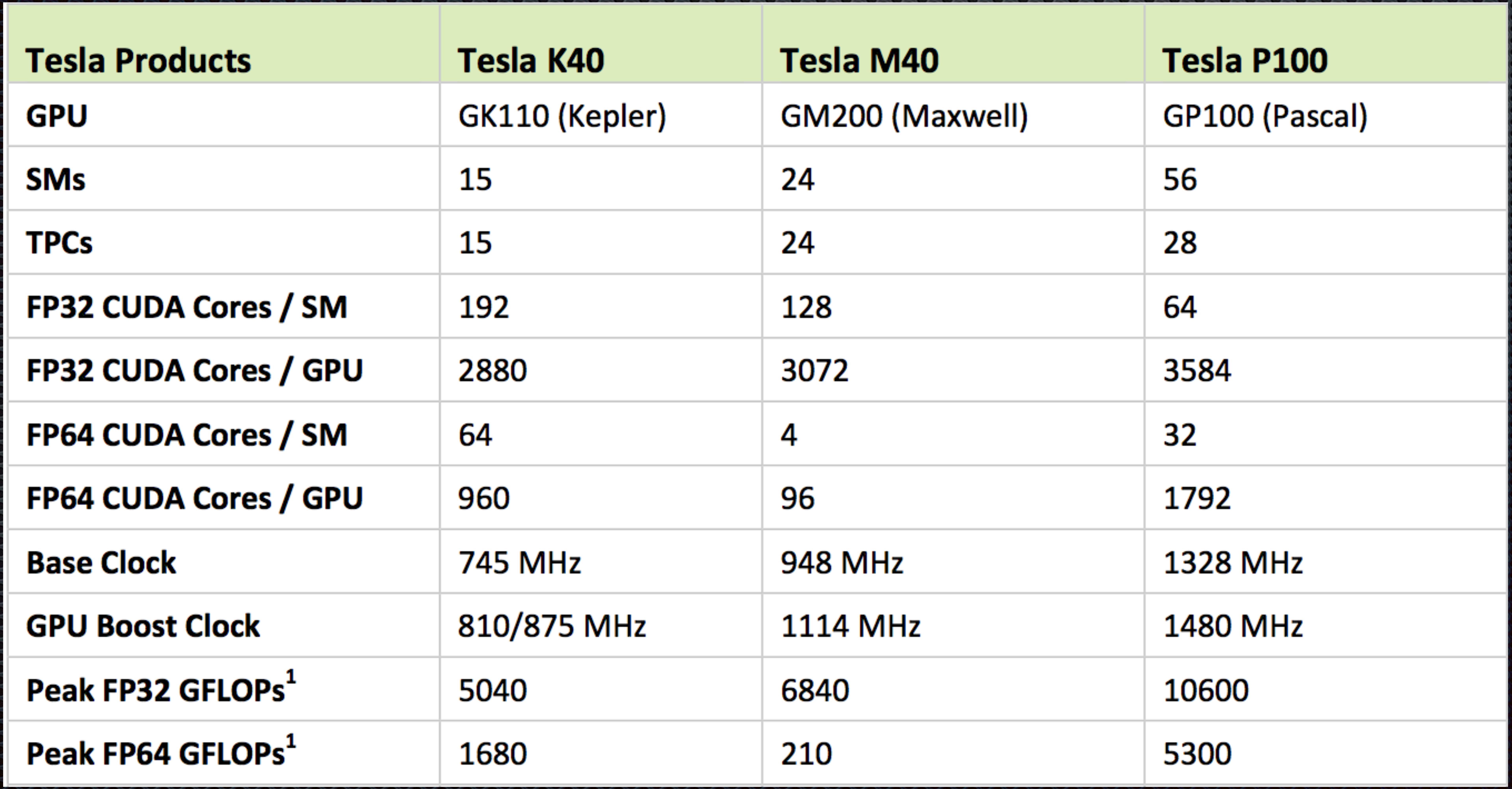
Pascal

- Maxwell-like ISA + 16-bit FP (for deep learning)
- 6 GPCs, each with 10 SMs, Each with 64 cores
  - = 3840 total
- 4GB L2 Cache
- 10.6 TFLOPS (32-bit) 21.6 TFLOPS (16 bit)
- NVLink 160 GB/s bidirectional network interface
- Die-stack 3D memory (16GB)

Chip Architecture

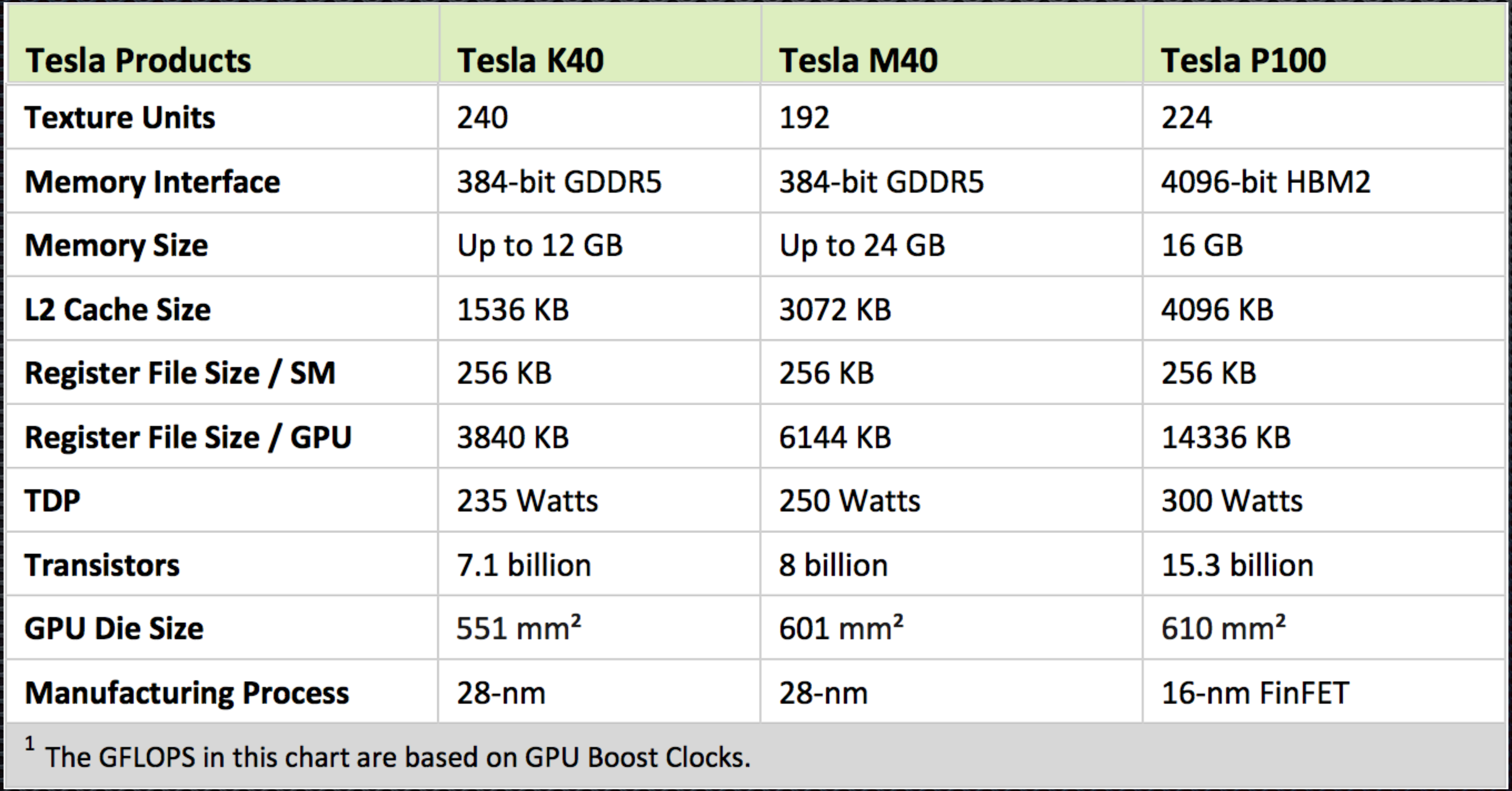


Comparison



Tesla Products	Tesla K40	Tesla M40	Tesla P100
GPU	GK110 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)
SMs	15	24	56
TPCs	15	24	28
FP32 CUDA Cores / SM	192	128	64
FP32 CUDA Cores / GPU	2880	3072	3584
FP64 CUDA Cores / SM	64	4	32
FP64 CUDA Cores / GPU	960	96	1792
Base Clock	745 MHz	948 MHz	1328 MHz
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz
Peak FP32 GFLOPs[1]	5040	6840	10600
Peak FP64 GFLOPs[1]	1680	210	5300

Comparison (2)



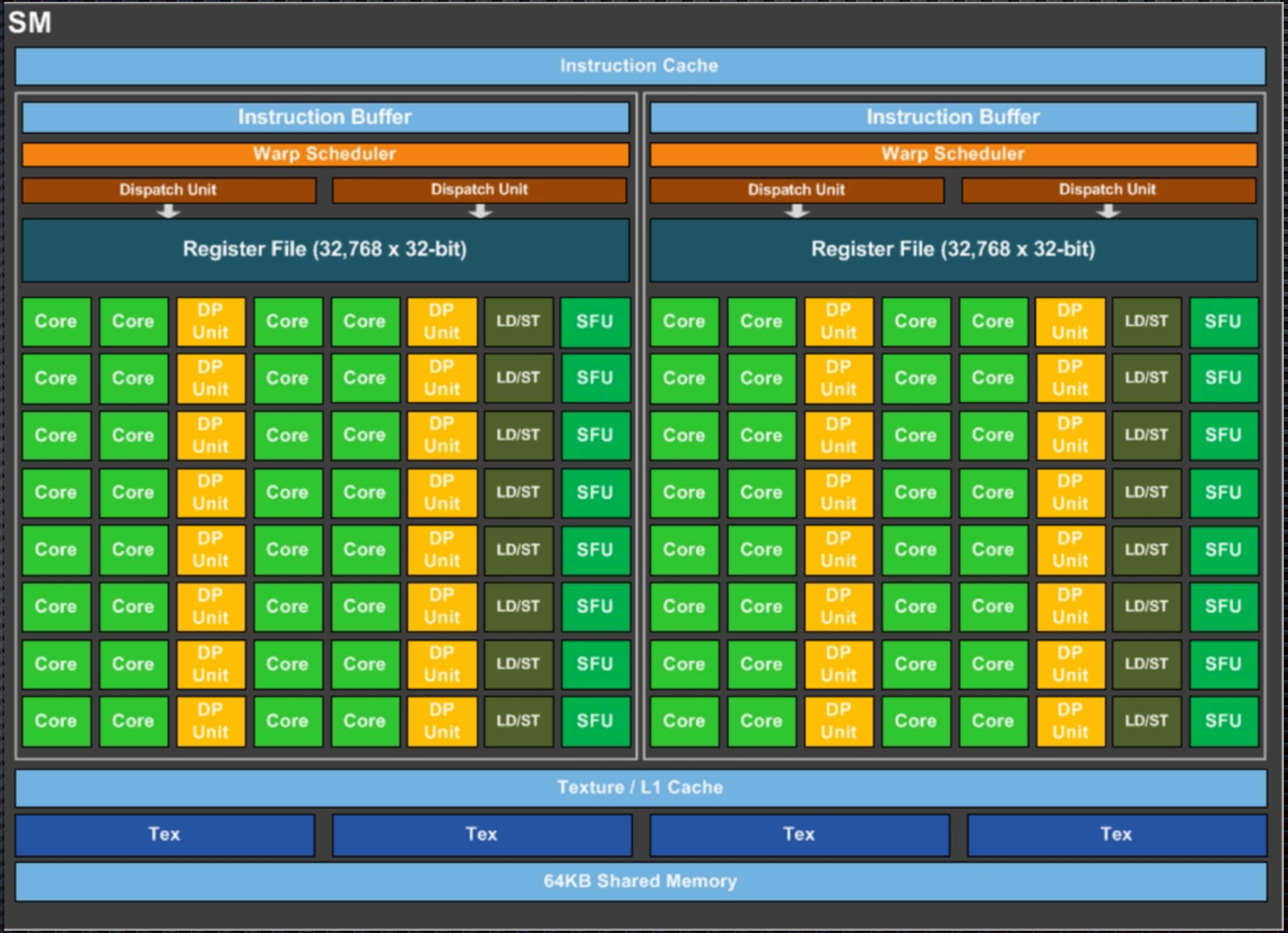
Tesla Products	Tesla K40	Tesla M40	Tesla P100
Texture Units	240	192	224
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB
Register File Size / SM	256 KB	256 KB	256 KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB
TDP	235 Watts	250 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion
GPU Die Size	551 mm^2	601 mm^2	610 mm^2
Manufacturing Process	28-nm	28-nm	16-nm FinFET

[1] The GFLOPS in this chart are based on GPU Boost Clocks.

Notes

- Half the cores/SM but same number of registers
    - Registers are often the limiting factor in thread count
- More SMs and GPC, so more cores, more registers, more warp schedulers overall
    - Less sharing, but more local throughput, more parallelism

SM Architecture

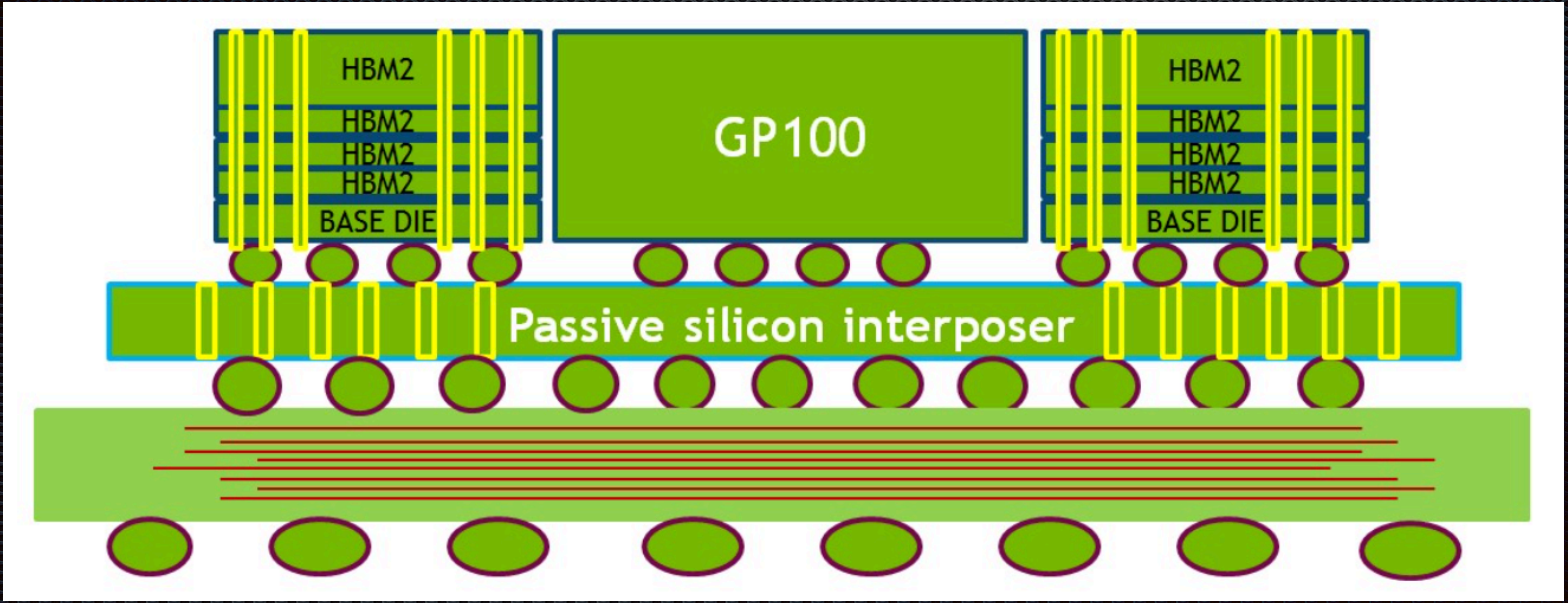


Compute Comparison

GPU	Kepler GK110	Maxwell GM200	Pascal GP100
Compute Capability	3.5	5.2	6.0
Threads / Warp	32	32	32
Max Warps / Multiprocessor	64	64	64
Max Threads / Multiprocessor	2048	2048	2048
Max Thread Blocks / Multiprocessor	16	32	32
Max 32-bit Registers / SM	65536	65536	65536
Max Registers / Block	65536	32768	65536
Max Registers / Thread	255	255	255
Max Thread Block Size	1024	1024	1024
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB

Note: Pascal has fewer cores/SM

Memory Stack



Memory Stack Photo

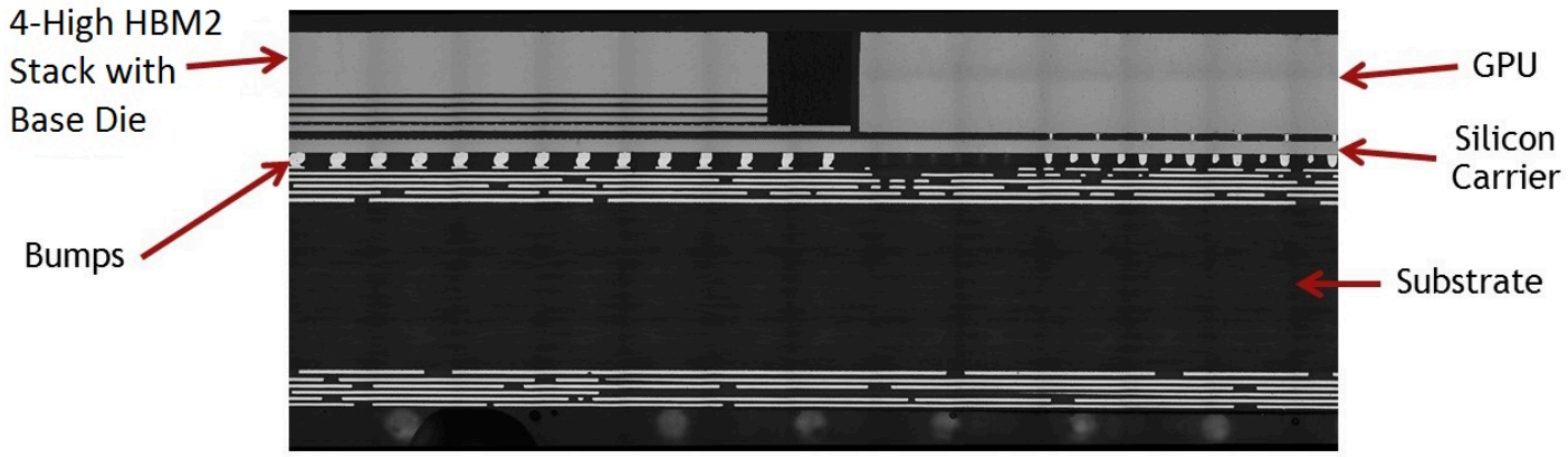
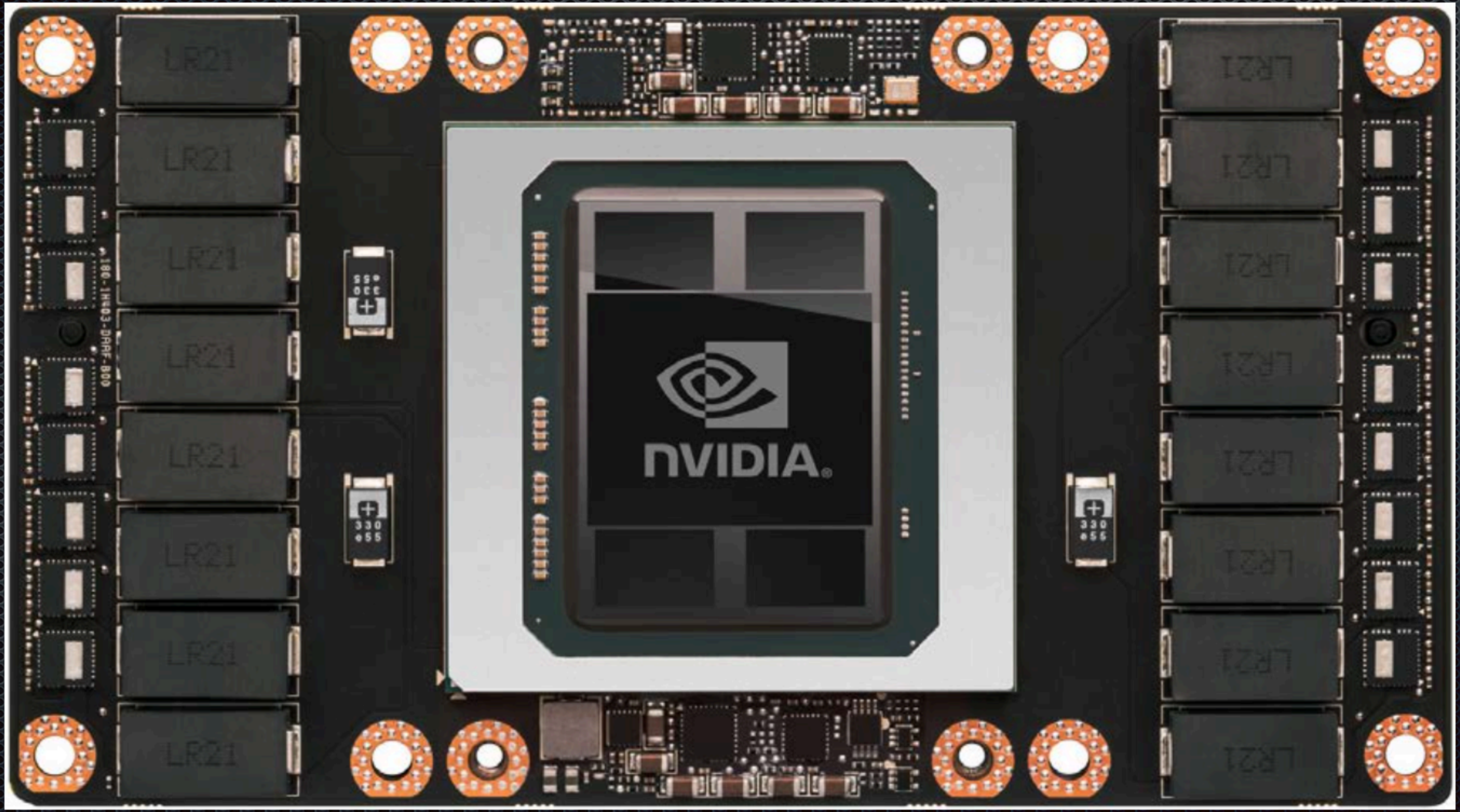


Figure 10. Cross-section Photomicrograph of a P100 HBM2 stack and GP100 GPU

Top View



NVLink

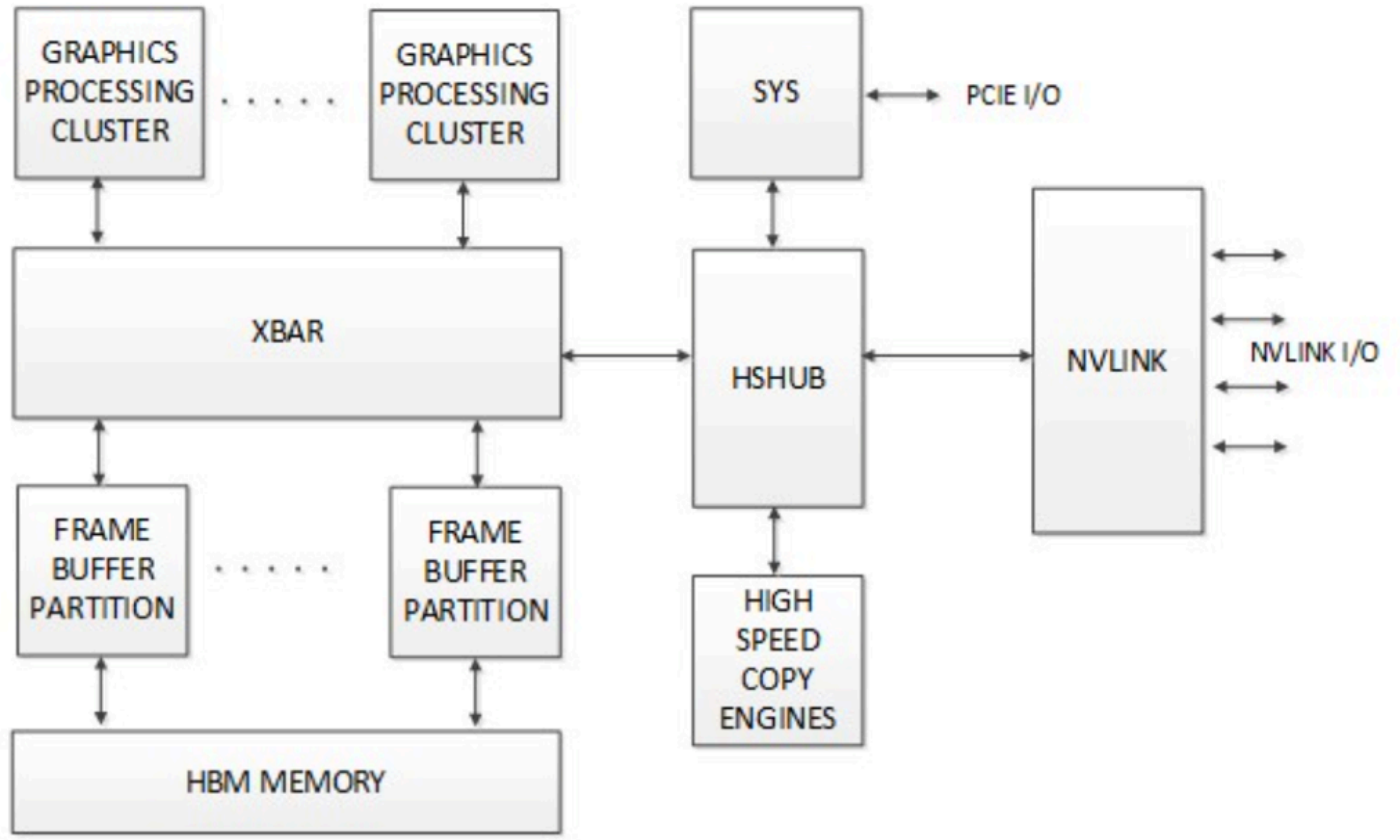


Figure 18. NVLink relationship to other major blocks in GP100

Unified Memory

- Previously, a CPU had to set up shared data before kernel launch
- Pascal adds 49-bit virtual addressing to map all of CPU and GPU space
- Also adds automatic bi-directional CPU/GPU page faulting, with thread suspension
- Can now just malloc on both

Compute Preemption

- Previously, a CUDA application would monopolize the GPU (no graphics display)
- Pascal adds the ability to shelve a task and run graphics as needed at the same time
- Can now use the same GPU for display and compute

Deep Learning Box

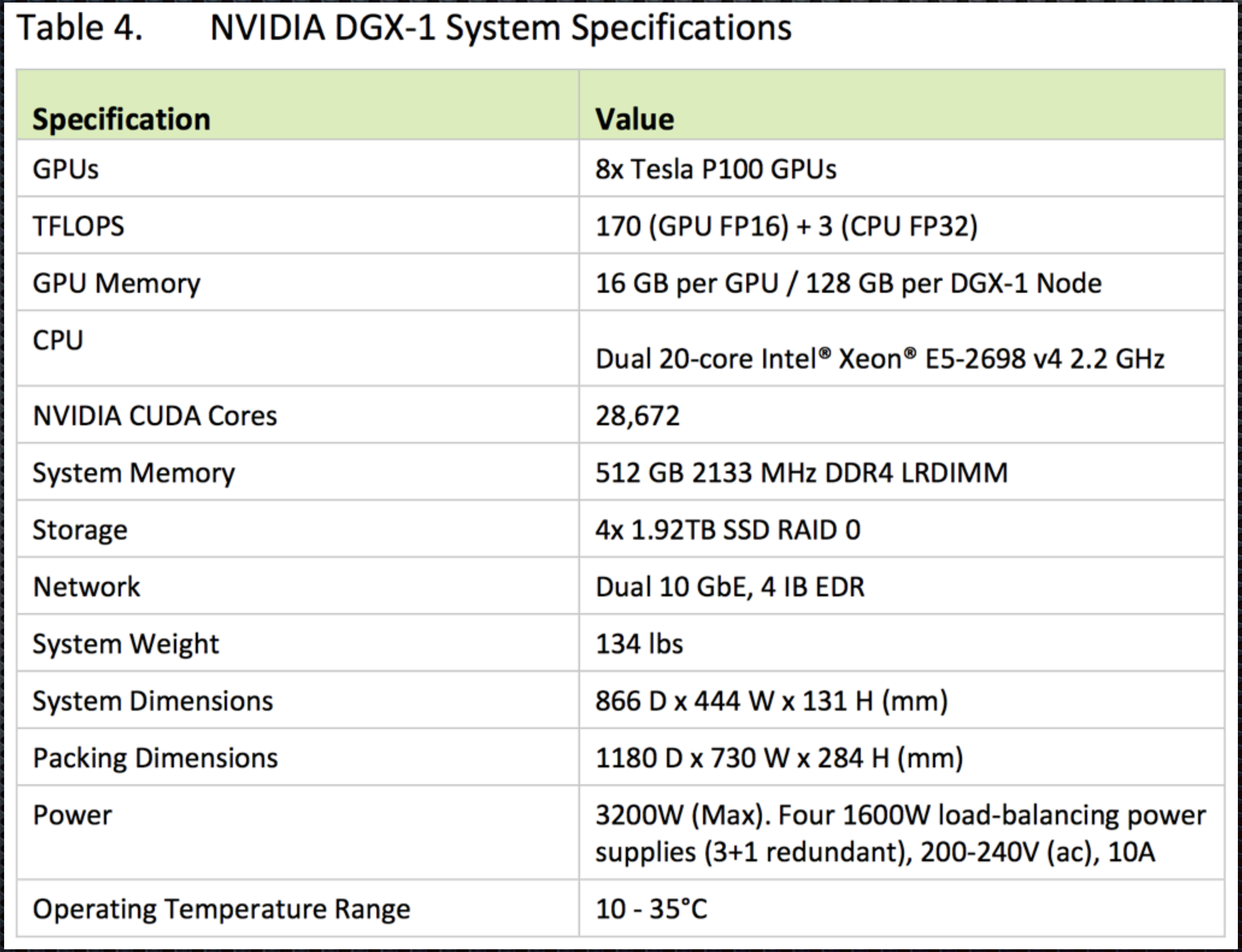


Table 4. NVIDIA DGX-1 System Specifications

Specification	Value
GPUs	8x Tesla P100 GPUs
TFLOPS	170 (GPU FP16) + 3 (CPU FP32)
GPU Memory	16 GB per GPU / 128 GB per DGX-1 Node
CPU	Dual 20-core Intel® Xeon® E5-2698 v4 2.2 GHz
NVIDIA CUDA Cores	28,672
System Memory	512 GB 2133 MHz DDR4 LRDIMM
Storage	4x 1.92TB SSD RAID 0
Network	Dual 10 GbE, 4 IB EDR
System Weight	134 lbs
System Dimensions	866 D x 444 W x 131 H (mm)
Packing Dimensions	1180 D x 730 W x 284 H (mm)
Power	3200W (Max). Four 1600W load-balancing power supplies (3+1 redundant), 200-240V (ac), 10A
Operating Temperature Range	10 - 35°C

Discussion

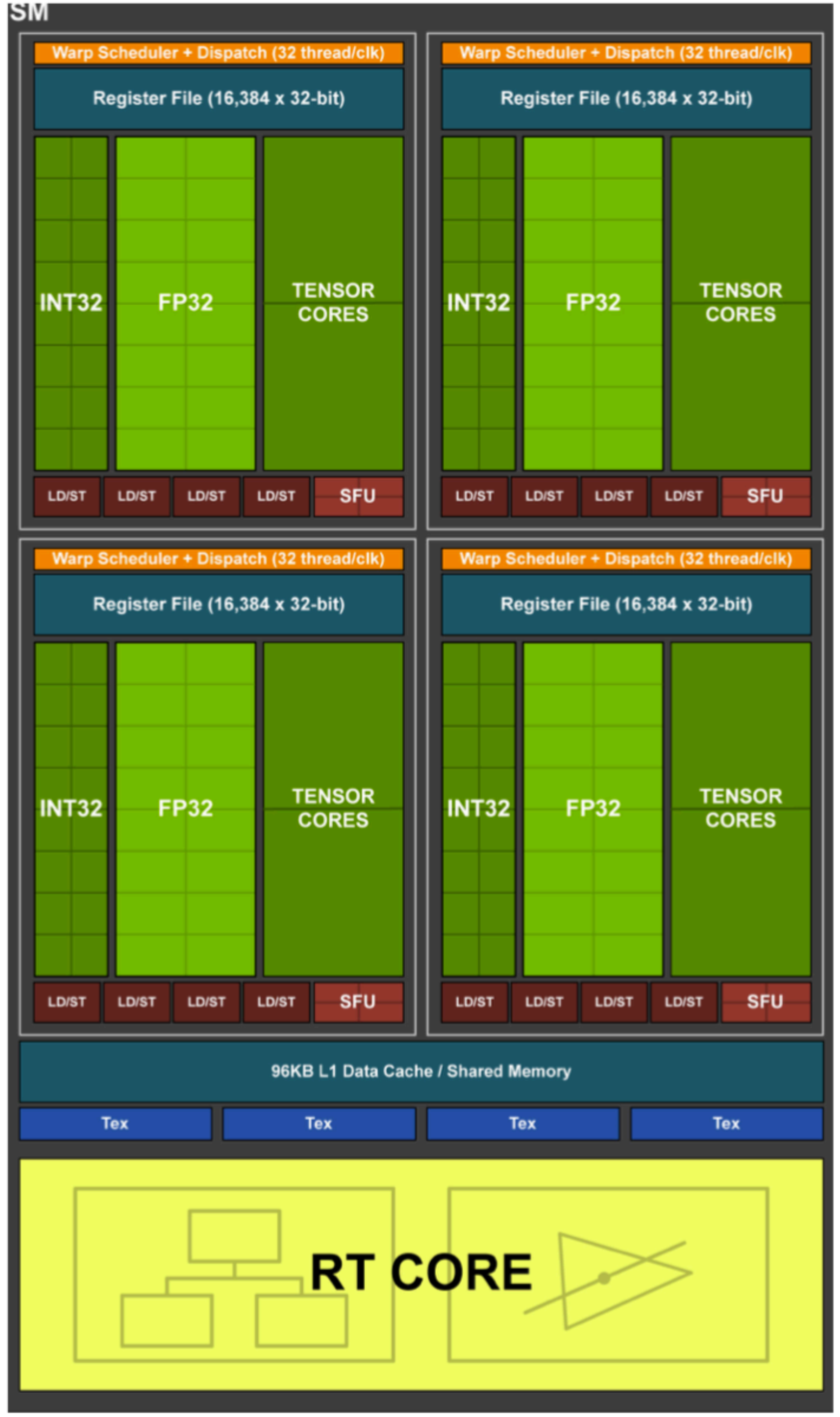
Turing GPU Architecture

NVIDIA 2018

Significant Redesign

- New emphasis on tensor processing, FP16, Int8, Int4 arithmetic
- Unifies multiple memories into a configurable L1 cache
- Allows simultaneous issue of Int and FP ops (like Fermi)
- Changes in graphics generation (ray tracing and shading) using AI
- New memory interface and more NVLink channels
- Reduced support for 64 bit FP

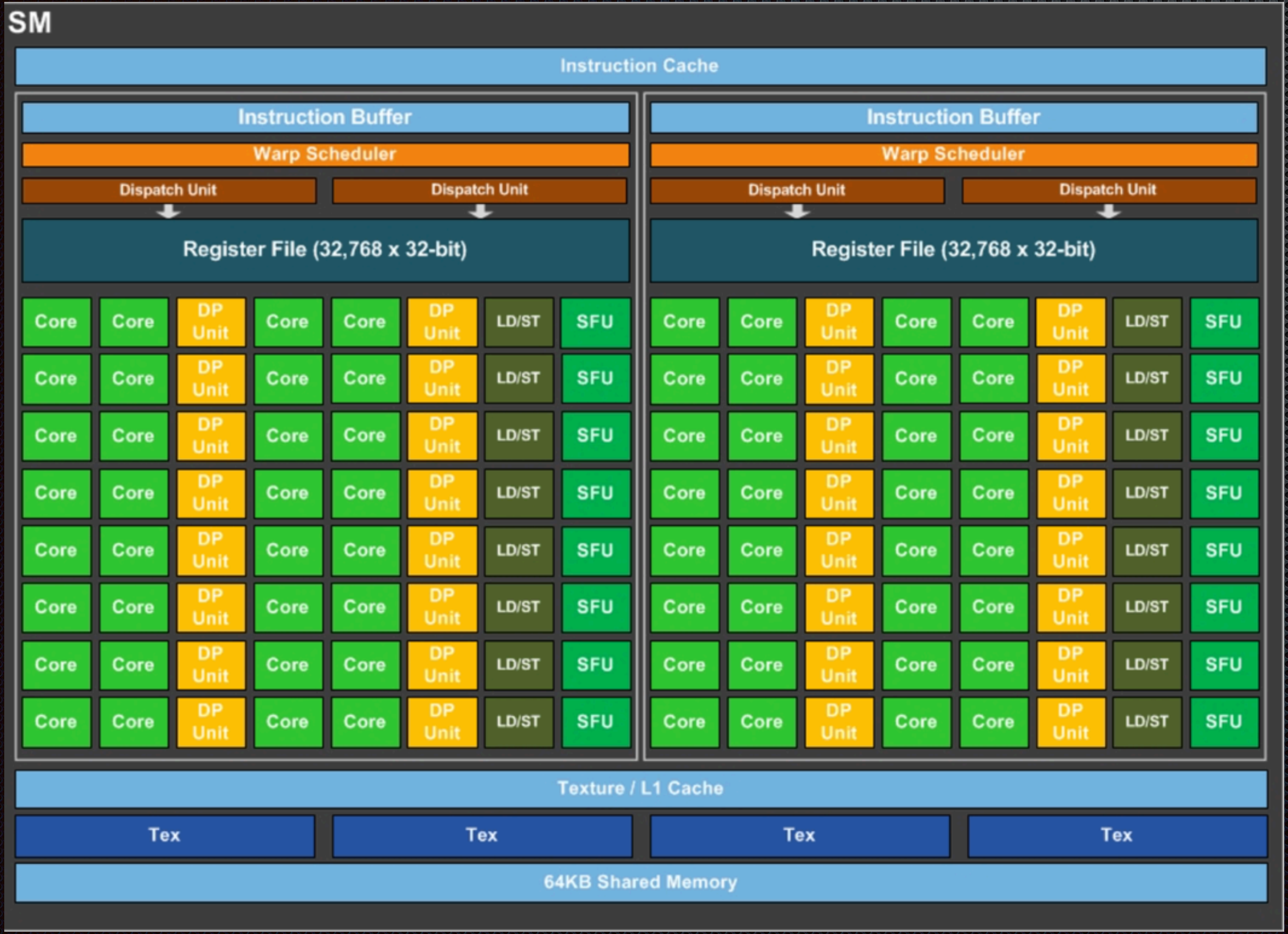
SM



SM
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
SFU
96KB L1 Data Cache / Shared Memory
Tex
RT CORE

Figure 4. Turing TU102/TU104/TU106 Streaming Multiprocessor

SM vs. Pascal



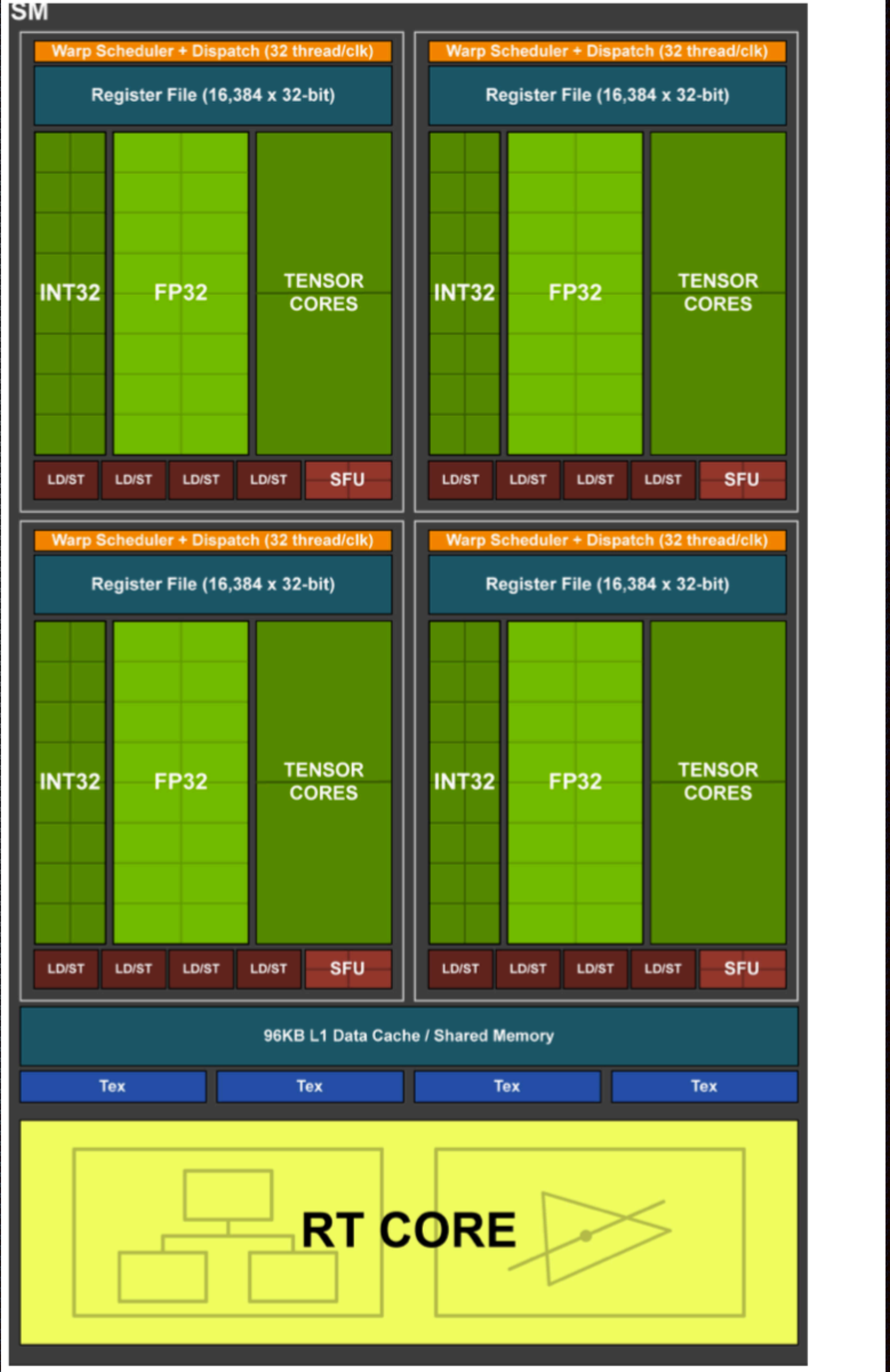

SM
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
SFU
96KB L1 Data Cache / Shared Memory
Tex
RT CORE

Figure 4. Turing TU102/TU104/TU106 Streaming Multiprocessor

System Comparison

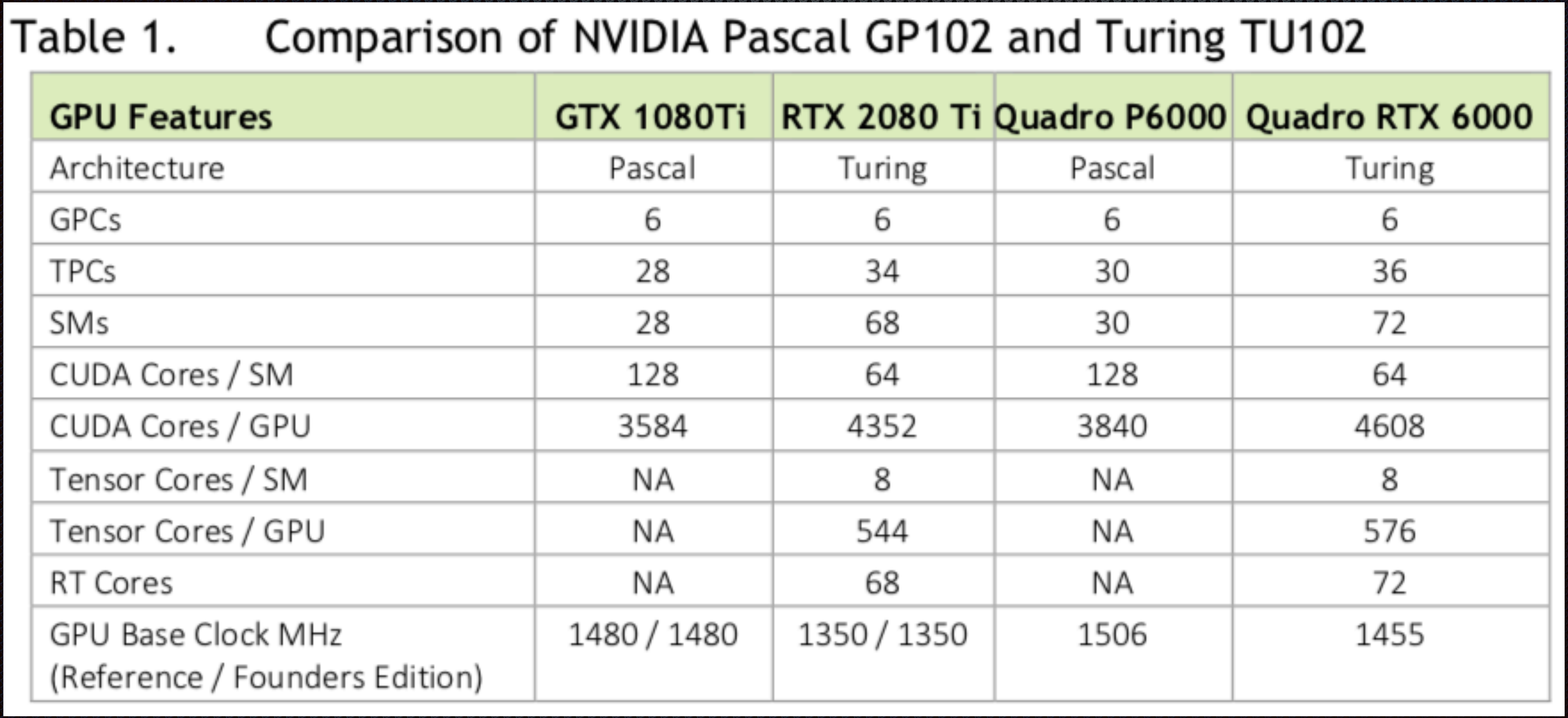


Table 1. Comparison of NVIDIA Pascal GP102 and Turing TU102

GPU Features	GTX 1080Ti	RTX 2080 Ti	Quadro P6000	Quadro RTX 6000
Architecture	Pascal	Turing	Pascal	Turing
GPCs	6	6	6	6
TPCs	28	34	30	36
SMs	28	68	30	72
CUDA Cores / SM	128	64	128	64
CUDA Cores / GPU	3584	4352	3840	4608
Tensor Cores / SM	NA	8	NA	8
Tensor Cores / GPU	NA	544	NA	576
RT Cores	NA	68	NA	72
GPU Base Clock MHz (Reference / Founders Edition)	1480 / 1480	1350 / 1350	1506	1455

Memory Comparison

2x L1 Bandwidth

Lower L1 Hit Latency

Up to 2.7x L1 Capacity

2x L2 Capacity

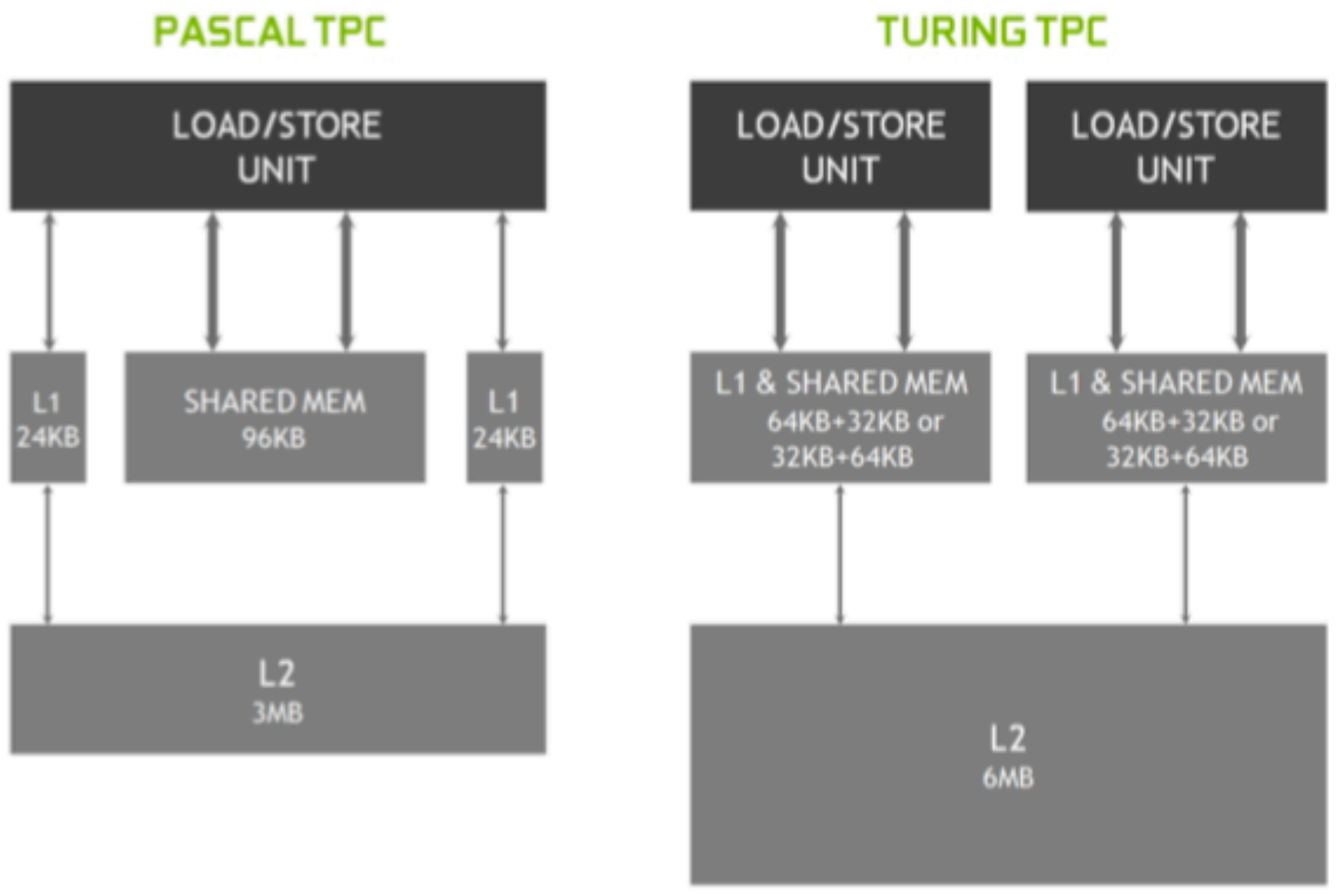


Figure 6. New Shared Memory Architecture

New Memory Interface

- High speed memory uses wave shaping to reduce noise
- Circuits train on actual conditions to adapt wave shaping and filtering
- GDDR6 is a graphics-specific DRAM interface for high throughput of streaming data

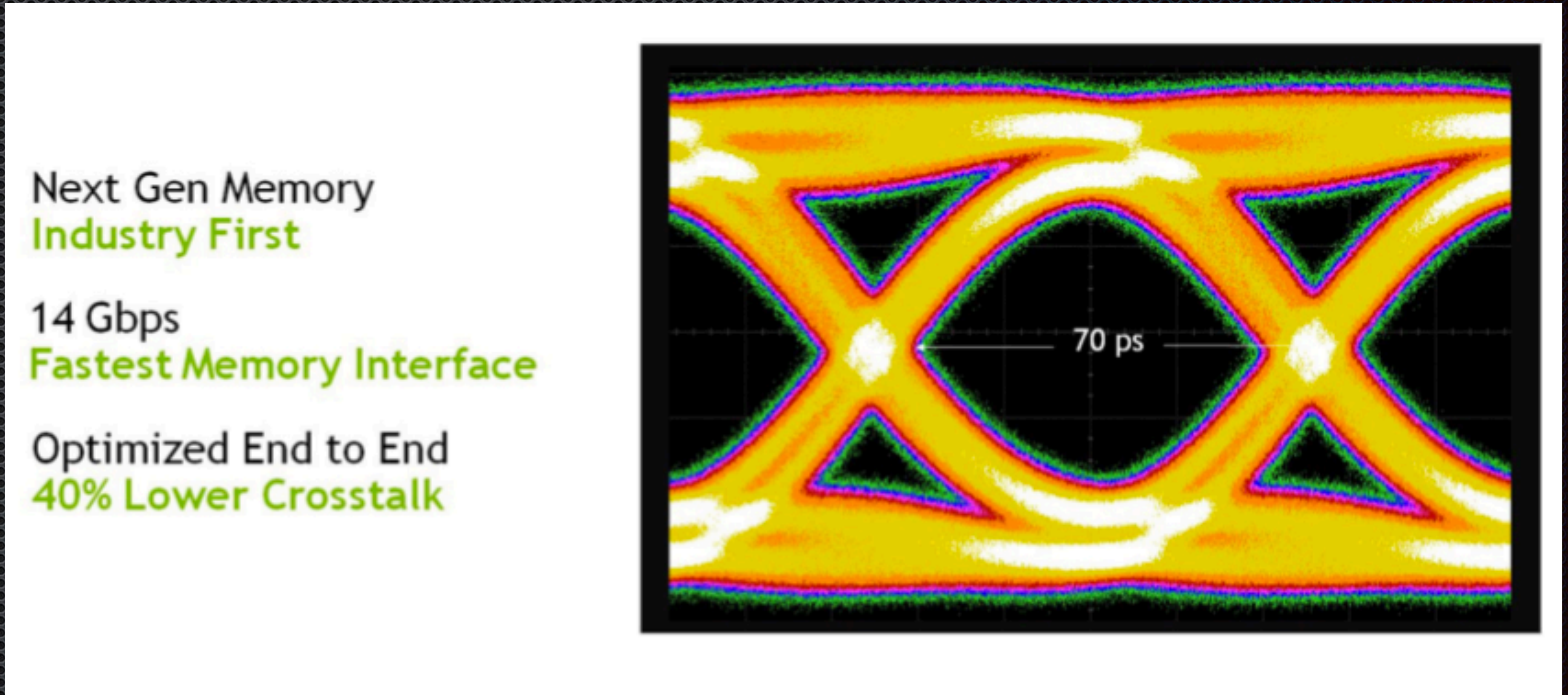
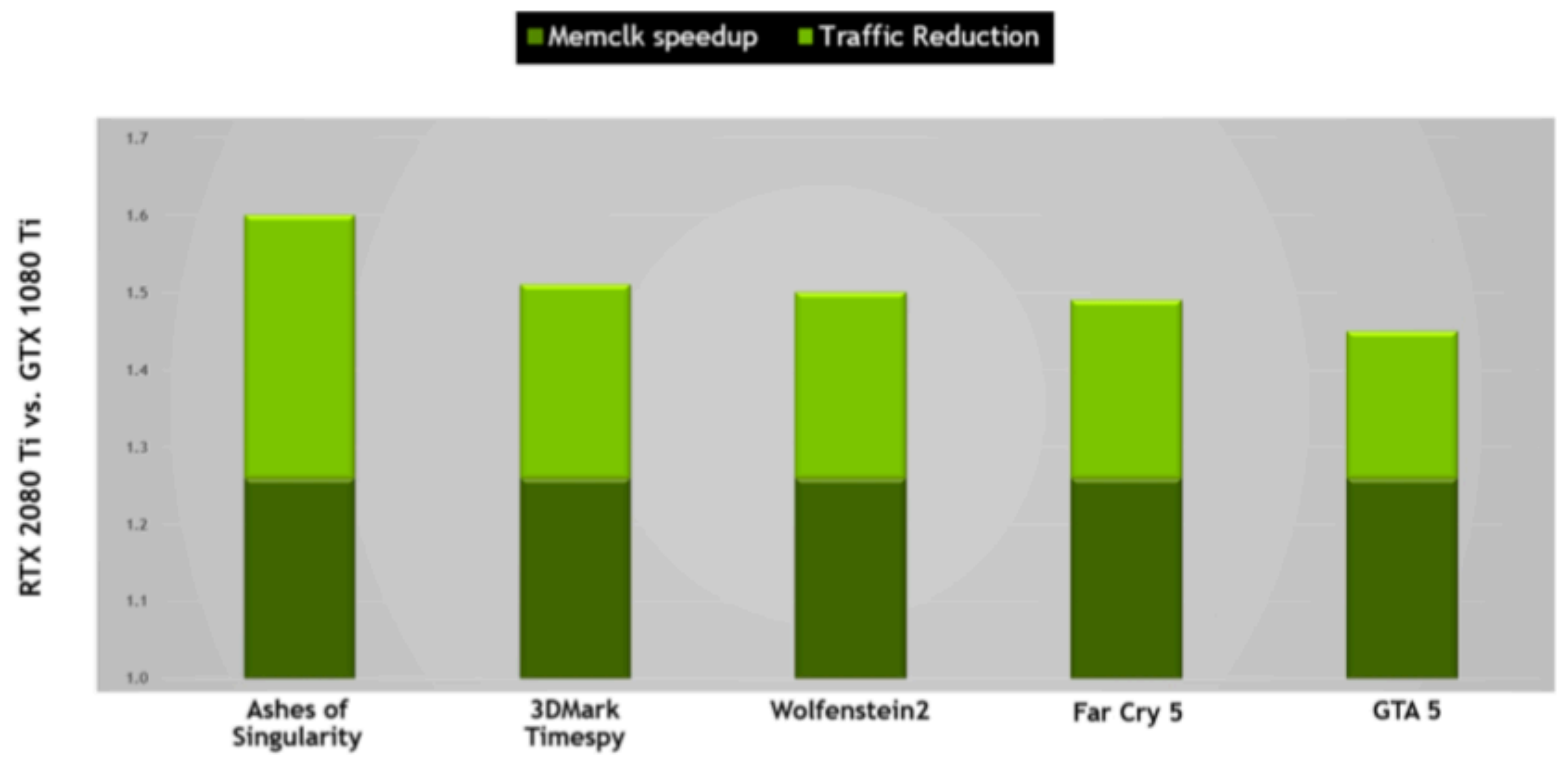


Figure 11. Turing GDDR6

Memory Compression

- Compress data for transmission
- Increases throughput, especially for highly compressible graphics data streams
- Necessary to keep up with processor demand



The memory subsystem and compression (traffic reduction) improvements of Turing TU102-based RTX 2080 Ti deliver approximately 50% effective bandwidth improvements over the Pascal GP102-based 1080 Ti.

Figure 12. 50% Higher Effective Bandwidth

Tensor Support Comparison

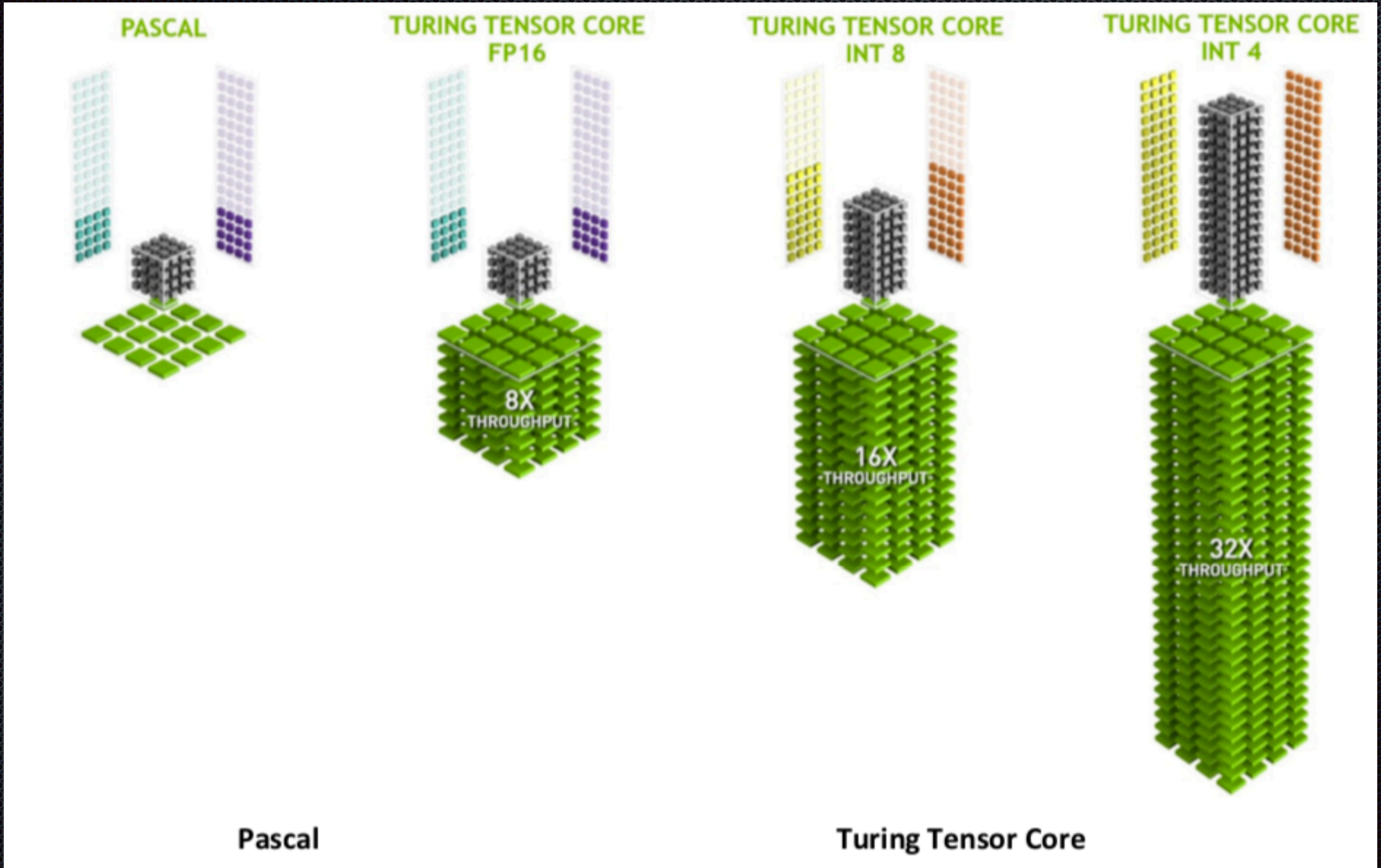
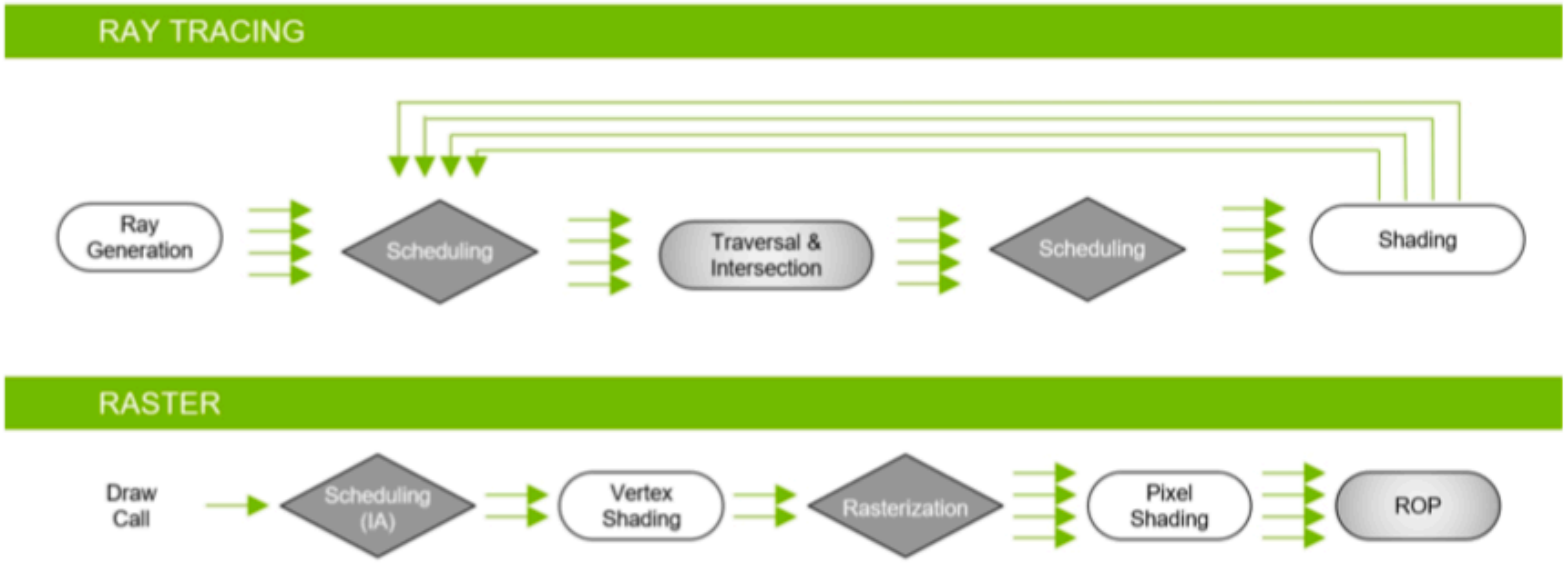


Figure 8. New Turing Tensor Cores Provide Multi-Precision for AI Inference

Parallel Graphics Pipes



Both Ray tracing and Rasterization pipeline operate simultaneously and cooperatively in Hybrid Rendering model used in Turing GPUs.

Figure 17. Details of Ray Tracing and Rasterization Pipeline Stages

Hardware Ray Tracing vs. Software

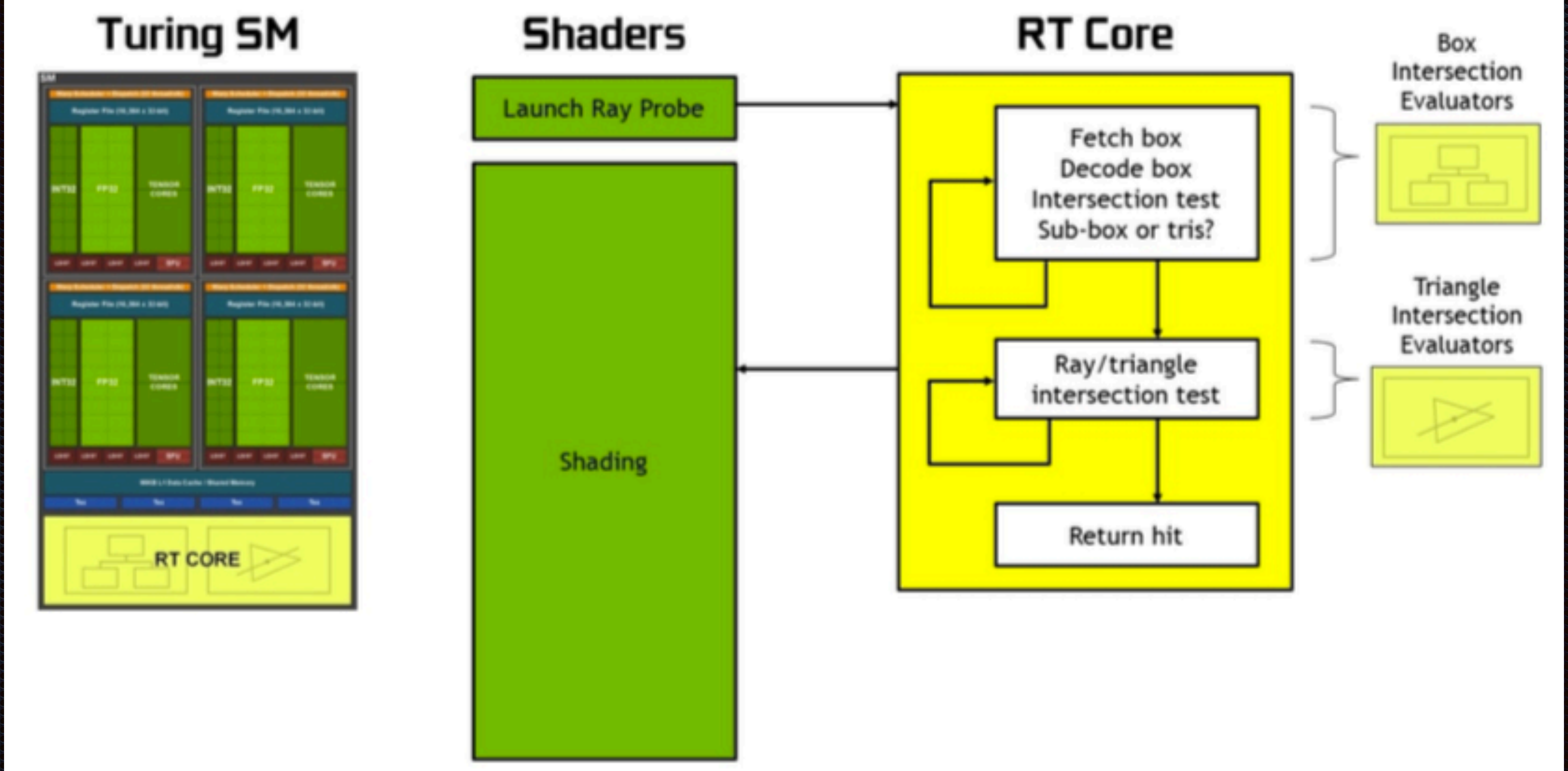
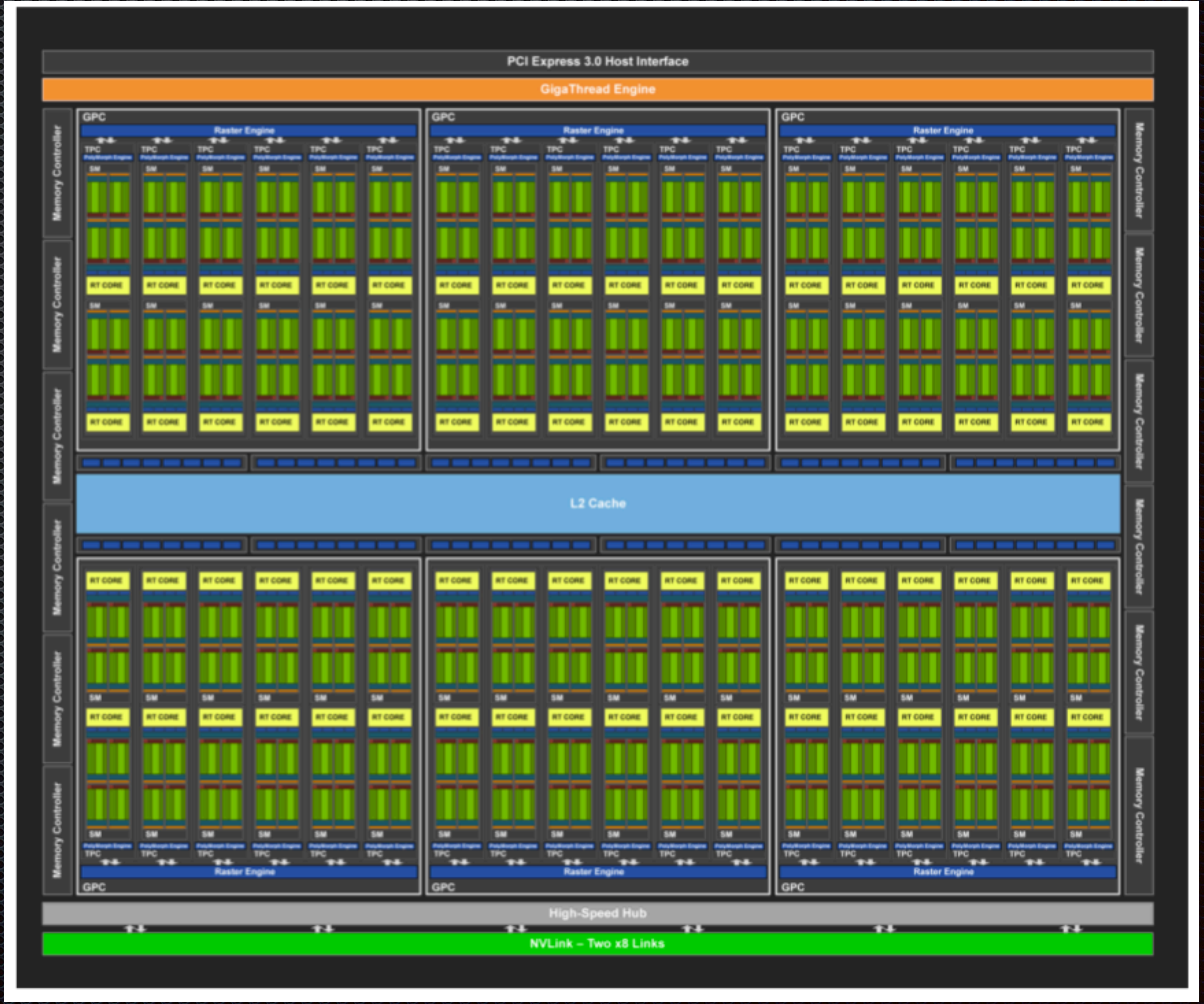


Figure 20. Turing Ray Tracing with RT Cores

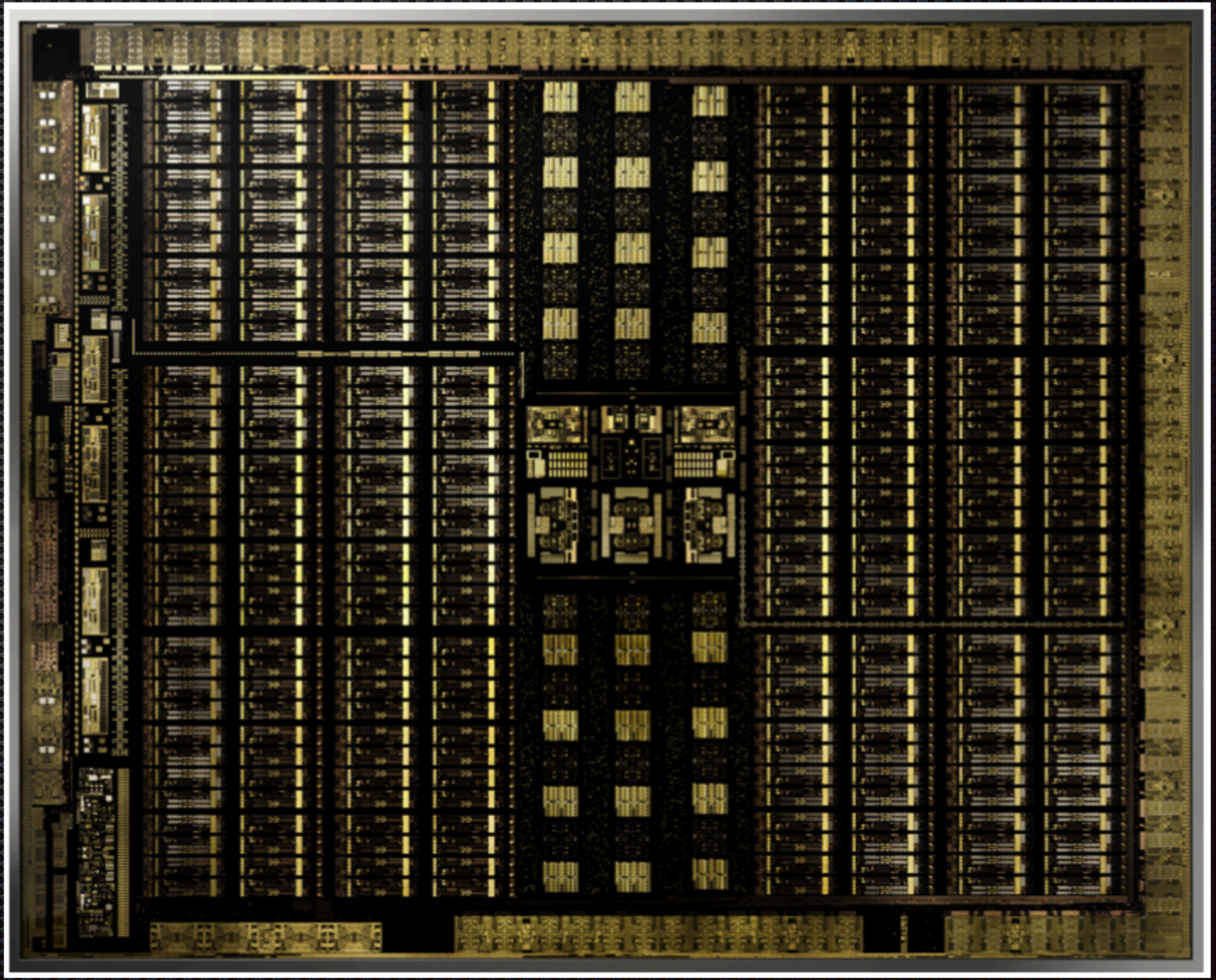
Overall System

- 4608 CUDA cores/72 SMs
- 72 RT cores
- 576 Tensor cores
- 288 Texture units
- 12 GDDR6 memory controllers
- 6.144 MB L2 cache
- 18.432 MB register file

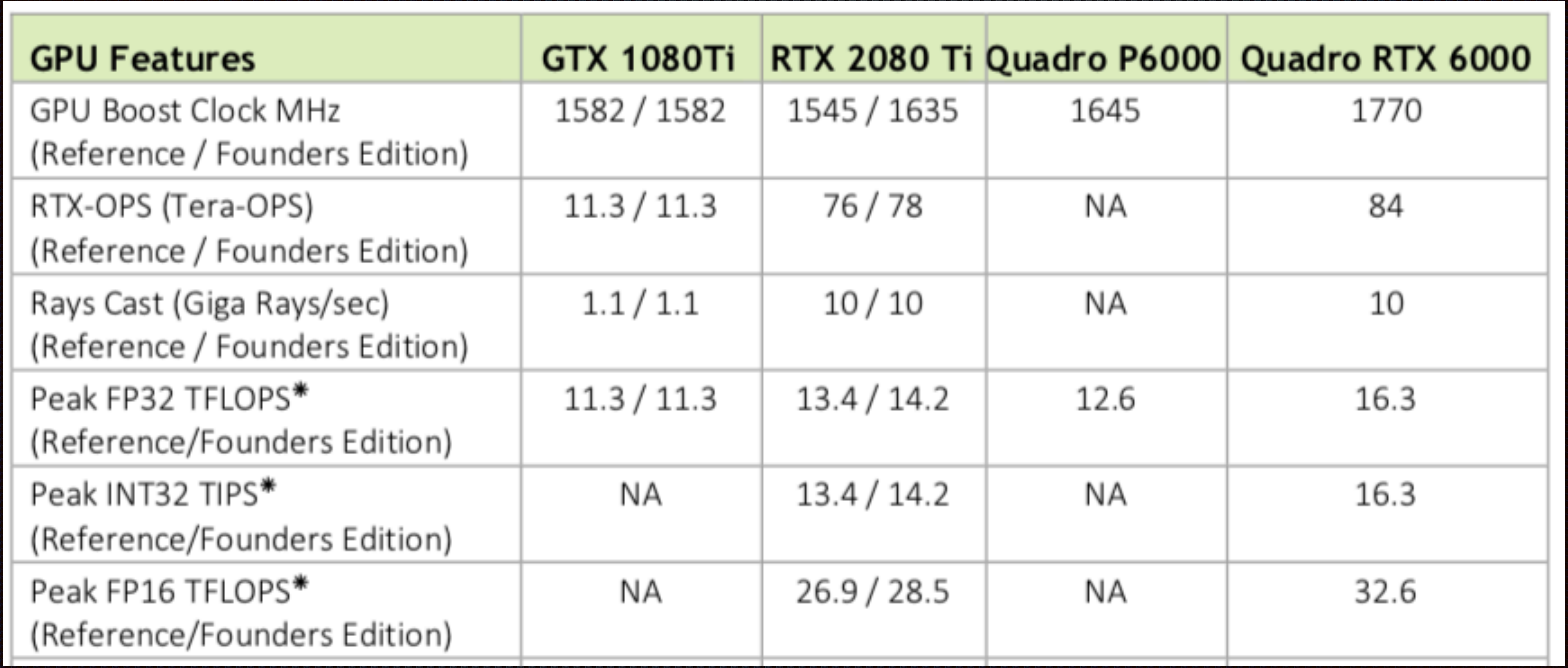


Chip Parameters

- 18.6 B transistors
- 12nm FinFet process
- 260 Watts
- 754 mm^2 (about 30 x 25 mm)

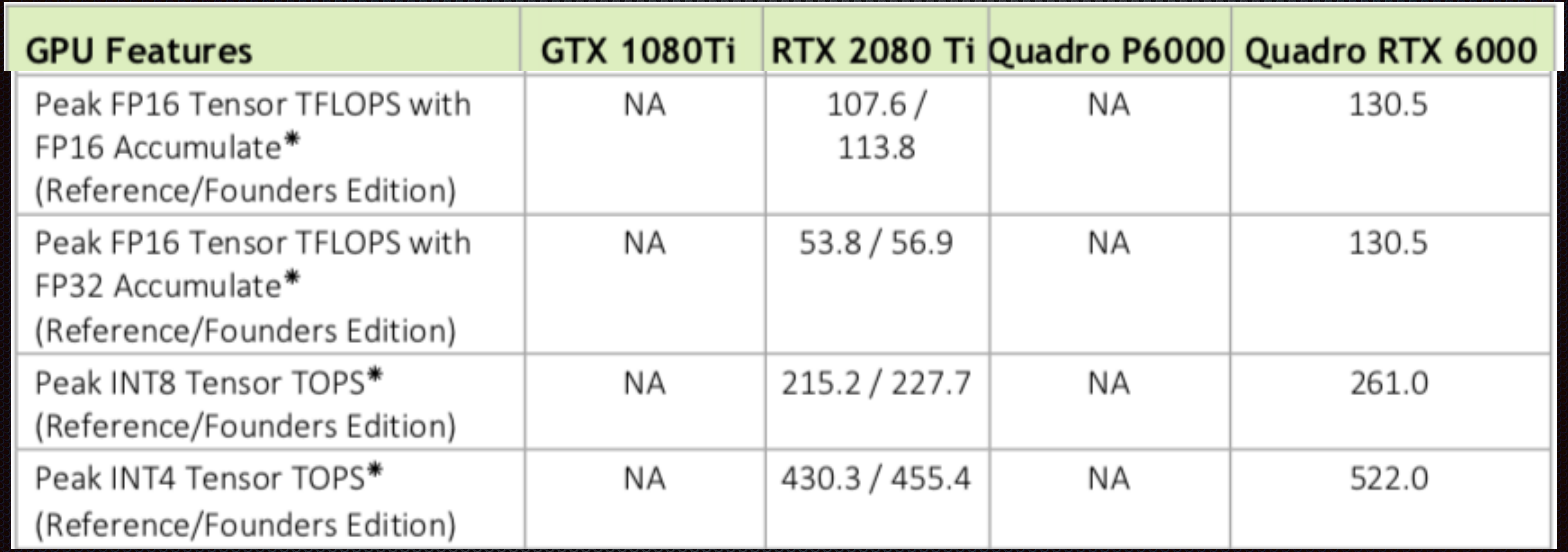


Stated Performance



GPU Features	GTX 1080Ti	RTX 2080 Ti	Quadro P6000	Quadro RTX 6000
GPU Boost Clock MHz (Reference / Founders Edition)	1582 / 1582	1545 / 1635	1645	1770
RTX-OPS (Tera-OPS) (Reference / Founders Edition)	11.3 / 11.3	76 / 78	NA	84
Rays Cast (Giga Rays/sec) (Reference / Founders Edition)	1.1 / 1.1	10 / 10	NA	10
Peak FP32 TFLOPS* (Reference/Founders Edition)	11.3 / 11.3	13.4 / 14.2	12.6	16.3
Peak INT32 TIPS* (Reference/Founders Edition)	NA	13.4 / 14.2	NA	16.3
Peak FP16 TFLOPS* (Reference/Founders Edition)	NA	26.9 / 28.5	NA	32.6

Stated Tensor Performance



GPU Features	GTX 1080Ti	RTX 2080 Ti	Quadro P6000	Quadro RTX 6000
Peak FP16 Tensor TFLOPS with FP16 Accumulate* (Reference/Founders Edition)	NA	107.6 / 113.8	NA	130.5
Peak FP16 Tensor TFLOPS with FP32 Accumulate* (Reference/Founders Edition)	NA	53.8 / 56.9	NA	130.5
Peak INT8 Tensor TOPS* (Reference/Founders Edition)	NA	215.2 / 227.7	NA	261.0
Peak INT4 Tensor TOPS* (Reference/Founders Edition)	NA	430.3 / 455.4	NA	522.0

Discussion