AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

HiPANQ

Overview of NVIDIA GPU Architecture and Introduction to CUDA/OpenCL Programming, and Parallelization of LDPC codes

Ian Glendinning

Outline

- NVIDIA GPU cards
- CUDA & OpenCL
- Parallel Implementation of LDPC codes

NVIDIA GPU Card Families

- **GeForce** - gaming graphics processing products Nvidia is best known for
- **Quadro** - computer-aided design workstation graphics processing products
- **Tesla** – General Purpose GPU for high-end image generation applications

GeForce

- GeForce 400 Series
  - 11th generation of GeForce, introducing the Fermi architecture, with GF-codenamed chips
  - First cards were GeForce GTX 470 and GTX 480, released April 2010, based on the Fermi architecture, codenamed GF100, 448 & 480 cores
  - Michael Rauter (AIT) uses a GTX 460, released July 2010, based on GF104 architecture, with 336 cores, lower power, better performance

- GeForce 500 Series
  - First card was GTX 580, Nov. 2010, 512 cores, GF110 architecture
  - Madrid have a GeForce GTX 570, which has 480 cores

GeForce

- GeForce 600 Series
  - Introducing the Kepler architecture, with GK-codenamed chips
  - The series contains products with the older Fermi architecture
  - First Kepler card was GeForce GTX 680, released March 2012, with a GK104 architecture and 1536 cores
  - Madrid have a GTX 670, released May 2012, based on GK104 architecture, with 1344 cores

Quadro

- Essentially the same hardware at a premium price for professional markets
- Driver software and firmware to selectively enable features vital to segments of the workstation market, which prevent high-end customers from using the less expensive products
- A system used for gaming can shut down textures, shading, or rendering after only approximating a final output, but algorithms on a CAD-oriented card tend to complete all rendering operations, prioritising accuracy and rendering quality over speed
- Christoph Pacher has a Quadro FX 580, with 32 cores, G96 based on GeForce 9500 (9 series), and Ian Glendinning has a Quadro 2000M, with 192 cores, GF106GLM, like GeForce GTS 450

Tesla

- Based on high-end GPUs from the GeForce 8 series upwards, as well as the Quadro family
- NVIDIA's first dedicated General-Purpose GPU
- The main difference between Tesla and GeForce/Quadro was the lack of ability to output images to a display, but the latest C-class products include a Dual-Link DVI port
- The main difference between Fermi-based Tesla cards and the GeForce 500 series is the unlocked double precision performance, giving ½ of peak single-precision performance, compared with 1/8 of peak for GeForce cards
- Tesla cards also have ECC-protected memory and up to 6 GByte on-board memory

NVIDIA GPU Architectures

- G80
  - Introduced in the GeForce 8800 (8 series), Nov. 2006
  - First GPU to support C
  - First GPU to replace separate vertex and pixel pipelines by a single unified processor
  - First GPU to use a scalar thread processor, eliminating the need for programmers to manually manage vector registers
  - Introduced the single-instruction multiple-thread (SIMT) execution model
  - Introduced shared memory and barrier synchronization for inter-thread communication

- GT200
  - Introduced in the GeForce GTX 280, June 2008
  - More cores, threads, added hardware memory-access coalescing and double precision floating point support

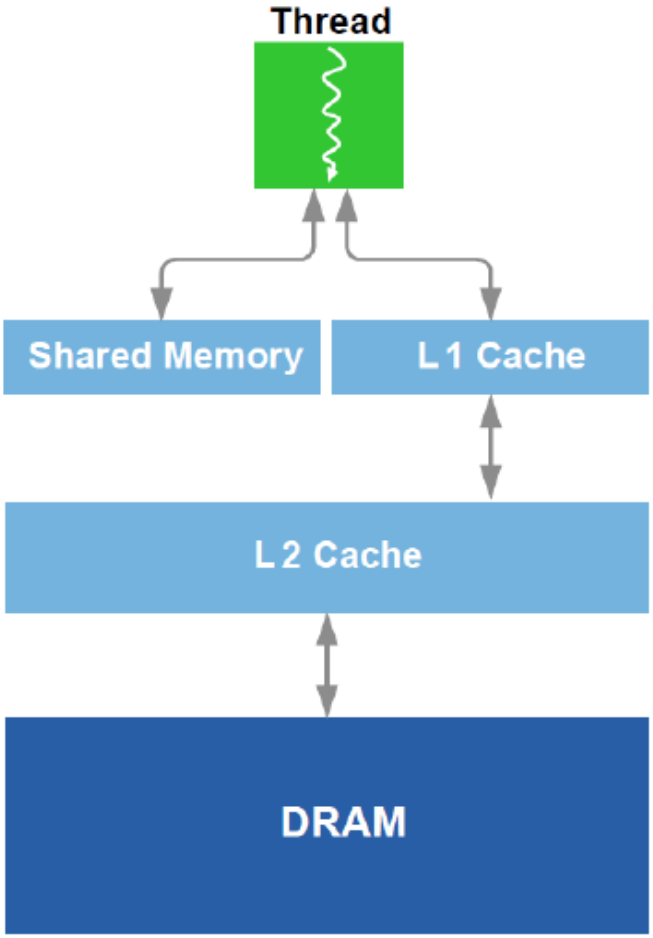
NVIDIA GPU Architectures

- Fermi
  - Up to 512 CUDA cores, each executing one floating point or integer instruction per clock
  - Cores organized into streaming multiprocessors (SMs) with 32 cores
  - Six 64-bit memory partitions for a 384-bit memory interface supporting up to 6 GB of GDDR5 DRAM memory
  - A host interface connects the GPU to the CPU via PCI-Express
  - The GigaThread global scheduler distributes thread blocks to SM thread schedulers
  - Introduced shared memory and barrier synchronization for inter-thread communication
  - GF100's architecture is built from a number of hardware blocks called Graphics Processing Clusters (GPCs), containing a raster engine and up to four SMs
  - Unified address space enables full C++ support

Fermi GF100 Architecture

Fermi GF100 Streaming Multiprocessor

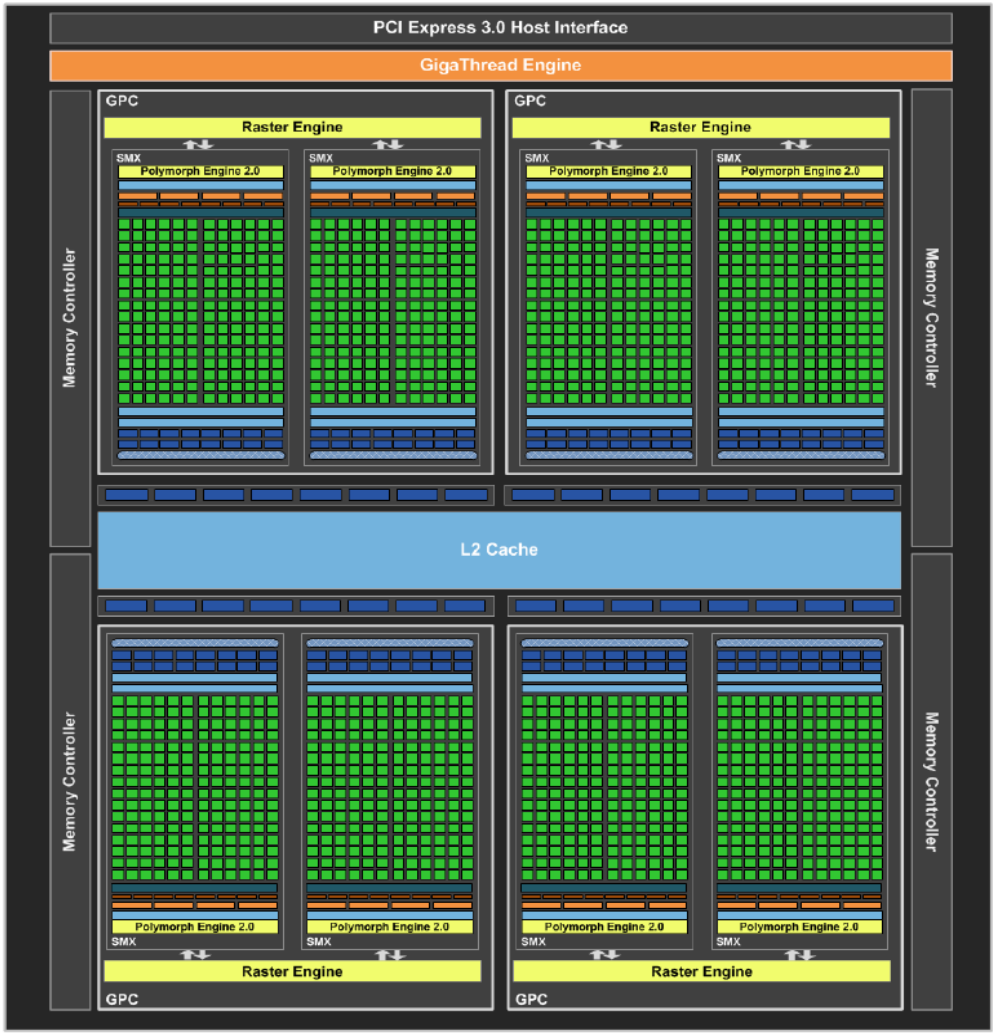
Fermi Memory Hierarchy



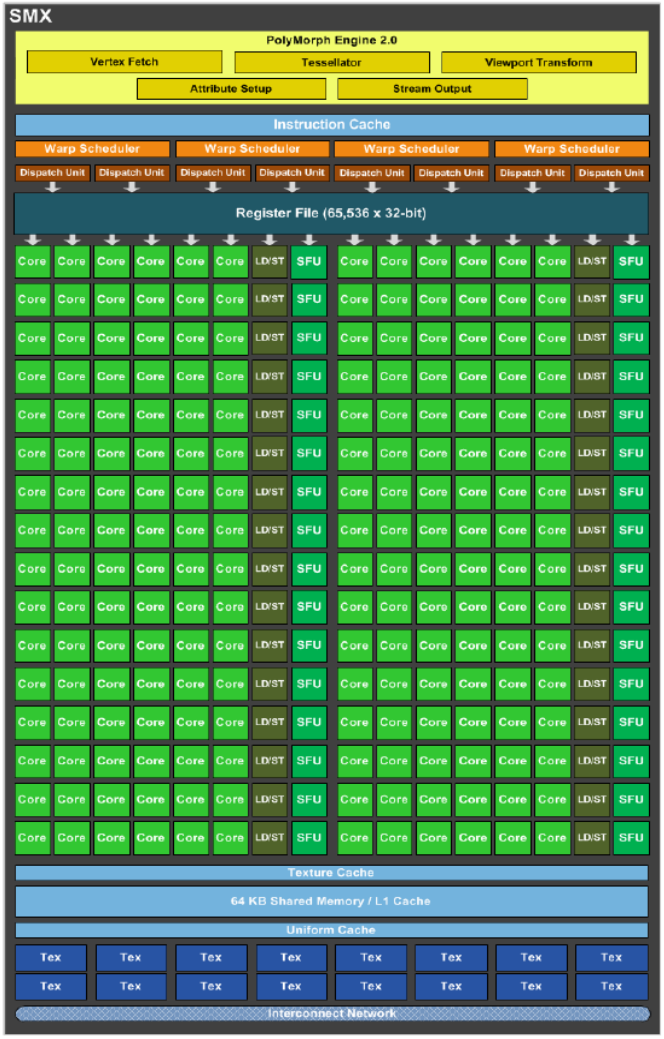
NVIDIA GPU Architectures

- Kepler
  - Like Fermi, Kepler GPUs are composed of different configurations of Graphics Processing Clusters (GPCs), Streaming Multiprocessors (SMs) and memory controllers
  - The GeForce GTX 680 GPU consists of four GPCs, eight next-generation Streaming Multiprocessors (SMX) and four memory controllers
  - Key features of the architecture are
    - The new SMX processor architecture
    - An enhanced memory subsystem, offering additional caching capabilities, more bandwidth at each level of the hierarchy, and a redesigned and faster DRAM I/O implementation
    - Hardware support to enable new programming model capabilities

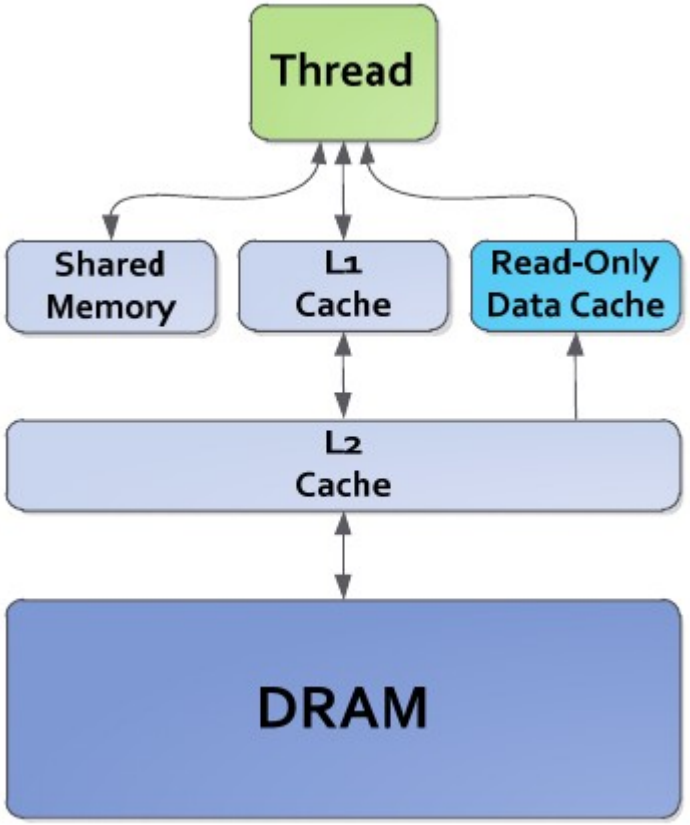
Kepler Architecture (GeForce GTX 680)



Kepler Architecture (GeForce GTX 680)



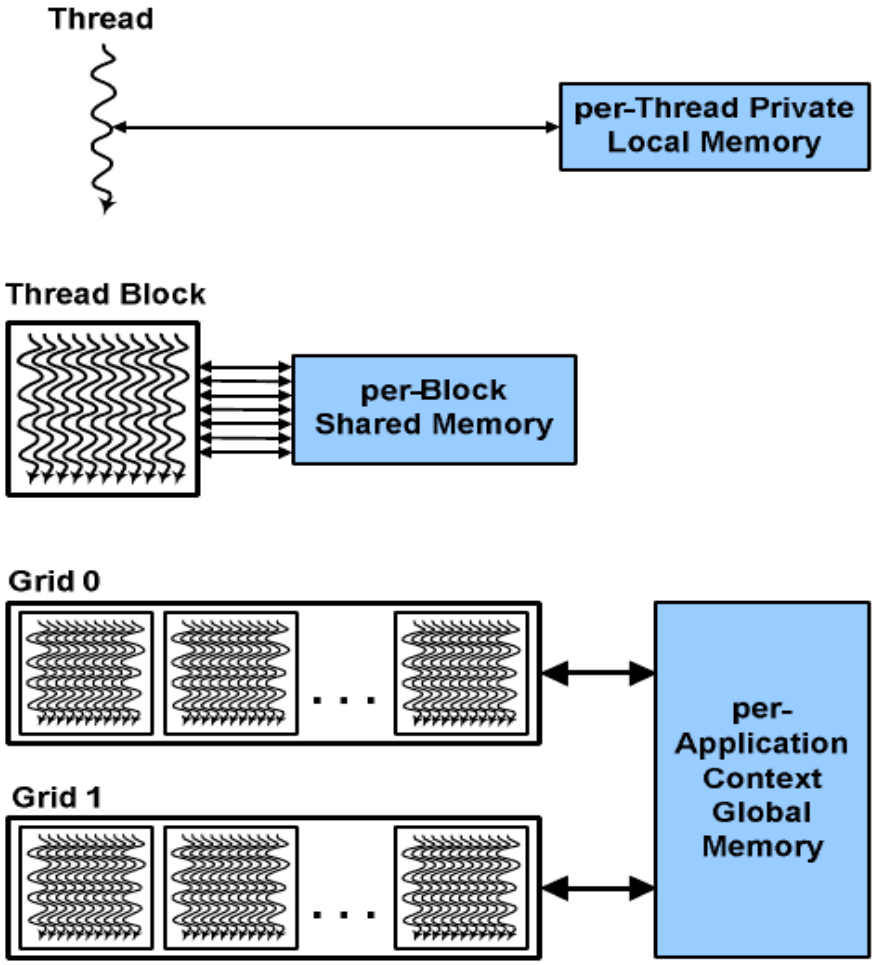
Kepler Memory Hierarchy (GK110)



CUDA & OpenCL

- CUDA
  - Is the hardware and software architecture that enables NVIDIA GPUs to execute programs written in C, C++, Fortran, OpenCL, DirectCompute and other languages (Compute Unified Device Architecture)
  - A CUDA program calls parallel kernels
  - A kernel executes in parallel across a set of parallel threads
  - The programmer or compiler organizes threads in thread blocks and grids of thread blocks
  - Each thread within a thread block executes an instance of the kernel and has a thread ID
  - A thread block is a set of threads that can cooperate through barrier synchronization and shared memory, and has a block ID within its grid
  - A grid is an array of thread blocks that execute the same kernel, read inputs from global memory, write outputs to global memory, and synchronize between dependent kernel calls

CUDA Thread Hierarchy and Memory Spaces

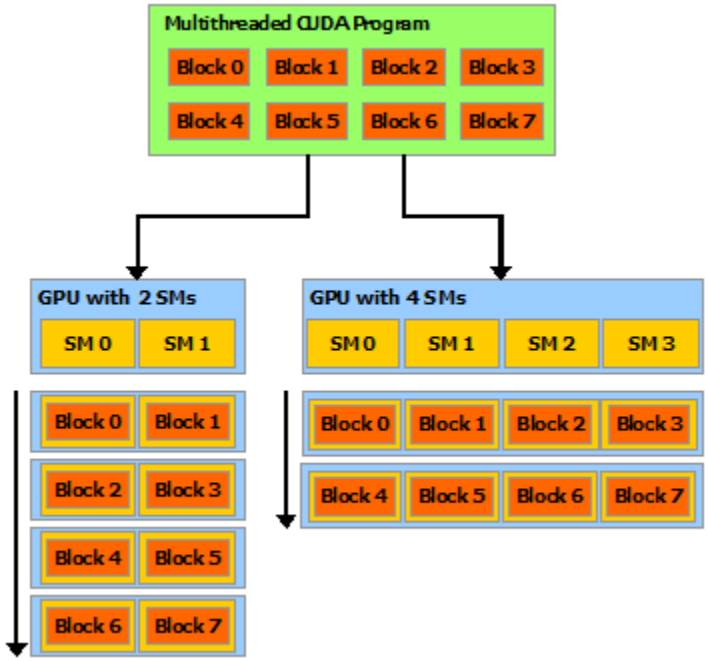


Hardware Execution

- CUDA's hierarchy of threads maps to a hierarchy of processors on the GPU
  - A GPU executes one or more kernel grids
  - A streaming multiprocessor (SM) executes one or more thread blocks
  - CUDA cores and other execution units in the SM execute threads
  - The SM executes threads in blocks of 32 threads called a warp
  - While programmers can generally ignore warp execution for functional correctness, they can greatly improve performance by having threads in a warp execute the same code path and access memory in nearby addresses

Automatic Scalability

- Blocks of threads execute independently from each other, so that a GPU with more multiprocessors will automatically execute the program in less time than a GPU with fewer multiprocessors
- Only the runtime system needs to know the physical processor count



OpenCL

- The CUDA architecture is a close match to the OpenCL architecture
  - A CUDA Streaming Multiprocessor corresponds to an OpenCL compute unit
  - A multiprocessor executes a *thread* for each OpenCL work item and a *thread block* for each OpenCL work group
  - A kernel is executed over an OpenCL NDRange by a *grid of thread blocks*
  - Each of the thread blocks that execute a kernel is uniquely identified by its work group ID and each thread by its global ID or by a combination of its local ID and its work group ID

- No OpenCL support in CUDA 5, must rely on CUDA 4.2 for now

- NVIDIA is developing OpenACC with Cray, PGI and others

Parallel Implementation of LDPC Codes

- Implementation of Decoders for LDPC Block Codes and LDPC Convolutional Codes Based on GPUs, *Yue Zhao and Francis C.M. Lau,* July 2012
  - Up to 100 to 200 times speedup on NVIDIA GTX 460 with 336 cores
  - http://arxiv.org/abs/1204.0334

- High-Throughput GPU-Based LDPC Decoding, *Yang-Lang Chang, Cheng-Chun Chang, Min-Yu Huang and Bormin Huang*, Proc. Of SPIE Vol. 7810, 781008 (2010)
  - Regular LDPC codes
  - Achieved 271 times speedup on NVIDIA Tesla 1060 with 240 cores

Parallel Implementation of LDPC Codes

- A Massively Parallel Implementation of QC-LDPC Decoder on GPU, *Guohui Wang, Michael Wu, Yang Sun, and Joseph R. Cavallaro*, 2011 IEEE 9th Symposium on Application Specific Processors (SASP)
  - http://gpuscience.com/cs/a-massively-parallel-implementation-of-low-density-parity-check-decoder-on-gpu/
  - Quasi-Cyclic LDPC (QC-LDPC)
  - LDPC decoder for IEEE 802.11n WiFi and 802.16e WiMAX LDPC codes as examples, irregular codes
  - Achieve throughput of up to 100.3 Mbps on an NVIDIA GTX 470 with 448 cores

Details of IEEE 9th SASP Paper

- Mapping LDPC Decoding Algorithm to GPU Kernels
  - Decoding can be split into two stages: horizontal processing & APP update (a posteriori probability)
  - One computational kernel for each stage, running on the GPU, and host code performs initialization and memory copy between host and device
  - CUDA Kernel 1: Horizontal Processing
    - Since the Check-node to Variable-node (CTV) messages are calculated independently, many parallel threads can be used to process them
    - For an M x N parity-check matrix **H**, M threads are spawned, and each thread processes a row
    - **H** consists of M_{sub} x N_{sub} sub-matrices, and is generated by the expansion of a Z x Z base matrix
    - M_{sub} thread blocks are used, each with Z threads
    - E.g. in 802.11g: 12 thread blocks each with 81 threads, 972 total

Details of IEEE 9th SASP Paper

- Mapping LDPC Decoding Algorithm to GPU Kernels
  - CUDA Kernel 2: APP value update
    - There are N values to be updated, and the APP update is independent among the variable nodes
    - N_{sub} thread blocks are used, each with Z threads
    - Kernel 2 finally makes a hard decision for each bit, by quantizing the APP value into 1 and 0, to get the decoded bit
- Multi-codeword Parallel Decoding
  - Since the number of threads and thread blocks are limited by the dimension of the **H** matrix, multi-codeword decoding is needed to further increase the parallelism of the workload
  - A two-level multi-codeword scheme is used
  - N_{cw} codewords are first packed into one macro-codeword (MCW)
  - Each MCW is decoded by a thread block and N_{mcw} MCWs are decoded by a group of thread blocks

Conclusions and Future Work

- LDPC codes can be efficiently implemented on NVIDIA GPUs

- Next steps:
  - Analyse the code from Madrid
  - Compare it with published work
  - Implement new code

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

AIT Austrian Institute of Technology

your ingenious partner

Ian Glendinning
ian.glendinning.fl@ait.ac.at