

January 01, 2007

Fast and Parallel Implementation of Image Processing Algorithm Using CUDA Technology on GPU Hardware

Neha Patil

Rensselaer Polytechnic Institute

Badrinath Roysam

Rensselaer Polytechnic Institute

Recommended Citation

Patil, Neha and Roysam, Badrinath, "Fast and Parallel Implementation of Image Processing Algorithm Using CUDA Technology on GPU Hardware" (2007). *Research Thrust R3 Presentations*. Paper 6. <http://hdl.handle.net/2047/d10009103>

This work is available open access, hosted by Northeastern University.



Fast and parallel implementation of Image Processing Algorithm using CUDA Technology On GPU Hardware

Neha Patel Badrinath Roysam

Department of Electrical & Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590

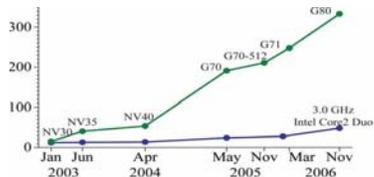


ABSTRACT

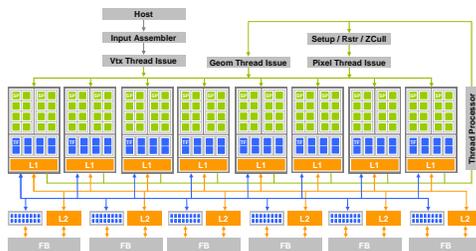
CUDA ("Compute Unified Device Architecture") is high level language for GPU programming. GPU Computing with CUDA on the GeForce 8 series is a new approach to computing where hundreds of on-chip processors simultaneously communicate and cooperate to solve complex computing problems up to 100 times faster than traditional approaches. A CUDA-enabled GPU operates as either a flexible thread processor, where thousands of computing programs called threads work together to solve complex problems, or as a streaming processor in specific applications such as imaging where threads do not communicate. CUDA-enabled applications use the GPU for fine grained data-intensive processing, and the multi-core CPUs for complicated coarse grained tasks such as control and data management. We use it here for Image processing algorithms like smoothing to achieve a faster implementation of it. It is well suited to address problems that can be expressed as data-parallel computations – the same program is executed on many data elements in parallel.

GPU AS DATA-PARALLEL COMPUTING DEVICE

GPU devotes more transistors to data processing. Same program is executed on many data elements in parallel.



G80 THREAD COMPUTING PIPELINE

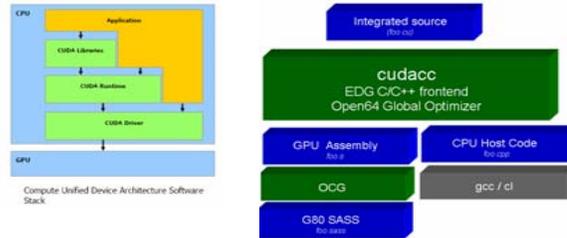


TECHNICAL SPECIFICATIONS

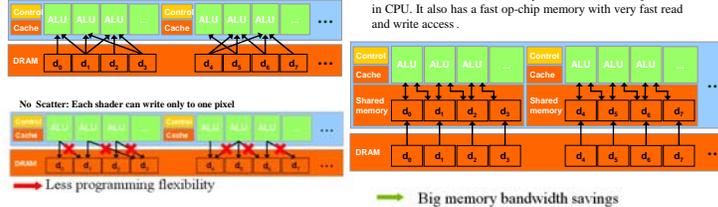
- Maximum number of threads per block: **512**
- Maximum size of each dimension of a grid: **65,535**
- Number of streaming multiprocessors (SM): **16 @ 675 MHz**
- Device memory: **768 MB**
- Shared memory per multiprocessor: **16KB** divided in **16 banks**
- Constant memory: **64 KB**
- Warp size: **32 threads** (16 Warps/Block)

CUDA on G80

CUDA stands for **Compute Unified Device Architecture** and is a new hardware and software architecture for issuing and managing computations on the GPU as a data-parallel computing device without the need of mapping them to a graphics API.

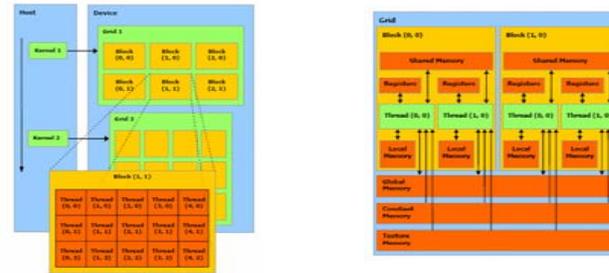


Only gather: Can read data from other pixels



It can read and write data at any location in DRAM just like in CPU. It also has a fast on-chip memory with very fast read and write access.

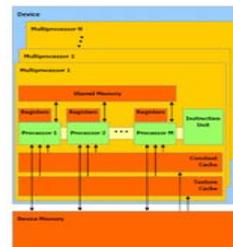
PROGRAMMING MODEL



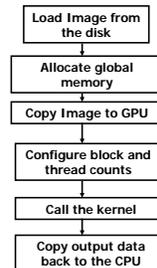
The host issues a succession of kernel invocations to the device. Each kernel is executed as a batch of threads organized as a grid of thread blocks.

A thread has access to the device's DRAM and on-chip memory through a set of memory spaces of various scopes

HARDWARE MODEL



A set of SIMD multiprocessor with on-chip shared memory



Flowchart for a CUDA program.

"This work was supported in part by Gordon-CenSSIS, the Bernard M. Gordon Center for Subsurface Sensing and Imaging Systems, under the Engineering Research Centers Program of the National Science Foundation (Award Number EEC-9986821)."

EXPERIMENTAL RESULTS

To measure the performance of CUDA, a code for mean filtering was written in CUDA and C++ and the execution time in both cases was found out.

Mean Filtering : Center pixel of a block is average of the neighborhood pixels.

Kernel

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Input Image and the mean filtered image.

Execution Time

Time taken by C++ code	123 ms
Time taken by CUDA code with global memory	3ms
Time taken by CUDA code with shared memory	1ms

Block Size	Time Taken
2	3.48 ms
4	1.837 ms
8	1.33 ms
16	1.25 ms
30	.74 ms

CONCLUSION

The execution time using CUDA is almost 100 times faster. The capability to achieve faster speed depends upon parallelism in the program and proper parameters like block size. Currently only one kernel can run at a time on the card, efforts are being made to run more than one kernel simultaneously to achieve more parallelism. Supports only single precision (32 bits).

REFERENCES

- [1] NVIDIA CUDA website : <http://developer.nvidia.com/object/cuda.html>

CONTACT INFORMATION

Badrinath Roysam , Professor
 Dept. of Electrical, Computer, and Systems Engineering
 Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180
 Phone: (518)276-8067; Fax: 518-276-8715;
 Email: roysam@ecse.rpi.edu