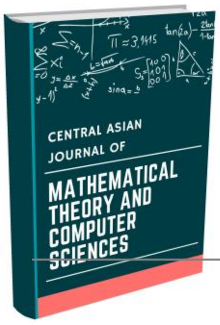

CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES
https://cajmtcs.centralasianstudies.org

Volume: 04 Issue: 1 | Jan 2023 ISSN: 2660-5309

ANALYSIS OF THE DEVICE FEATURES OF GENERAL-PURPOSE PROGRAMMABLE GRAPHICS PROCESSORS

Rakhimov Bakhtiyar Saidovich
Head of the Department of Biophysics and information technologies of Urgench branch of Tashkent Medical Academy, Uzbekistan
bahtiyar1975@mail.ru

Yakubov Durumboy Musaevich
Assistant - professor, Urgench branch of Tashkent Medical Academy, Uzbekistan
durumboymusaevich@mail.ru

Saidov Atabek Bakhtiyarovich, Saidova Zarina Bakhtiyar Qizi
Students Urgench branch of Tashkent University of Information Technologies named after Muhammad al Khwarizmi, Uzbekistan
ootabek2001@mail.ru, zarina2003@mail.ru

Abstract

The main function of graphics processors since their inception has been graphics processing. Subsequently, after it became possible to program the processing of model vertices and pixels of rendered three-dimensional scenes using special programs (shaders), the architecture of graphic processors changed significantly. After the advent of the first general-purpose programmable graphics processors (G80 architecture from NVIDIA, R600 architecture from ATI), it became possible to process commands not only for graphic data in vector form, but also to perform ordinary calculations for arbitrary data on a variety of special cores, while implementing data parallelism.

ARTICLEINFO

***Article history**:*
Received 6 Nov 2022
Revised form 5 Dec 2022
Accepted 30 Jan 2023

Keywords: *Grafic processors, algorithm, memory, hardware, software.*

--***--

Introduction.

Data parallelism is an approach to the parallelization of calculations, in which one operation is performed immediately on all elements of the data array.

Therefore, GPUs show high efficiency rates when parallelizing programs that process a lot of data of the same type. Such programs include shaders [7]: a vertex shader processes 3D vertices with different parameters, a pixel shader processes 2D pixels on the screen using interpolated data. Even before the advent of general-purpose kernels, there were attempts to simulate the processing of arbitrary data of the same type written into textures (image matrices) using pixel shaders [3], which, of course, gave a performance boost compared to the CPU.

To develop an abstract model of the GPU, consider the general structure of modern GPUs, as well as models designed for their programming.

For the first time, the G80 architecture was introduced in November 2006 [89] and became widespread in several variants of graphic processors at once, differing in the number of multiprocessors installed on them [10]. If we remove all architectural elements associated with graphics processing, we get the following block diagram, shown in Fig. 1.3.

The following main elements stand out in the G80 architecture:

1) A flow control block designed to generate a schedule and control the execution of flows. This block is controlled by the main system processor (CPU), which delegates a parallel task consisting of many threads to the GPU;

2) Computing unit consisting of a set of streaming multiprocessors that accept and process parallel streams (working in MIMD mode);

3) Hierarchy of memory, in which the main element is video memory (graphic adapter memory). It is accessed through caching at the L1 and L2 levels. However, a memory access operation is very expensive and can cost anywhere from 400 to 600 multiprocessor cycles.

Objective Statement

Obviously, not all computer vision algorithms can be parallelized on GPUs. Any artificial computer vision system, regardless of its area of application, should include the following typical stages of work:

1) image acquisition (photo or video filming);

2) preliminary processing;

3) highlighting characteristic features;

4) detection or segmentation;

5) high-level processing.

Materials nad Methods.

All scalar processors operate in SIMD mode, while executing a block of threads (in the G80, the number of threads in a block is 32), called a warp (bundle). In this case, in 4 cycles of the multiprocessor, all streams of the bundle are processed at once when executing floating-point operations, with double precision - in 32 cycles, and transcendental functions - in 16 cycles. The number of threads per multiprocessor is limited. To synchronize threads, special instructions have been developed that interrupt the execution of a bundle and start the next bundles in the queue until all bundles are interrupted. Due to this mechanism, threshold synchronization is achieved with minimal time. Usually it is intended for communication of scalar processors through shared memory.

A multi-stage memory hierarchy allows access to global data through caches of the first and second levels. The second cache level is shared between data and texture units, while the first level is shared between the two multiprocessors and is for data only. When computing on a multiprocessor, in addition to data from global memory, two additional types of memory are used: constant memory and shared memory. Constant memory is read-only. It is stored in video memory and cached on the multiprocessor at 8 KB each (the total amount for all multiprocessors is limited and is 64 KB; the memory of constants is the same for all multiprocessors). The delay in accessing it can be the same time as for accessing global memory with a cache miss, but if there is a hit in the cache, then the access will be carried out in 2 multiprocessor cycles [4,5]. Shared memory is intended for reading/writing and is organized in the form of memory banks (16 or 32 each), each of which can be accessed in 2 cycles. The request for access to shared memory receives the entire bundle. In this case, there may be requests to one bank of shared memory at once, which will lead to a conflict and ordering of requests in a queue with an increase in the access time to bank data for the entire bundle. Therefore, it is important to take into account the coherence of queries when creating algorithms [1]. The amount of shared memory is also limited (16 KB), but in later versions of GPUs (Compute Compatibility 2.x) it can be adjusted depending on the task. In the process of studying various types of access to memory caches, it was revealed that

Results and Discussion

The next architectural solution for NVIDIA GPUs was the GT200, introduced in June 2008 [9]. The main differences from the G80 in terms of general calculations are:

1) the number of TPCs (Thread Processing Cluster, multiprocessor clusters) has increased;
2) in each TPC, the number of multiprocessors has increased to 3 per cluster, while the L1 cache has also increased;
3) the number of registers for program instructions has doubled;
4) each multiprocessor has a block for calculating operations on double-precision floating-point numbers.

Thus, this architecture made a breakthrough in the number of scalar processors on the graphics adapter and the amount of data processed, but the structural elements and their location remained the same.

In contrast to the GT200 architecture, the Fermi [1] third-generation architecture introduced in September 2009

Conclusions

The central element of the architecture is the second-level cache, which accesses the video memory. Accordingly, its volume has increased significantly. Streaming multiprocessors are formed around this cache, which have also changed from previous architectures:

1) the number of scalar processors has increased to 32, moreover, they are optimized for working on 64-bit data;
2) it became possible to execute two competing bundles of threads on one multiprocessor;
3) the number of blocks for calculating transcendental functions has increased to 4;
4) it became possible to control the amount of shared memory and the first-level cache for data;
5) blocks appeared for calculating data addresses located in a single address space.

NVIDIA GPUs are programmed using the Compute Unified Device Architecture (CUDA) heterogeneous model [3, 12, 13]. CUDA includes a programming model for parallel computing on superscalar architectures, as well as libraries and extensions with a compiler for several high-level languages.

References

1. Rakhimov BS, Mekhmanov MS, Bekchanov BG. Parallel algorithms for the creation of medical database. J Phys Conf Ser. 2021;1889(2):022090. doi:10.1088/1742-6596/1889/2/022090
2. Rakhimov BS, Rakhimova FB, Sobirova SK. Modeling database management systems in medicine. J Phys Conf Ser. 2021;1889(2):022028. doi:10.1088/1742-6596/1889/2/022028
3. Rakhimov B, Ismoilov O. Management systems for modeling medical database. In: ; 2022:060031. doi:10.1063/5.0089711
4. Rakhimov BS, Khalikova GT, Allaberganov OR, Saidov AB. Overview of graphic processor architectures in data base problems. In: ; 2022:020041. doi:10.1063/5.0092848
5. P. P. Kudryashov Algorithms for detecting a human face for solving applied problems of image analysis and processing: author. dis. Cand. tech. Sciences: 05.13.01. - M, 2007.

6. Tanenbaum E. Modern operating systems. 2nd ed. - SPb .: Peter, 2002 .-- 1040 p .: ill.
7. Forsyth DA, Pons, J. Computer vision. Modern approach / D.A. Forsyth, J. Pons: Trans. from English - M .: Publishing house "Williams", 2004. - 928 p .: ill. - Parallel. tit. English
8. Allaberganov, O.R., Rakhimov, B.S., Sobirova, S.K., Rakhimova, F.B., Saidov, A.B. Problem for Medical System with Infinite Zone Potential in the Half Line AIP Conference Proceedings, 2022, 2647, 050025
9. RB Saidovich, SA Bakhtiyarovich, BB Farkhodovich, KDA Ugli, MMZ Qizi Analysis And Using of the Features Graphics Processors for Medical Problems Texas Journal of Medical Science 7, 105-110
10. Frolov V. Solution of systems of linear algebraic equations by the preconditioning method on graphic processor devices
11. Brodtkorb A.R., Dyken C., Hagen T.R., Hjelmervik J.M., Storaasli O.O. State-of-the-art in heterogeneous computing / A.R. Brodtkorb, C. Dyken, T.R. Hagen, J.M. Hjelmervik, O.O. Storaasli // Scientific Programming, T. 18, 2010. - S. 1-33.
12. Zaynidinov H., Mallayev O., Kuchkarov M. Parallel algorithm for modeling temperature fields using the splines method 2021 IEEE International IOT, Electronics and Mechatronics Conference, IEMTRONICS 2021 - Proceedings, 2021, 9422645
13. Zaynidinov H., Makhmudjanov S., Rajabov F., Singh D. IoT-Enabled Mobile Device for Electrogastrography Signal Processing Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021, 12616 LNCS, стр. 346–356
14. Zaynidinov H.N., Yusupov I., Juraev J.U., Singh D. Digital Processing of Blood Image by Applying Two-Dimensional Haar Wavelets Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in bioinformatics), 2021, 12615 LNCS, стр. 83–94