Heterogeneous
Parallel
Programming

Lesson 1.4

Introduction to CUDA

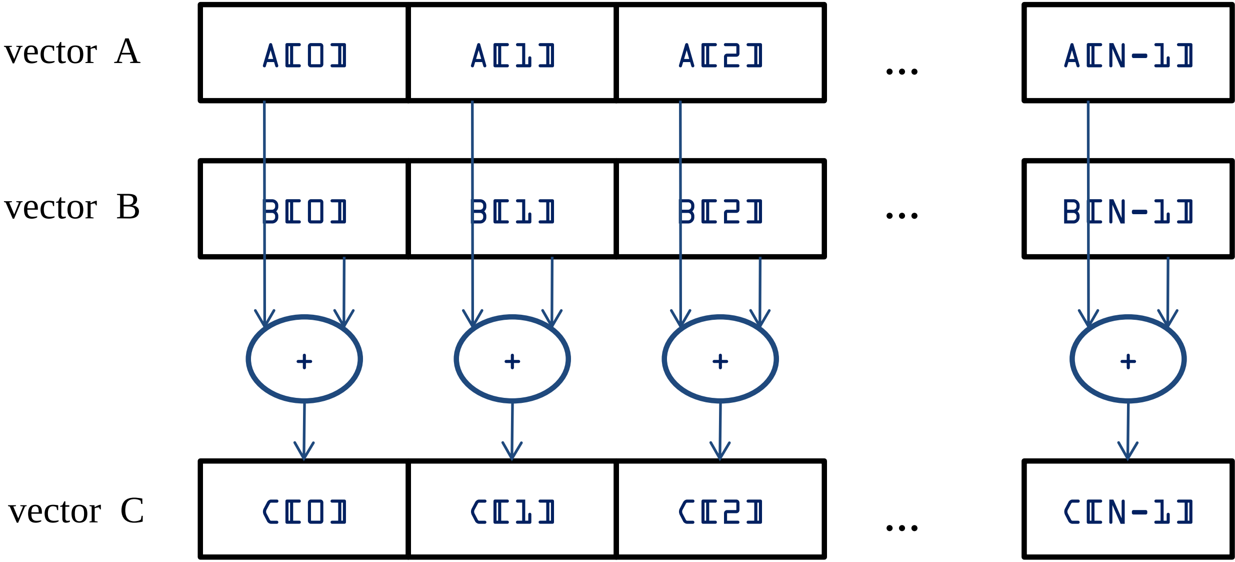
- Data Parallelism and Threads

Wen-mei Hwu - University of Illinois at Urbana-Champaign

Objective

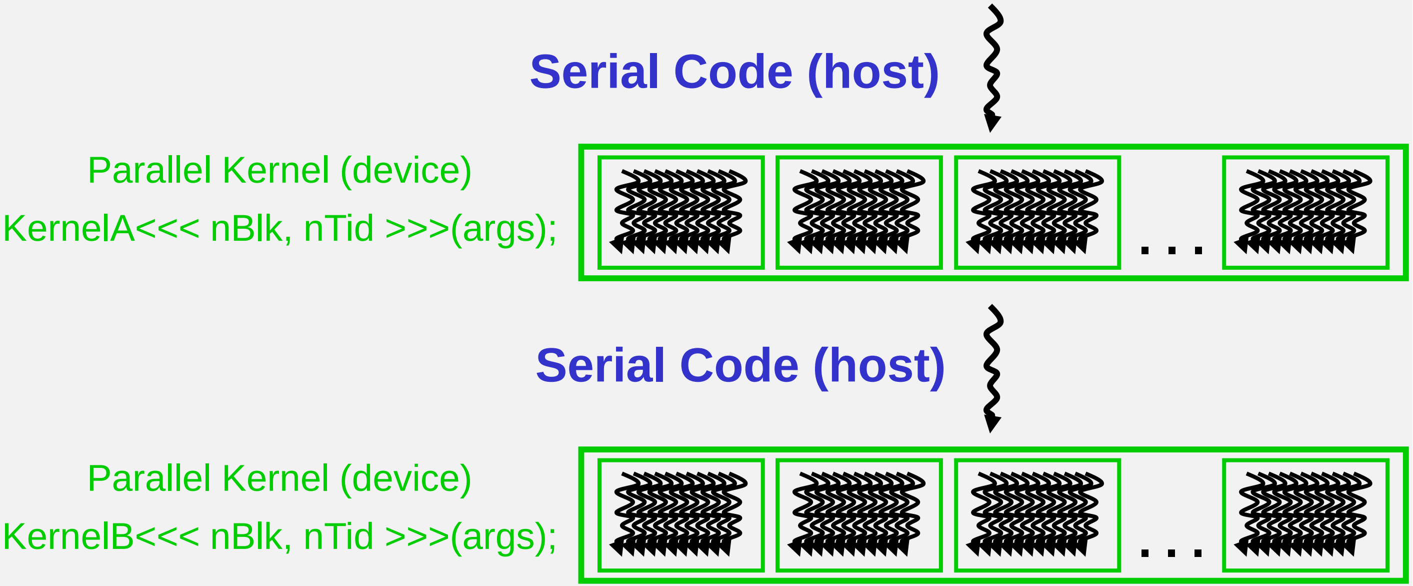
- To learn about data parallelism and the basic features of CUDA C, a heterogeneous parallel programming interface that enables exploitation of data parallelism
  - Hierarchical thread organization
  - Main interfaces for launching parallel execution
  - Thread index to data index mapping

Data Parallelism - Vector Addition Example



CUDA /OpenCL - Execution Model

- Heterogeneous host+device application C program
  - Serial parts in **host** C code
  - Parallel parts in **device** SPMD kernel C code



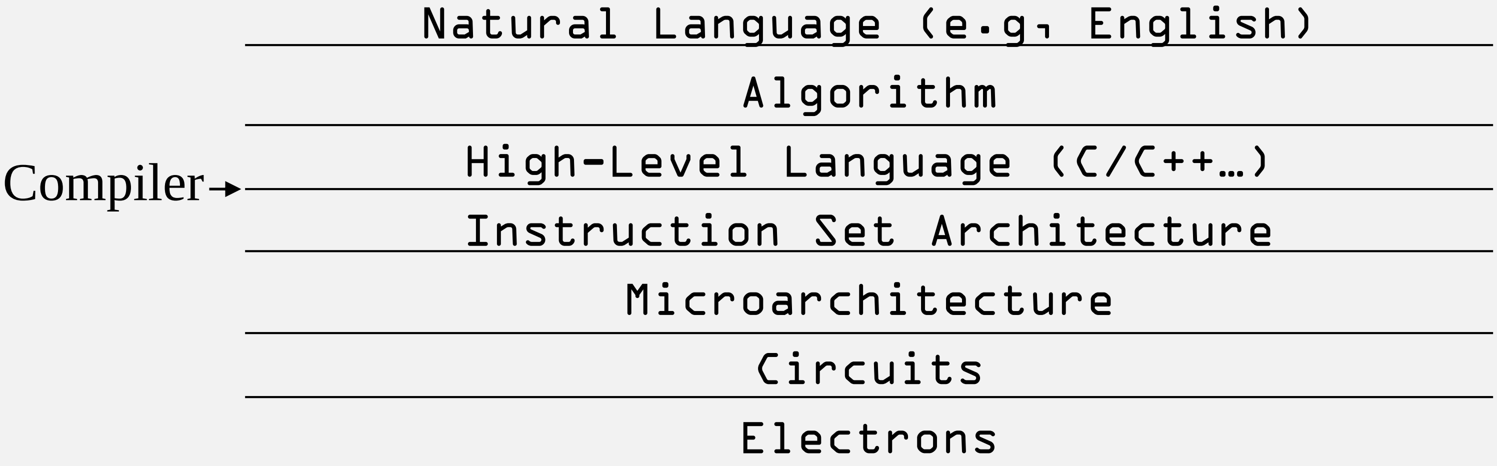
Serial Code (host)

Parallel Kernel (device)
KernelA<<< nBlk, nTid >>>(args);

Serial Code (host)

Parallel Kernel (device)
KernelB<<< nBlk, nTid >>>(args);

From Natural Language to Electrons



Natural Language (e.g, English)

Algorithm

High-Level Language (C/C++…)

Compiler→

Instruction Set Architecture

Microarchitecture

Circuits

Electrons

©Yale Patt and Sanjay Patel, *From bits and bytes to gates and beyond*

The ISA

- An Instruction Set Architecture (ISA) is a contract between the hardware and the software.

- As the name suggests, it is a set of instructions that the architecture (hardware) can execute.

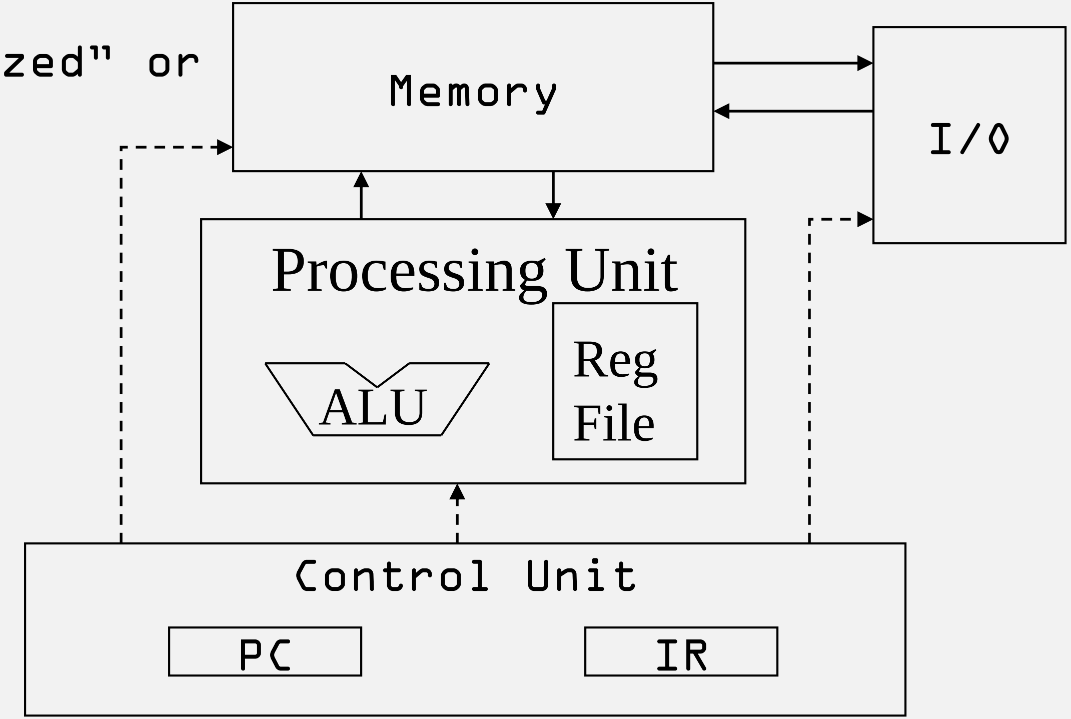
A program at the ISA level

- A program is a set of instructions stored in memory that can be read, interpreted, and executed by the hardware.

- Program instructions operate on data stored in memory or provided by Input/Output (I/O) device.

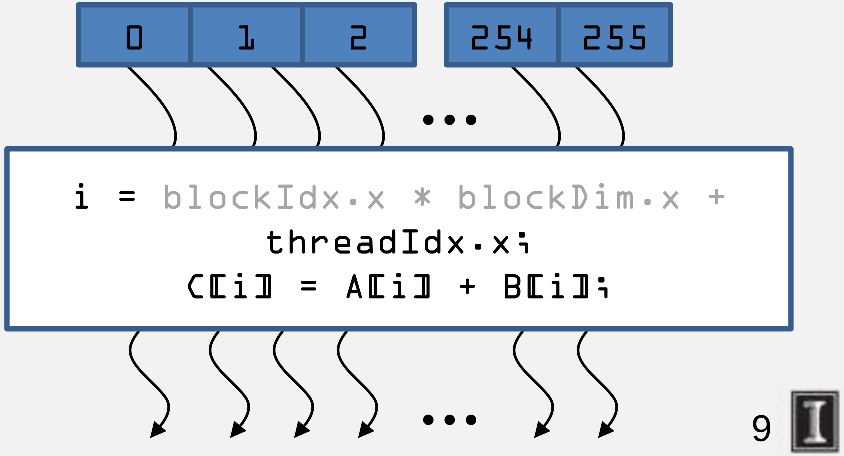
A Von-Neumann Processor

A thread is a “virtualized” or “abstracted” Von-Neumann Processor



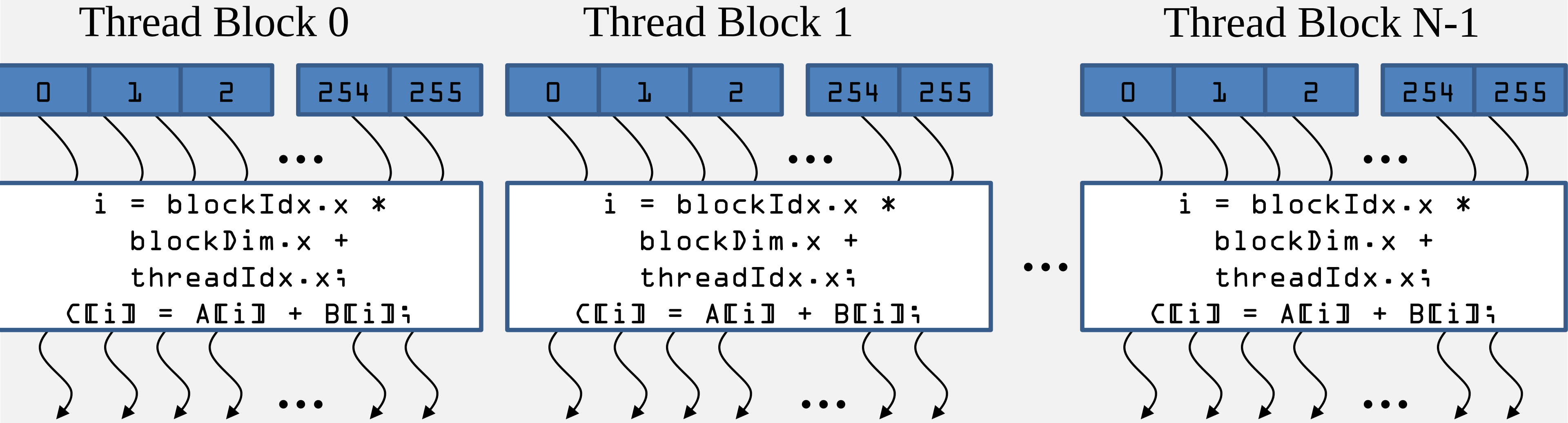
Arrays of Parallel Threads

- A CUDA kernel is executed by a grid (array) of threads
  - All threads in a grid run the same kernel code (SPMD)
  - Each thread has indexes that it uses to compute memory addresses and make control decisions



```
i = blockIdx.x * blockDim.x +
        threadIdx.x;
C[i] = A[i] + B[i];
```

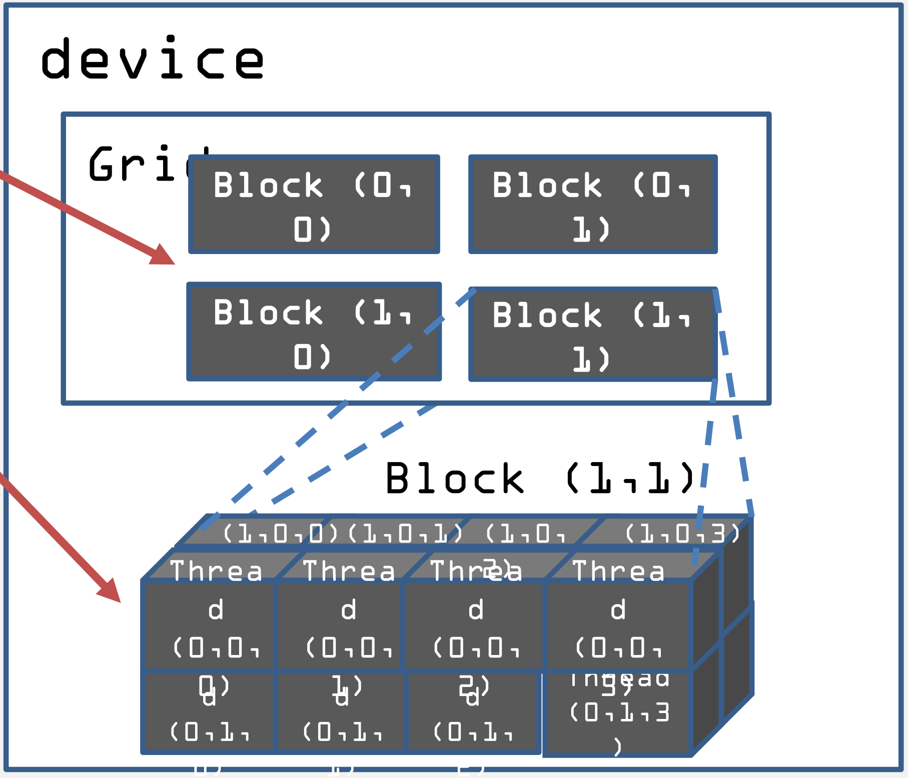
Thread Blocks: Scalable Cooperation



- Divide thread array into multiple blocks
  - Threads within a block cooperate via **shared memory, atomic operations** and **barrier synchronization**
  - Threads in different blocks do not interact

blockIdx and threadIdx

- Each thread uses indices to decide what data to work on
  - blockIdx: 1D, 2D, or 3D (CUDA 4.0)
  - threadIdx: 1D, 2D, or 3D
- Simplifies memory addressing when processing multidimensional data
  - Image processing
  - Solving PDEs on volumes
  - …



device

Grid

Block (0, 0)

Block (0, 1)

Block (1, 0)

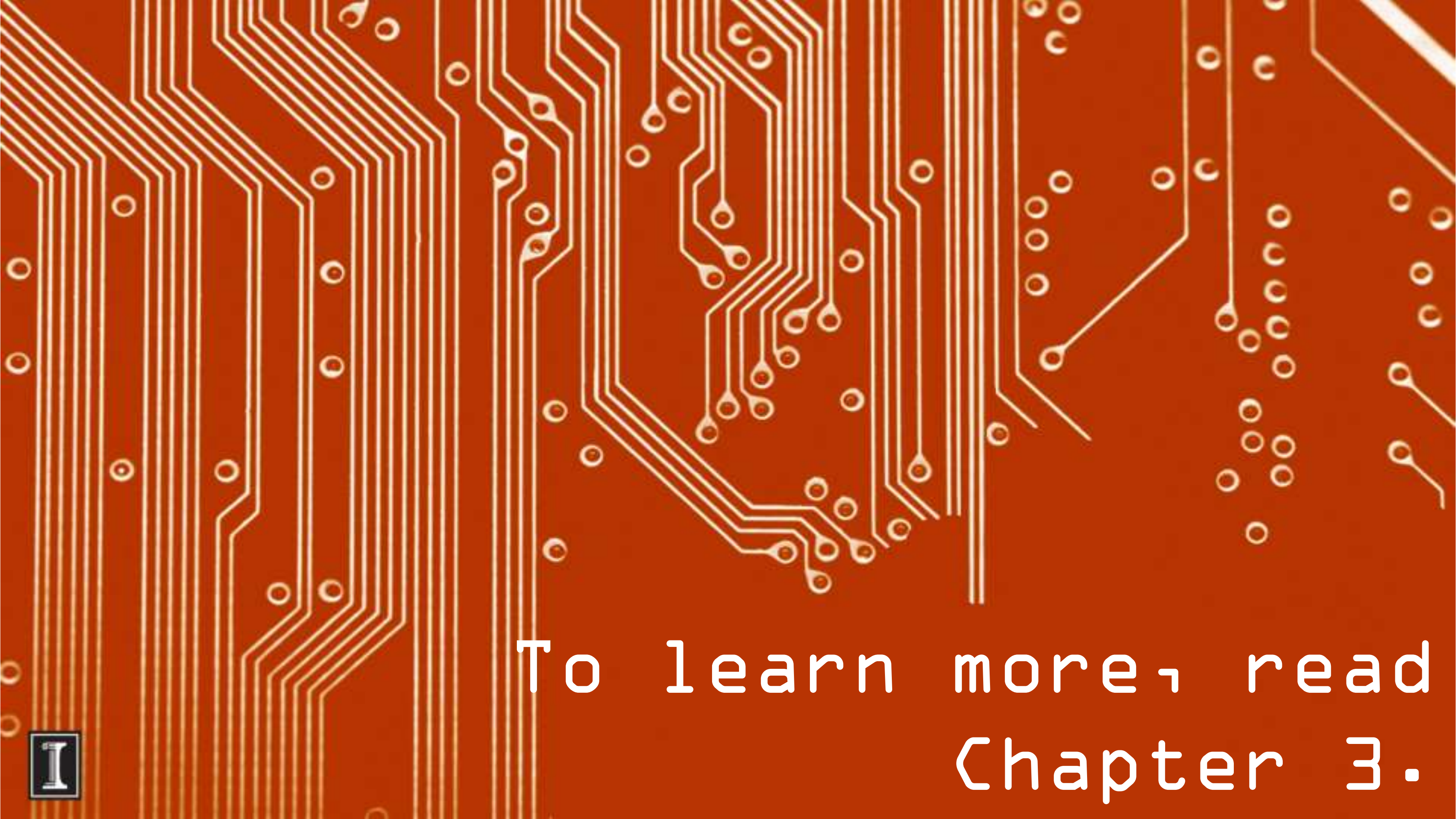
Block (1, 1)

Block (1,1)

(1,0,0) (1,0,1) (1,0,2) (1,0,3)

Thread (0,0,0) Thread (0,0,1) Thread (0,0,2) Thread (0,0,3)

Thread (0,1,0) Thread (0,1,1) Thread (0,1,2) Thread (0,1,3)



To learn more, read Chapter 3.