Type	More descriptive name	Closest old term outside of GPUs	Official CUDA/NVIDIA GPU term	Book definition
Program abstractions	Vectorizable Loop	Vectorizable Loop	Grid	A vectorizable loop, executed on the GPU, made up of one or more Thread Blocks (bodies of vectorized loop) that can execute in parallel.
	Body of Vectorized Loop	Body of a (Strip-Mined) Vectorized Loop	Thread Block	A vectorized loop executed on a multithreaded SIMD Processor, made up of one or more threads of SIMD instructions. They can communicate via Local Memory.
	Sequence of SIMD Lane Operations	One iteration of a Scalar Loop	CUDA Thread	A vertical cut of a thread of SIMD instructions corresponding to one element executed by one SIMD Lane. Result is stored depending on mask and predicate register.
Machine object	A Thread of SIMD Instructions	Thread of Vector Instructions	Warp	A traditional thread, but it contains just SIMD instructions that are executed on a multithreaded SIMD Processor. Results stored depending on a per-element mask.
	SIMD Instruction	Vector Instruction	PTX Instruction	A single SIMD instruction executed across SIMD Lanes.
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).

Figure 4.12 Quick guide to GPU terms used in this chapter. We use the first column for hardware terms. Four groups cluster these 11 terms. From top to bottom: Program Abstractions, Machine Objects, Processing Hardware, and Memory Hardware. Figure 4.21 on page 309 associates vector terms with the closest terms here, and Figure 4.24 on page 313 and Figure 4.25 on page 314 reveal the official CUDA/NVIDIA and AMD terms and definitions along with the terms used by OpenCL.

abstraction:

Grid: 可被平行分解為多个 thread blocks，並平行地在 GPU 上執行

thread block: 可分配給 SM 執行，由 SIMD thread 組成，之間共享 - local mem 用作 comm
= - 群对於資料執行相同指令的 thread 構成的

SIMD thread: 对於資料執行的一个指令為 SIMD thread of instr.

Wrap: - 般的 thread，只是裡面執行的是 SIMD 指令，會以 lock-step fashion 執行（所有 warp execution 完成，下一指令才能執行

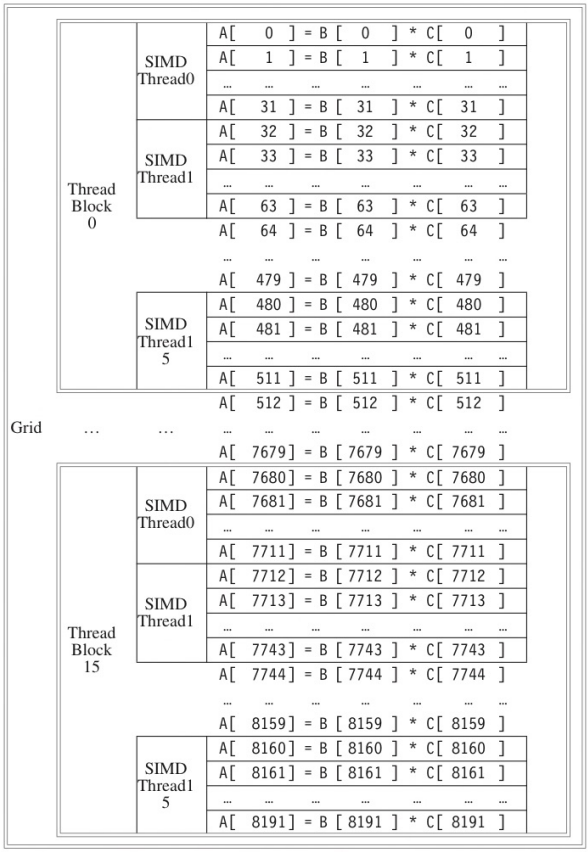

Figure 4.13 The mapping of a Grid (vectorizable loop), Thread Blocks (SIMD basic blocks), and threads of SIMD instructions to a vector–vector multiply, with each vector being 8192 elements long. Each thread of SIMD instructions calculates 32 elements per instruction, and in this example each Thread Block contains 16 threads of SIMD instructions and the Grid contains 16 Thread Blocks. The hardware Thread Block Scheduler assigns Thread Blocks to multithreaded SIMD Processors and the hardware Thread Scheduler picks which thread of SIMD instructions to run each clock cycle within a SIMD Processor. Only SIMD Threads in the same Thread Block can communicate via Local Memory. (The maximum number of SIMD Threads that can execute simultaneously per Thread Block is 16 for Tesla-generation GPUs and 32 for the later Fermi-generation GPUs.)

會有一 high-level scheduler 分配 thread block 給 SM
而 SM 中的 warp scheduler 才會分配 thread block 中的 SIMD thread
至 wrap execution unit 執行

SM structure

keep 一個 scoreboard，∵ 要由 SW (complier) 處理太多時 (CPU 就比 GPU 太遠)

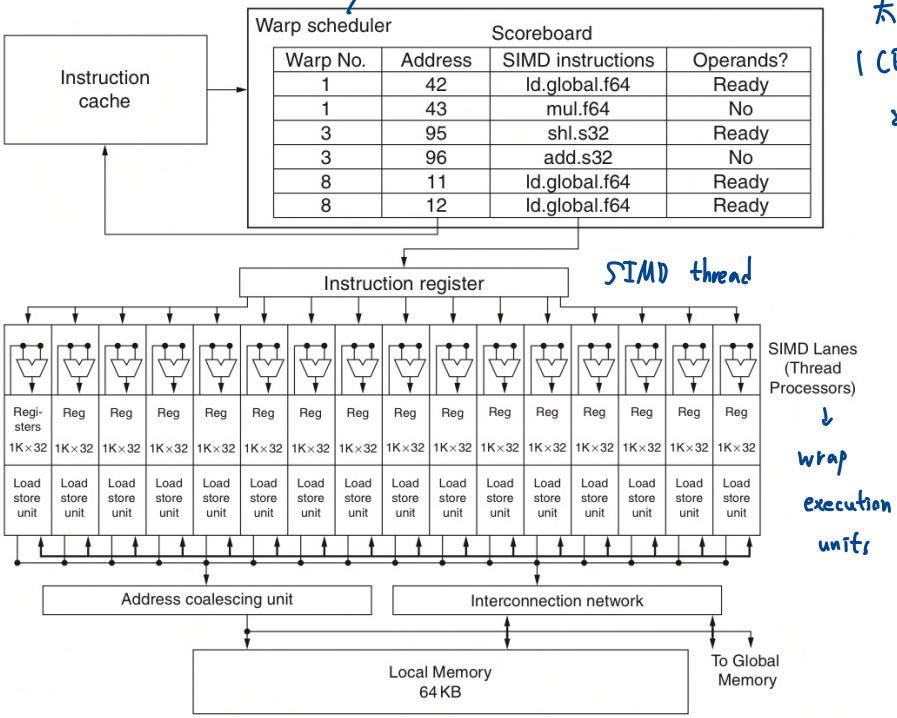

Figure 4.14 Simplified block diagram of a Multithreaded SIMD Processor. It has 16 SIMD lanes. The SIMD Thread Scheduler has, say, 48 independent threads of SIMD instructions that it schedules with a table of 48 PCs.

Warp 觀念：

shows the floor plan of the GTX 480 implementation of the Fermi architecture.

Dropping down one more level of detail, the machine object that the hardware creates, manages, schedules, and executes is a *thread of SIMD instructions*. It is a traditional thread that contains exclusively SIMD instructions. These threads of SIMD instructions have their own PCs and they run on a multithreaded SIMD Processor. The *SIMD Thread Scheduler* includes a scoreboard that lets it know which threads of SIMD instructions are ready to run, and then it sends them off to a dispatch unit to be run on the multithreaded SIMD Processor. It is identical to a hardware thread scheduler in a traditional multithreaded processor (see Chapter 3), just that it is scheduling threads of SIMD instructions. Thus, 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 SIMD instructions of these threads are 32 wide, so each thread of SIMD instructions in this example would compute 32 of the elements of the computation. In this example, Thread Blocks would contain 512/32 = 16 SIMD threads (see Figure 4.13).

warp size：一次 compute 的 data size = 32 elements

Since the thread consists of SIMD instructions, the SIMD Processor must have parallel functional units to perform the operation. We call them *SIMD Lanes*, and they are quite similar to the Vector Lanes in Section 4.2.

Since by definition the threads of SIMD instructions are independent, the SIMD Thread Scheduler can pick whatever thread of SIMD instructions is ready, and need not stick with the next SIMD instruction in the sequence within a thread. The SIMD Thread Scheduler includes a scoreboard (see Chapter 3) to keep track of up to 48 threads of SIMD instructions to see which SIMD instruction is ready to go. This scoreboard is needed because memory access instructions can take an unpredictable number of clock cycles due to memory bank conflicts, for example. Figure 4.16 shows the SIMD Thread Scheduler picking threads of SIMD instructions in a different order over time. 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 NVIDIA Fermi GPU includes an L2 cache (see Section 4.7).

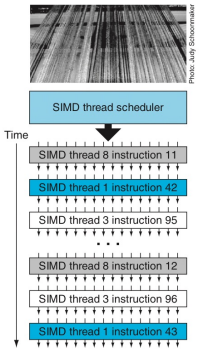

Figure 4.16 Scheduling of threads of SIMD instructions. The scheduler selects a ready thread of SIMD instructions and issues an instruction synchronously to all the SIMD Lanes executing the SIMD thread. Because threads of SIMD instructions are independent, the scheduler may select a different SIMD thread each time.