Архитектура вычислительных систем

Лекция 13.

Яревский Е.А.

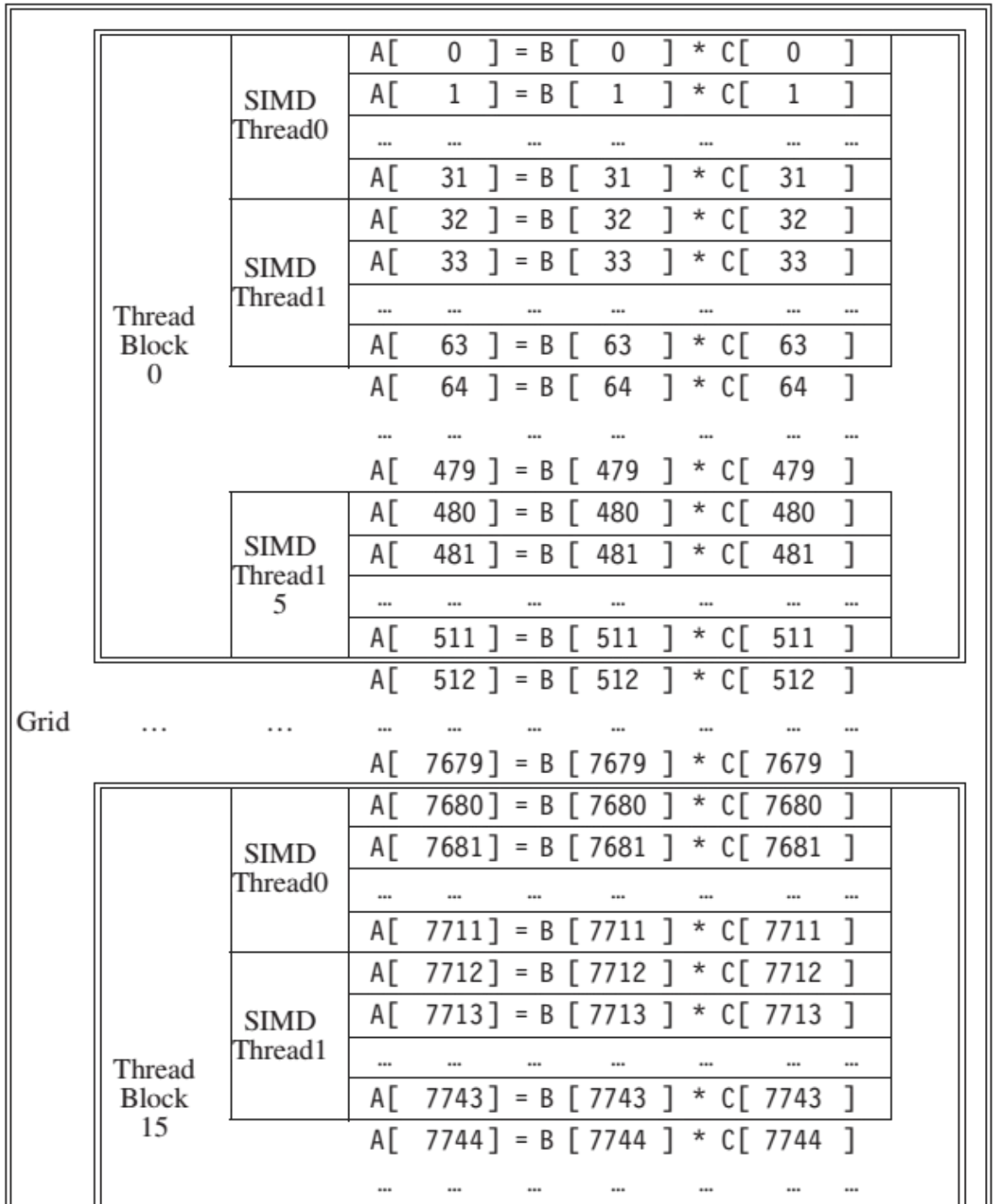
Кафедра вычислительной физики

АРХИТЕКТУРА СОВРЕМЕННЫХ GPU

Processing hardware	Multithreaded SIMD Processor	(Multithreaded) Vector Processor	Streaming Multiprocessor	A multithreaded SIMD Processor executes threads of SIMD instructions, independent of other SIMD Processors.
	Thread Block Scheduler	Scalar Processor	Giga Thread Engine	Assigns multiple Thread Blocks (bodies of vectorized loop) to multithreaded SIMD Processors.
	SIMD Thread Scheduler	Thread scheduler in a Multithreaded CPU	Warp Scheduler	Hardware unit that schedules and issues threads of SIMD instructions when they are ready to execute; includes a scoreboard to track SIMD Thread execution.
	SIMD Lane	Vector Lane	Thread Processor	A SIMD Lane executes the operations in a thread of SIMD instructions on a single element. Results stored depending on mask.
Memory hardware	GPU Memory	Main Memory	Global Memory	DRAM memory accessible by all multithreaded SIMD Processors in a GPU.
	Private Memory	Stack or Thread Local Storage (OS)	Local Memory	Portion of DRAM memory private to each SIMD Lane.
	Local Memory	Local Memory	Shared Memory	Fast local SRAM for one multithreaded SIMD Processor, unavailable to other SIMD Processors.
	SIMD Lane Registers	Vector Lane Registers	Thread Processor Registers	Registers in a single SIMD Lane allocated across a full thread block (body of vectorized loop).

Терминология, используемая в (NVIDIA) GPU

«a GPU is a multiprocessor composed of multithreaded SIMD Processors»


Thread Block 0
SIMD Thread0
SIMD Thread1
SIMD Thread15
A[0] = B [0] * C[0]
A[1] = B [1] * C[1]
A[31] = B [31] * C[31]
A[32] = B [32] * C[32]
A[33] = B [33] * C[33]
A[63] = B [63] * C[63]
A[64] = B [64] * C[64]
A[479] = B [479] * C[479]
A[480] = B [480] * C[480]
A[481] = B [481] * C[481]
A[511] = B [511] * C[511]
A[512] = B [512] * C[512]
Grid
A[7679] = B [7679] * C[7679]
Thread Block 15
A[7680] = B [7680] * C[7680]
A[7681] = B [7681] * C[7681]
A[7711] = B [7711] * C[7711]
A[7712] = B [7712] * C[7712]
A[7713] = B [7713] * C[7713]
A[7743] = B [7743] * C[7743]
A[7744] = B [7744] * C[7744]

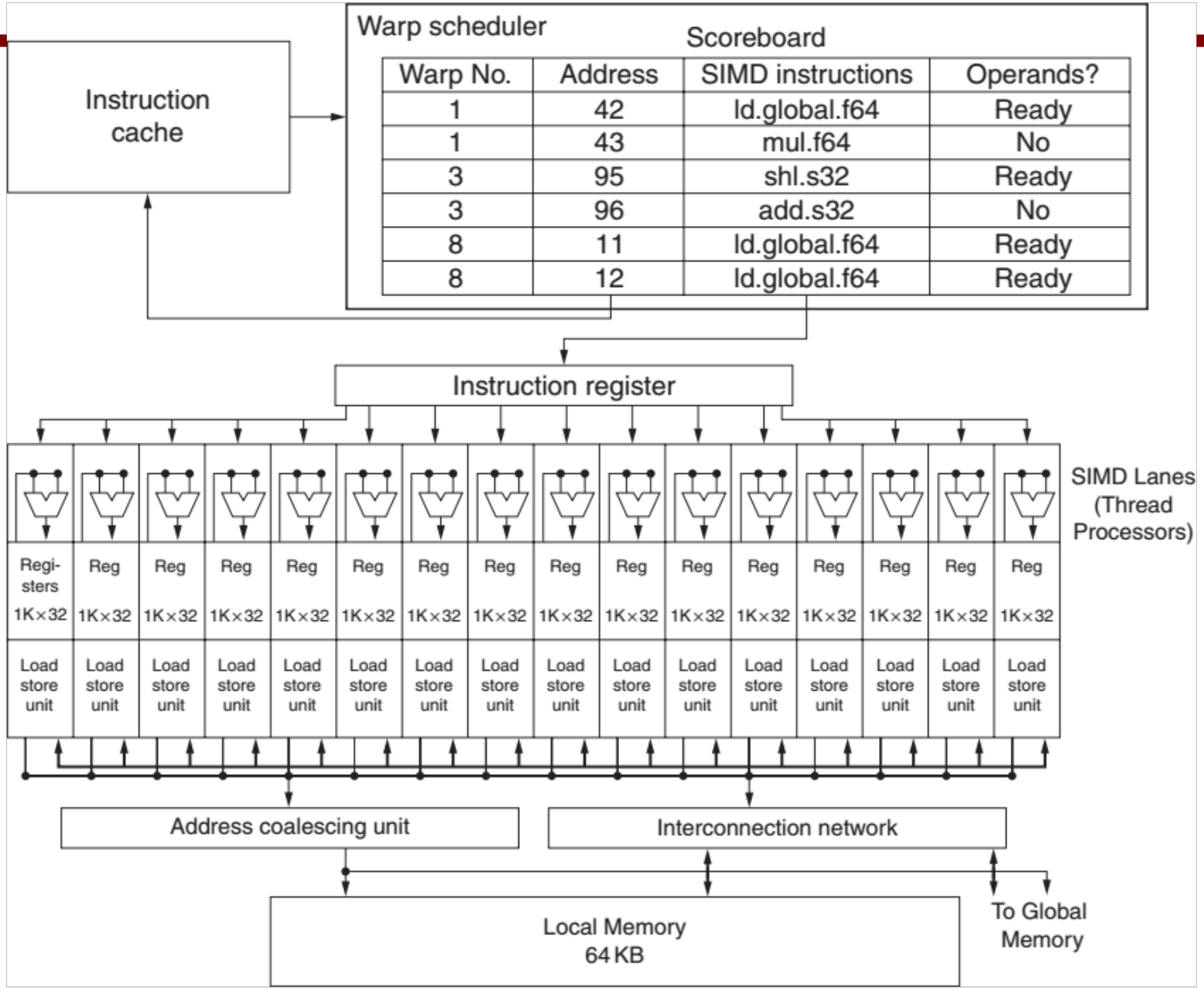


Figure 4.14 Simplified block diagram of a Multithreaded SIMD Processor. It has 16 SIMD lanes. The SIMD Thread Scheduler has, say, 48 independent threads of SIMD instructions that it schedules with a table of 48 PCs.

GPU hardware has two levels of hardware schedulers:
(1) the Thread Block Scheduler that assigns Thread Blocks (bodies of vectorized loops) to multithreaded SIMD Processors, which ensures that thread blocks are assigned to the processors whose local memories have the corresponding data, and
(2) the SIMD Thread Scheduler within a SIMD Processor, which schedules when threads of SIMD instructions should run.

The thread consists of SIMD instructions, so the SIMD Processor must have parallel functional units to perform the operation – **SIMD Lanes**.

The assumption of GPU architects is that GPU applications have so many threads of SIMD instructions that multithreading can both hide the latency to DRAM and increase utilization of multithreaded SIMD Processors.
However, to hedge their bets, the recent GPUs includes an L2 cache.

NVIDA GPU Instruction Set Architecture

PTX (Parallel Thread Execution) provides a stable instruction set for compilers as well as compatibility across generations of GPUs.

The format of a PTX instruction is

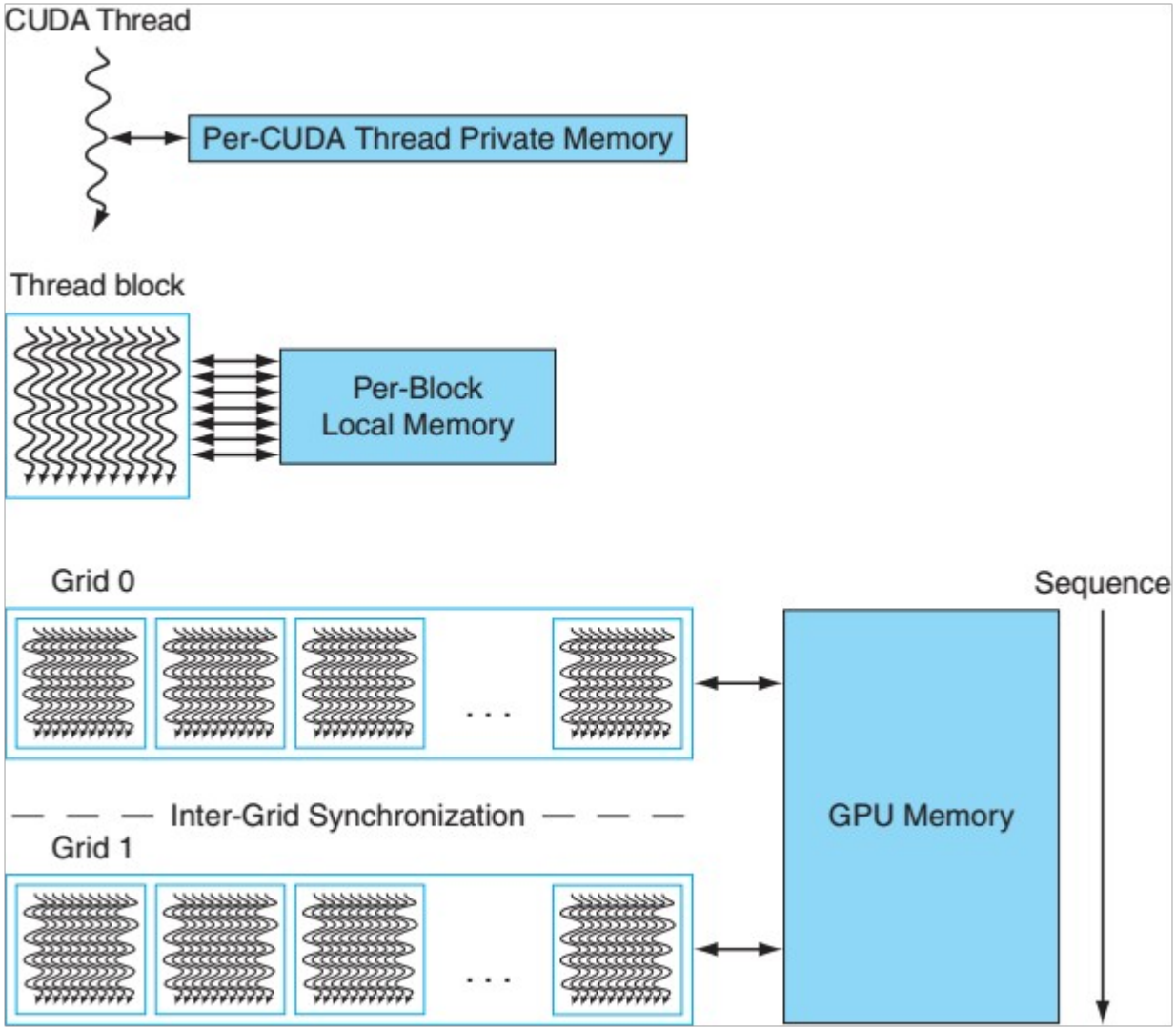
```
opcode.type d, a, b, c;
```

where `d` is the destination operand; `a`, `b`, and `c` are source operands; and the operation type is one of the following:

Type	.type Specifier
Untyped bits 8, 16, 32, and 64 bits	`.b8, .b16, .b32, .b64`
Unsigned integer 8, 16, 32, and 64 bits	`.u8, .u16, .u32, .u64`
Signed integer 8, 16, 32, and 64 bits	`.s8, .s16, .s32, .s64`
Floating Point 16, 32, and 64 bits	`.f16, .f32, .f64`

Source operands are 32-bit or 64-bit registers or a constant value. Destinations are registers, except for store instructions.

Group	Instruction	Example	Meaning	Comments
Arithmetic	arithmetic .type = .s32, .u32, .f32, .s64, .u64, .f64			
	add.type	add.f32 d, a, b	d = a + b;	
	sub.type	sub.f32 d, a, b	d = a – b;	
	mul.type	mul.f32 d, a, b	d = a * b;	
	mad.type	mad.f32 d, a, b, c	d = a * b + c;	multiply-add
	div.type	div.f32 d, a, b	d = a / b;	multiple microinstructions
	rem.type	rem.u32 d, a, b	d = a % b;	integer remainder
	abs.type	abs.f32 d, a	d = |a|;	
	neg.type	neg.f32 d, a	d = 0 - a;	
	min.type	min.f32 d, a, b	d = (a < b)? a:b;	floating selects non-NaN
	max.type	max.f32 d, a, b	d = (a > b)? a:b;	floating selects non-NaN
	setp.cmp.type	setp.lt.f32 p, a, b	p = (a < b);	compare and set predicate
	numeric .cmp = eq, ne, lt, le, gt, ge; unordered cmp = equ, neu, ltu, leu, gtu, geu, num, nan			
	mov.type	mov.b32 d, a	d = a;	move
	selp.type	selp.f32 d, a, b, p	d = p? a: b;	select with predicate
	cvt.dtype.atype	cvt.f32.s32 d, a	d = convert(a);	convert atype to dtype
Special Function	special .type = .f32 (some .f64)			
	rcp.type	rcp.f32 d, a	d = 1/a;	reciprocal
	sqrt.type	sqrt.f32 d, a	d = sqrt(a);	square root
	rsqrt.type	rsqrt.f32 d, a	d = 1/sqrt(a);	reciprocal square root
	sin.type	sin.f32 d, a	d = sin(a);	sine
	cos.type	cos.f32 d, a	d = cos(a);	cosine
	lg2.type	lg2.f32 d, a	d = log(a)/log(2)	binary logarithm
	ex2.type	ex2.f32 d, a	d = 2 ** a;	binary exponential
Logical	logic.type = .pred,.b32, .b64			
	and.type	and.b32 d, a, b	d = a & b;	
	or.type	or.b32 d, a, b	d = a | b;	
	xor.type	xor.b32 d, a, b	d = a ^ b;	
	not.type	not.b32 d, a, b	d = ~a;	one's complement
	cnot.type	cnot.b32 d, a, b	d = (a==0)? 1:0;	C logical not
	shl.type	shl.b32 d, a, b	d = a << b;	shift left
	shr.type	shr.s32 d, a, b	d = a >> b;	shift right
	memory space = .global, .shared, .local, .const; type = .b8, .u8, .s8, .b16, .b32, .b64			


CUDA Thread
Per-CUDA Thread Private Memory
Thread block
Per-Block
Local Memory
Grid 0
Sequence
Inter-Grid Synchronization
Grid 1
GPU Memory

NVIDIA Tesla V100 accelerator (2017)

GV100 GPU: 21,1 млрд транзисторов, 12-нм техпроцессу FinFET(815 mm2)

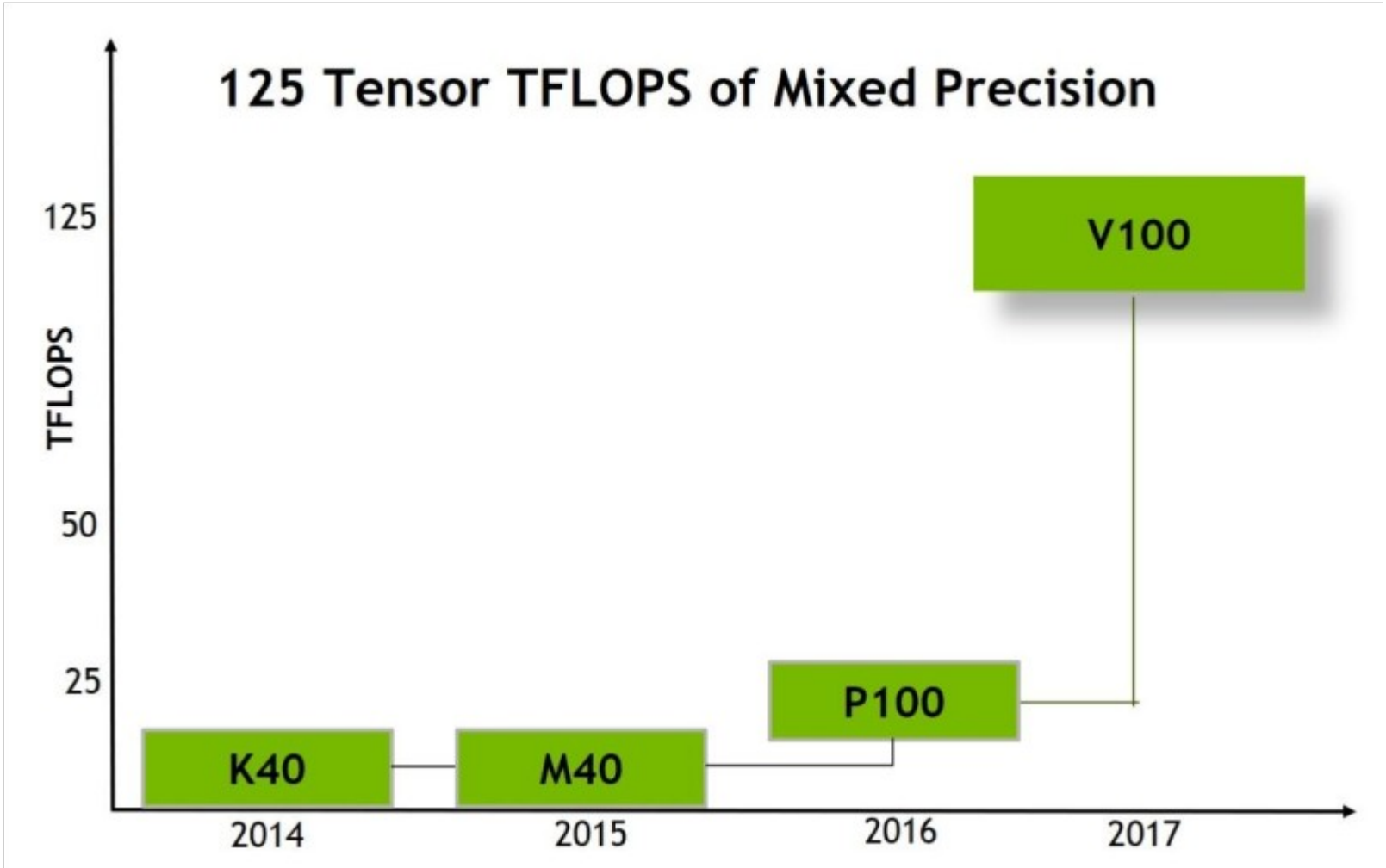
Особенности:

1) New Streaming Multiprocessor (SM) Architecture Optimized for Deep Learning
2) Second-Generation NVIDIA NVLink™
six NVLink links and total bandwidth of 300 GB/sec
3) HBM2 Memory: (High Bandwidth Memory)
Up to 16 GB HBM2 memory subsystem delivers 900 GB/sec peak memory bandwidth.
4 HBM2 модуля, 4 кристалла на каждом, расположены на той же подложке.
V100 HBM2 memory subsystem supports
    Single-Error Correcting Double-Error Detecting Code
    Eight ECC bits are created for each eight bytes of data.
    The same ECC is used to protect SM register file, L1 cache, and L2 cache.

Пиковая частота: 1530 MHz
Peak FP32 TFLOPS 15.7
Peak FP64 TFLOPS 7.8
Peak Tensor TFLOPS 125
Memory Interface 4096-bit HBM2
Memory Size Up to 16 GB
L2 Cache Size 6144 KB

NVIDIA Tesla V100 accelerator (2017)

- 7.8 TFLOPS[1] of double precision floating-point (FP64) performance
- 15.7 TFLOPS[1] of single precision (FP32) performance
- 125 Tensor TFLOPS[1]



Производительность для AI и HPC

NVIDIA Tesla V100 accelerator (2017)

V110 consists of
multiple GPU Processing Clusters (GPCs),
Texture Processing Clusters (TPCs),
Streaming Multiprocessors (SMs),
and memory controllers.

– Six GPCs

Each GPC has:

- Seven TPCs (each including two SMs)
- 14 SMs

– 84 Volta SMs

Each SM has:

- 64 FP32 cores
- 64 INT32 cores
- 32 FP64 cores
- 8 Tensor Cores
- Four texture units

– Eight 512-bit memory controllers (4096 bits total)

For 84 Sms:
5376 FP32 cores, 5376 INT32 cores, 2688 FP64 cores, 672 Tensor Cores, and 336 texture units.
Each HBM2 DRAM stack is controlled by a pair of memory controllers.
The full GV100 GPU includes a total of 6144 KB of L2 cache.

NVIDIA Tesla V100 accelerator (2017)

NVIDIA Tesla V100 accelerator (2017)

NVIDIA Tesla V100 accelerator (2017)

Тензорные ядра

The Tesla V100 GPU contains 640 Tensor Cores: 8 per SM and 2 per each processing block (partition) within an SM.

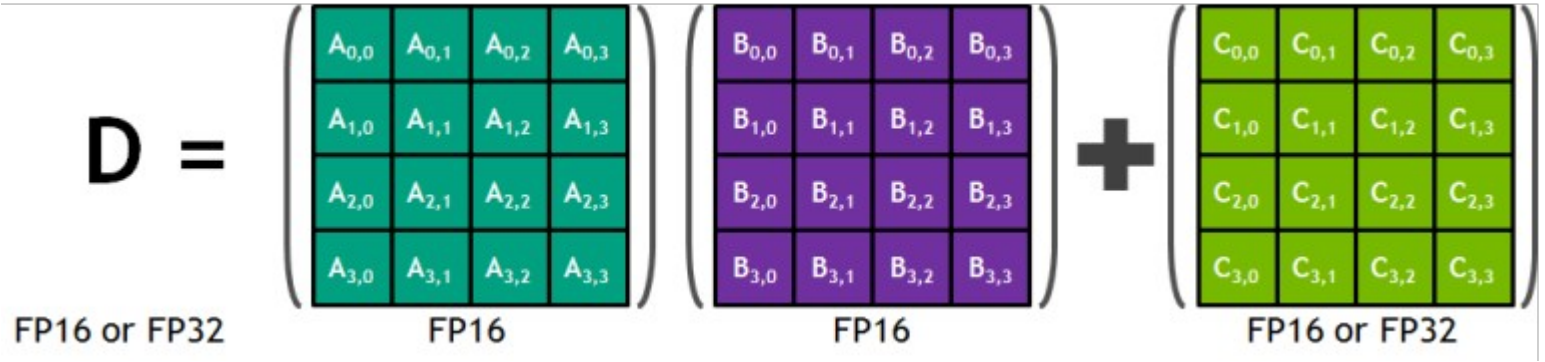
Each Tensor Core performs **64 floating point FMA operations per clock**, and eight Tensor Cores in an SM perform a total of 512 FMA operations (= 1024 individual floating point operations) per clock.

V100's Tensor Cores deliver up to 125 Tensor TFLOPS for training and inference apps.

Each Tensor Core operates on a 4x4 matrix and performs the following operation:

D = A×B + C

where **A**, **B**, **C**, and **D** are 4x4 matrices.



Подсистема памяти:

Комбинированный L1 кэш данных и Shared Memory

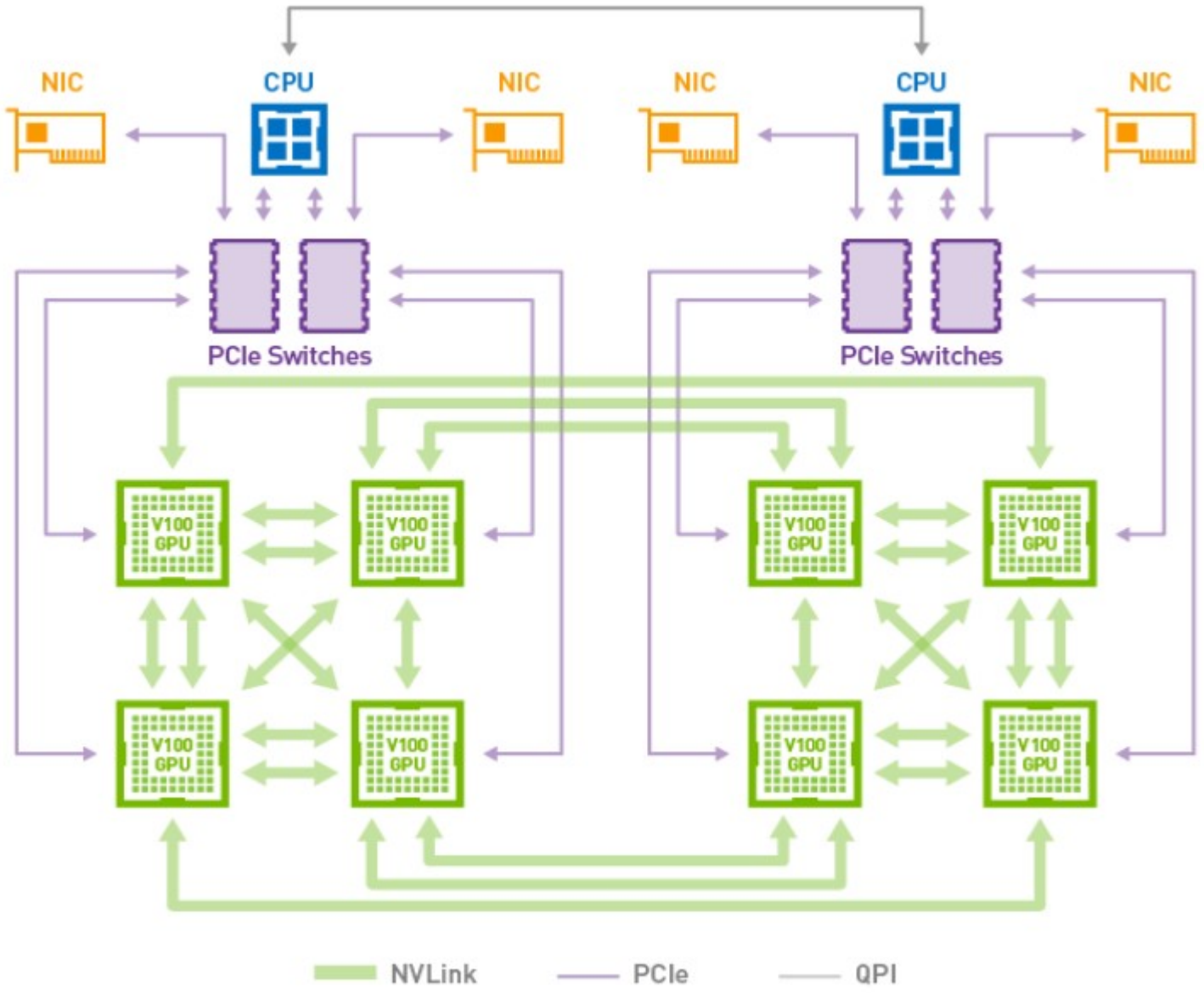
Объем: 128 Кб/SM

Конфигурируема до Shared Memory <= 96 Кб/SM (напр. текстуры)

Одновременное выполнение FP32 и INT32 операций.

NVIDIA Tesla V100 accelerator (2017)

NVLINKs



NVIDIA Tesla V100 accelerator (2017)