STUDY MODULE DESCRIPTION FORM

Name of the module/subject	Code
Advanced applications of graphics cards	**1010512331010519522**

Field of study	Profile of study (general academic, practical)	Year /Semester
Computing	**general academic**	**2 / 3**

Elective path/specialty	Subject offered in:	Course (compulsory, elective)
Distributed Systems	**Polish**	**elective**

Cycle of study:	Form of study (full-time,part-time)
Second-cycle studies	**full-time**

No. of hours				No. of credits
Lecture: **30**	Classes: **-**	Laboratory: **30**	Project/seminars: **-**	**4**

Status of the course in the study program (Basic, major, other)	(university-wide, from another field)
major	**from field**

Education areas and fields of science and art	ECTS distribution (number and %)
technical sciences	**4 100%**

Responsible for subject / lecturer:

dr inż. Witold Andrzejewski
email: Witold.Andrzejewski@cs.put.poznan.pl
tel. 61 6652965
Instytut Informatyki
ul. Piotrowo 2, 60-965 Poznań

Prerequisites in terms of knowledge, skills and social competencies:

1	**Knowledge**	Student starting this module should have basic knowledge regarding computer hardware architecture, procedural and object programming as well as algorithmic complexity. Additionally, he should have basic knowledge of computational geometry.
2	**Skills**	The student should have the skills to solve simple algorithmic problems, program in C/C++ and to acquire knowledge from the designated sources of information.
3	**Social competencies**	The student should also understand the need to extend his/her competences and be ready to work in a team. Moreover, the student should show such attitudes as honesty, responsibility, perseverance, curiosity, creativity, manners, and respect for other people.

Assumptions and objectives of the course:

1. Provide students with the knowledge regarding programming of graphics processing units (GPUs) including:

a. Selected aspects of hardware architecture of popular graphics cards.

b. Logical concurrency model in modern graphics cards in the context of CUDA and OpenCL APIs.

c. GPU program optimization.

d. Selected basic parallel primitives such as: map reduce, compact, sort, scan, search, combination/domain/permutation sequences.

e. selected data structurs used in parallel algorithms, including: hash tables, matrices, CSS-trees and the corresponding algorithms

f. Complexity analysis of parallel algorithms (PRAM model) and its relation to actual complexity of program utilizing GPUs

g. Applying graphics cards to solving practical problems related to data processing and visualization.

2. Develop students' skills in

a. Solving algorithmical problems with a focus on parallelization of data processing

b. GPU program execution optimization.

Study outcomes and reference to the educational results for a field of study

Knowledge:

1. has advanced and in-depth knowledge of tools and programming environments useful in programming graphics processing units - [K2st_W1]

2. has advanced and detailed knowledge of GPU-based parallel algorithms and methods for designing them - [K2st_W3]

3. has advanced and detailed knowledge of graphics card processing units architectures - [K2st_W5]

4. knows advanced methods, techniques and tools used for solving complex engineering tasks related to designing, and low-level optimization, of software which utilizes GPUs - [K2st_W6]

Skills:

1. is able to use communication and information techniques used during implementation of computer systemsis able to use communication and information techniques used during implementation of IT projects - [K2st_U2]

2. is able to integrate knowledge from multiple different areas of computer science and utilize a systematic approach while formulating and solving engineering tasks - [K2st_U5]

3. is able to assess usefulness and applicability of new achievements (methods and tools) as well as new IT products - [K2st_U6]

4. is able to assess usefulness and applicability of methods and tools (including their limits) dedicated to solving an engineering task, which consists in designing (or analyzing) of an information system or its components which utilize GPU acceleration - [K2st_U9]

5. is able to design and implement a GPU accelerated algorithm according to designated specification including non-technical requirements - [K2st_U11]

Social competencies:

1. understands that in computer science knowledge and skills can become obsolete very quickly - [K2st_K1]

2. knows examples (and understands causes) of faulty computer systems which caused substantial financial lossesunderstands the importance of utilizing the newest achievements in the field of computer science in solving research and practical problems. - [K2st_K2]

Assessment methods of study outcomes

Formative assessment:

a) lectures:

 * based on answers to questions related to subjects covered during previous lectures,

b) laboratory classes:

 * evaluation of doing correctly assigned tasks (following provided lab. instructions),

 * occasional evaluation of students preparation for classes (entry tests)

Total assessment:

a) verification of assumed learning objectives related to lectures:

 * evaluation of acquired knowledge on the basis of the written exam (a test, ~30 questions, total points achievable 30, 16 points needed to pass).

b) verification of assumed learning objectives related to laboratory classes:

 * based on the project implemented by a team of students, each students' grade is evaluated based on the quality of his/hers part as well as answering to several project related questions.

Additional elements cover:

* ability to utilize previously gained knowledge

* discussing more general and related aspects of the class topic,

* showing how to improve the instructions and teaching materials.

Course description

The lecture covers the following topics:

1. Motivation behind utilization of computer graphics cards for general computations. Introduction of several technological and architectural solutions which allow for parallel processing. Definition of terms used throughout the rest of the lectures. Short description of programming model used by programs for GPUs.

2. Detailed description of graphics cards' architecture. Relationships between graphics cards' architecture and programming model. Bit tricks.

3. Description of memory hierarchy and efficient memory access patterns for several different architectures. Exemplary optimization methods utilizing several levels of memory hierarchy. Description of methods for parallel data transfer and processing.

4. Thread communication and synchronization methods.

5. Theoretical basis for parallel algorithm complexity assessment. PRAM machine model. Introduction of several basic parallel primitives including: map, gather, scatter, reduce, compact, search, scan. Detailed description of scan algorithm. Complexity analysis of parallel and sequential versions of this algorithm.

6. Continuation of parallel primitive subject. Detailed description of compact, reduce, search and sort primitives. Complexity assessment of presented algorithms.

7. Algorithms for parallel generation of combination, domain and permutation sequences. Complexity analysis.

8. Horizontal and vertical joins based on merge-path algorithm.

9. Introduction to efficient parallel data structures. Efficient matrix processing (sparse and dense), hash tables, CSS trees and graph processing algorithms. Complexity analysis.

10. GPU applications for data analysis, exploration and visualization.

The laboratory lessons cover the following topics:

1. Introduction to CUDA API.

2. Exercises related to proper construction of computation grid.
3. Implementation of simple parallel algorithms.
4. Debugging of GPU code.
5. Complex exercises which require thread communication synchronization
6. Testing the performance of several different memory types.
7. Introduction to thrust library and its API. Simple exercises which utilize the library.
8. Introduction to CURAND library. Exercises utilizing thrust library. Implementing complex algorithms via parallel primitives implemented in thrust.
9. Introduction to OpenCL + simple exercises
10. Introduction to NVIDIA Optix API and its applications in data visualization.

Teaching methods:

1. Lectures: multimedia presentation, presentation illustrated with examples presented on black board, solving tasks
2. Labs: solving tasks, practical exercises, discussion, teamwork, multimedia showcase

Basic bibliography:

1. Programming Massively Parallel Processors - A hands-on Approach/ David B. Kirk, Wen-mei W. Hwu
2. OpenCL Programming Guide / Aaftab Munshi, Benedict R. Gaster, Timothy G. Mattson, James Fung, Dan Ginsburg
3. NVIDIA: CUDA C Programming Guide: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html
4. NVIDIA: CUDA C Best Practices Guide: http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html
5. NVIDIA: OpenCL Jumpstart Guide: http://www.cs.cmu.edu/afs/cs/academic/class/15668-s11/www/cuda-doc/OpenCL_Jumpstart_Guide.pdf
6. NVIDIA: OpenCL Best Practices Guide: http://www.nvidia.com/content/cudazone/CUDABrowser/downloads/papers/NVIDIA_OpenCL_BestPracticesGuide.pdf
7. NVIDIA Optix API documentation http://raytracing-docs.nvidia.com/optix/index.html

Additional bibliography:

1. Jianlong Zhong* and Bingsheng He. Medusa: Simplified Graph Processing on GPUs. Accepted by TPDS 2013: IEEE Transactions on Parallel and Distributed System
2. Bingsheng He and Jeffrey Xu Yu. High-Throughput Transaction Executions on Graphics Processors. Proceedings of Very Large Data Bases (VLDB) 2011
3. Andrzejewski, Witold; Boinski, Pawel Efficient spatial co-location pattern mining on multiple GPUs Journal Article Expert Systems with Applications, 93 (Supplement C), pp. 465?483, 2018, ISBN: 0957-4174.
4. Andrzejewski, Witold; Boinski, Pawel Parallel GPU-based Plane-sweep Algorithm for Construction of iCPI-trees Journal Article Journal of Database Management, 26 (3), pp. 1-20, 2015, ISSN: 1063-8016.
5. Andrzejewski, Witold; Boinski, Pawel Parallel approach to incremental co-location pattern mining, Information Sciences, accepted for publication
6. CUDA Application Design and Development / Rob Farber
7. CUDA By Example / Jason Sanders/Edward Kandrot

Result of average student's workload

Activity	Time (working hours)
1. participating in laboratory classes / tutorials:	16
2. preparing to laboratory classes:	16
3. consulting issues related to the subject of the course; especially related to laboratory classes and projects,	4
4. preparing to tests	16
5. participating in lectures	16
6. studying literature / learning aids	16
7. preparing to and participation in the final test	16

Student's workload

Source of workload	hours	ECTS
Total workload	100	4
Contact hours	62	2
Practical activities	45	2