GPU Architecture continued

Balaji

IISc Bangalore

A checklist

Table 6.1 A checklist of optimizations.

Optimization	Benefit to compute cores	Benefit to memory	Strategies
Maximizing occupancy	More work to hide pipeline latency	More parallel memory accesses to hide DRAM latency	Tuning usage of SM resources such as threads per block, shared memory per block, and registers per thread
Enabling coalesced global memory accesses	Fewer pipeline stalls waiting for global memory accesses	Less global memory traffic and better utilization of bursts/ cache lines	Transfer between global memory and shared memory in a coalesced manner and performing uncoalesced accesses in shared memory (e.g., corner turning) Rearranging the mapping of threads to data Rearranging the layout of the data

A checklist

Minimizing control divergence	High SIMD efficiency (fewer idle cores during SIMD execution)	–	Rearranging the mapping of threads to work and/or data Rearranging the layout of the data
Tiling of reused data	Fewer pipeline stalls waiting for global memory accesses	Less global memory traffic	Placing data that is reused within a block in shared memory or registers so that it is transferred between global memory and the SM only once
Privatization (covered later)	Fewer pipeline stalls waiting for atomic updates	Less contention and serialization of atomic updates	Applying partial updates to a private copy of the data and then updating the universal copy when done
Thread coarsening	Less redundant work, divergence, or synchronization	Less redundant global memory traffic	Assigning multiple units of parallelism to each thread to reduce the price of parallelism when it is incurred unnecessarily

Memory coalescing

- Consecutive threads access consecutive memory locations.
- Benefit: Multiple accesses treated like one
- Reason: Multiple bursts of data in DRAM can be fetched in one go.
- Example: Corner turning in tiled matrix multiplication where first matrix is in row major and second in column major

Parallelization In DRAM

- Two forms of parallel organization: banks and channels
- Channel: memory controller with a bus that connects a set of DRAM banks to the processor
- Multiple channels and many banks associated with each bank.
- If accesses are made todifferent banks of the same channel, can make up for access latency.
- Enough threads needed to make accesses from different channels aka occupancy.

Demo pictures

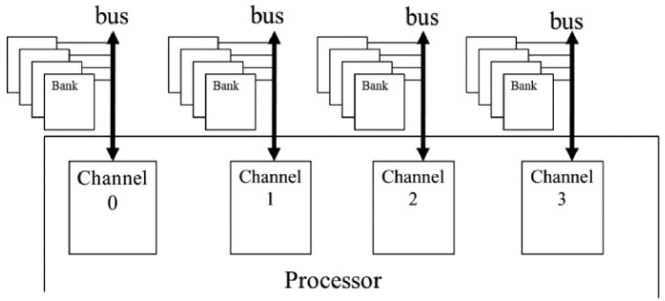

FIGURE 6.7

Channels and banks in DRAM systems.

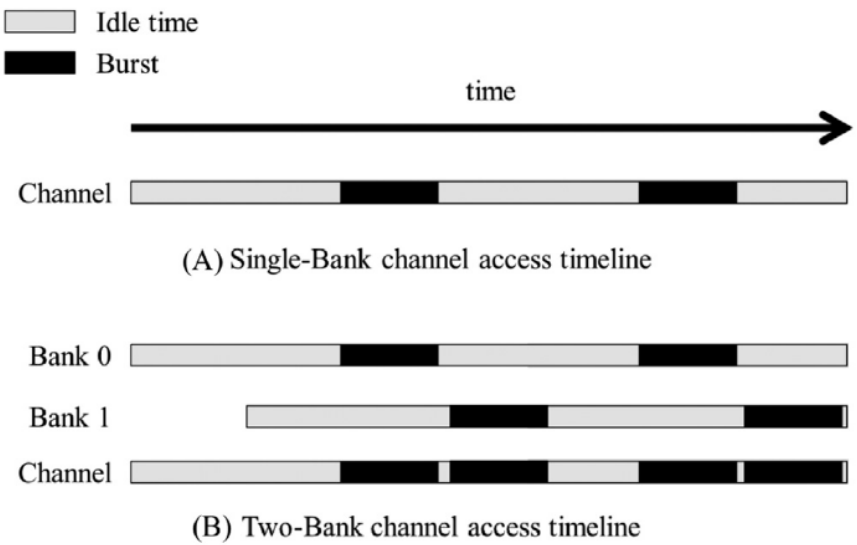

FIGURE 6.8

Banking improves the utilization of data transfer bandwidth of a channel.

Thread Coarsening

Serializing work of a thread and reducing the parallelism
Pros/Reason:

- Reduces the redundant workload that occurs with parallelization, which can reduce performance if blocks are not executed in parallel.

Cons:

- Can lead to underutilization of hardware resources.
- More serialization might require more registers and this could affect occupancy.

Constant Memory And Caching

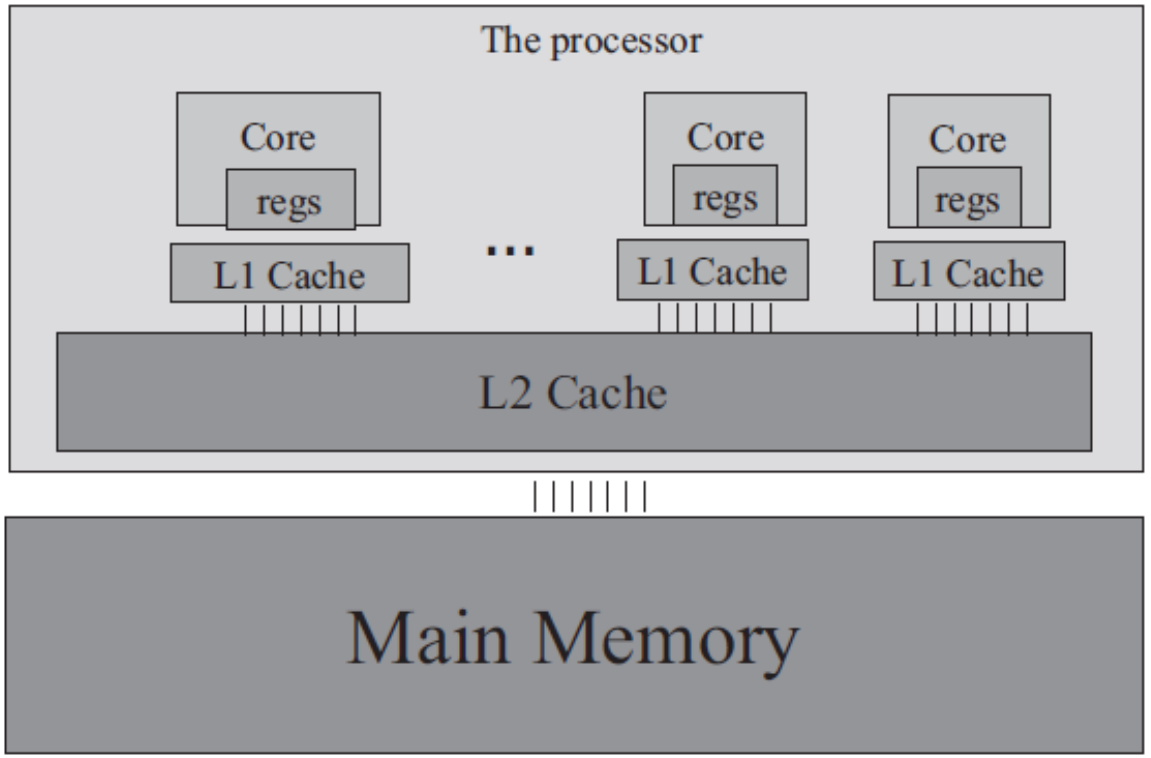

FIGURE 7.10

A simplified view of the cache hierarchy of modern processors.

Constant Memory And Caching

- Constant memory: Read-only memory, cached in L1 cache, stored in DRAM, aggressively cached during kernel execution.
- Small in size, around 64 KB
- Benefit: Reduces the number of accesses to global memory.
- L1 cache - 1 per core, 16-64 KB size, speed close to processor in latency and bandwidth
- L2 cache - shared among multiple cores or SMs, 100s of KBs to small number of MBs in size, can take 10s of clock cycles for an access
- Some modern GPUs have L3 cache which is 100s of MBs in size.
- Special constant cache on GPUs for constant memory variables