NVIDIA Kepler GK110 Architecture (chip)
(As used in coit-grid08.uncc.edu K20 GPU server)

Highlights – To discuss in class

Extracted directly from:


Designed for performance and power efficiency

7.1 billion transistors

Over 1 TFlop of double precision throughput
3x performance per watt of Fermi

New features in Kepler GK110:
• Dynamic Parallelism
• Hyper-Q with GK110 Grid Management Unit (GMU)
• NVIDIA GPU Direct™ RDMA

Kepler GK110 Chip
Kepler GK110 supports the new CUDA Compute Capability 3.5

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

GTX 470/480s have GT100s
C2050s on grid06 and grid07 are compute cap 2.0
New streaming multiprocessor (now called SMX)

192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST).

Full Kepler GK110 has 15 SMXs
Some products may have 13 or 14 SMXs

Quad Warp Scheduler

The SMX schedules threads in groups of 32 parallel threads called warps.

Each SMX features four warp schedulers and eight instruction dispatch units, allowing four warps to be issued and executed concurrently. (128 threads)

Kepler GK110 allows double precision instructions to be paired with other instructions.
• Each thread can access up to 255 registers (x4 of Fermi)
• New Shuffle instruction which allows threads within a warp to share data without passing data through shared memory:

• Atomic operations: Improved by 9x to **one operation per clock** – fast enough to use frequently with kernel inner loops
Texture units improvements

- Not considered in class
- For image processing
- Speed improvements when programs need to operate on image data

---

**Kepler Memory Hierarchy**

New: 48 KB Read-only memory cache
Compiler/programmer can use to advantage

**Shared memory/L1 cache split:**
Each SMX has 64 KB on-chip memory, that can be configured as:

- 48 KB of Shared memory with 16 KB of L1 cache,
- 16 KB of shared memory with 48 KB of L1 cache
- (new) a 32KB / 32KB split between shared memory and L1 cache.
Dynamic Parallelism

• Fermi could only launch one kernel at a time on a single device. Kernel had to complete before calling for another GPU task.

• “In Kepler GK110 any kernel can launch another kernel, and can create the necessary streams, events and manage the dependencies needed to process additional work without the need for host CPU interaction.”

• “.. makes it easier for developers to create and optimize recursive and data-dependent execution patterns, and allows more of a program to be run directly on GPU.”

“Dynamic Parallelism allows more parallel code in an application to be launched directly by the GPU onto itself (right side of image) rather than requiring CPU intervention (left side of image).”

Dynamic Parallelism

GPU Adapts to Data, Dynamically Launches New Threads

CPU → Fermi GPU

CPU → Kepler GPU

Control must be transferred back to CPU before a new kernel can execute

Only return to CPU when all GPU operations are completed. Why is this faster?
“With Dynamic Parallelism, the grid resolution can be determined dynamically at runtime in a data dependent manner. Starting with a coarse grid, the simulation can “zoom in” on areas of interest while avoiding unnecessary calculation in areas with little change .... ”

Hyper-Q

“The Fermi architecture supported 16-way concurrency of kernel launches from separate streams, but ultimately the streams were all multiplexed into the same hardware work queue.”

“Kepler GK110 ... Hyper-Q increases the total number of connections (work queues) ... by allowing 32 simultaneous, hardware-managed connections.”

“... allows connections from multiple CUDA streams, from multiple Message Passing Interface (MPI) processes, or even from multiple threads within a process.

Applications that previously encountered false serialization across tasks, thereby limiting GPU utilization, can see up to a 32x performance increase without changing any existing code.”
Hyper-Q

"Each CUDA stream is managed within its own hardware work queue ... "

Fermi Model

<table>
<thead>
<tr>
<th>STREAM 1</th>
<th>STREAM 2</th>
<th>STREAM 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>P</td>
<td>X</td>
</tr>
<tr>
<td>B</td>
<td>Q</td>
<td>Y</td>
</tr>
<tr>
<td>C</td>
<td>R</td>
<td>Z</td>
</tr>
</tbody>
</table>

Kepler Hyper-Q Model

<table>
<thead>
<tr>
<th>STREAM 1</th>
<th>STREAM 2</th>
<th>STREAM 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>P</td>
<td>X</td>
</tr>
<tr>
<td>B</td>
<td>Q</td>
<td>Y</td>
</tr>
<tr>
<td>C</td>
<td>R</td>
<td>Z</td>
</tr>
</tbody>
</table>

Single hardware work queue

Each stream receives its own work queue
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.

NVIDIA GPUDirect™

“Kepler GK110 supports the RDMA feature in NVIDIA GPUDirect, which is designed to improve performance by allowing direct access to GPU memory by third-party devices such as IB adapters, NICs, and SSDs. When using CUDA 5.0, GPUDirect provides the following important features:

· Direct memory access (DMA) between NIC and GPU without the need for CPU-side data buffering. (Huge improvement for GPU-only Servers)
· Significantly improved MPISend/MPIRecv efficiency between GPU and other nodes in a network.
· Eliminates CPU bandwidth and latency bottlenecks
· Works with variety of 3rd-party network, capture, and storage devices.”
GPUDirect™
Direct Transfers between GPU and 3rd Party Devices

System Memory

GDDR5 Memory

GPU1

GPU2

Network Card

Server 1

System Memory

GDDR5 Memory

GPU2

GPU1

Network Card

Server 2

“GPUDirect RDMA allows direct access to GPU memory from 3rd-party devices such as network adapters, which translates into direct transfers between GPUs across nodes as well.”

Questions