Годишник Телекомуникации 2015, том 2, с. 9 - 17

Yearbook Telecommunications 2015, vol. 2, p. 9 - 17

eISSN 2534-854X https://telecommunications.nbu.bg/bg/godishnik-telekomunikacii

ВЪВЕДЕНИЕ В CUDA

Георги Петров

OVERVIEW OF CUDA

Georgi Petrov

Резюме: Настоящата публикация представлява обзор на технологията CUDA, и по-точно как CUDA разпределя ресурсите на паметта на видеокартата, как CUDA разпределя една изчислителна задача на изчислителни клъстери и отделни изчислителни процеси. Ще добиете обща представа за възможностите на технологията и възможното и развитие в близко бъдеще.
Ключови думи: SIMD, SIMT, CUDA, MISD, SISD, MIMD.

Abstract: This publication is an overview of CUDA technology, and more specifically how CUDA allocates memory card memory resources, how CUDA allocates a computing task to computing clusters and separate computing processes. The work presents a general idea of the possibilities of the technology and its possible development in the near future.
Keywords: SIMD, SIMT, CUDA, MISD, SISD, MIMD.

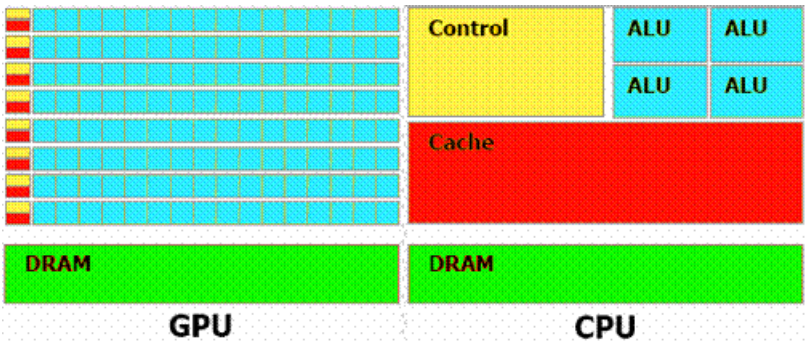
1. Обзор на хардуерната архитектура на CUDA

Технологията CUDA се счита за основоположник на либерализацията на “супер компютинга”, често е наричана демократична технология, именно защото дава равен достъп на всички инженери и учени до равноправен евтин и качествен изчислителен ресурс. Чрез CUDA съвместимите продукти всеки програмист или учен има достъп до огромен изчислителен ресурс в рамките на стандартните бюджети за закупуване на съвременно PC, като цените на полупрофесионалните продукти започват от 250$ до 1300$ за обикновенни офис платформи [1]. Не на последно място, CUDA показва колко е полезно да играете 3D компютърни игри и да правите всички онези безсмислени на пръв поглед неща, които рано или късно водят до нови технологии и нови приложения! Приложения и алгоритми чиято, проверка изискваше десетки дни, днес могат да бъдат изпълнени за часове или дори минути в зависимост от ползвания хардуер. Нещо повече, машини като Cray и IBM Blue Gene консумират много повече електроенергия и имат много по-високи текущи разходи по поддръжката отколкото един CUDA Tesla суперкомпютър. Подобна персонална супермашина може да поставите на вашето офис бюро! Освен това не бива да пренебрегвате и чувството, което досегът с тази супер технология ще ви донесе, ежедневно, помагайки ви да решите бързо, качествено и прецизно вашите инженерни изчислителни проблеми. В краен случай ако не можете да програмирате върху тази супер технология винаги може да я използвате за игра на любимите си 3D игри! Нещо което не може да направите с един университетски клъстер. Бързодействието на програмите, писани за CUDA съвместими устройства зависи основно от ползвания хардуер, като с малко трикове програмите ви ще могат да работят на всички поддържащи технологията устройства. Принципно архитектурата на GPU се различава от тази на CPU (Фиг. 4) [2]. Едновременно с това, когато през 2009г. започнах да използвам тази технология повечето програмисти дори не бяха чували за нея и възможностите и. Масовото схващане беше и остава, че трябва да пишат код за многоядрени паралелни процесори или FPGA и DSP платформи. Никой от тези проекти обаче все още не постигна заслужено развитие, което е обосновано поради високите начални инвестиции

необходими за създаване на реконфигурируем супер компютър с FPGA и DSP. CUDA е друга бира! Нещо съвсем различно, дефакто тя се преконфигурира за секунди, достатъчно е да напишете нова програма и да я стартирате. При нея не е нежно да създавате нов хардуер, да създавате платки, да пишете на интересни екзотични езици, като VHDL, да компилирате тествате и конфигурирате електронните компоненти.
По информация на производителя съвременните GPU не са част от самия графичен процесор на видео картата [3]. И двата чипа могат да използват общата видео памет, която е многократно по-бърза от стандартната RAM инсталирана на дънните платки на персоналните компютри. При GPU технологията се заделя сравнително малка част от кеша, наричан споделена памет (_shared_), в която може да се извършват обработки от множество АЛУ (аритметично логически устройства) едновременно. Друга особеност е че се ползва мнококанална видео памет наричана глобална (_global_). Един алгоритъм се разбива на под процеси, обединени в блокове, а те от своя страна в клъстери от блокове. В процеса на работа всеки един процес се изпълнява псевдо паралелно, като при това в зависимост от броят мултипроцесори се постига по-бързо изпълнение на всички тези процеси във времето. Характерно за CUDA е ползването на Single-Instruction Multiple-Thread (SIMT) архитектура, като при това данните записани в различни области от паметта могат да бъдат обработвани едновременно от множество АЛУ в няколко мултипроцесора едновременно. При стандартните процесори AMD и Intel голяма част от транзисторите в чипа се ползват за изграждане на локални кеш памети и по-сложна контролна логика на контролера на процесора. При CUDA има повече АЛУ, което означава, че повече еднотипни изчисления ще бъдат извършени върху даден масив от данни за определено време. Разбира се различни техники за оптимизацията на този процес могат да се използват в процеса на програмирането на системата, за да се постигне максимално високо бързодействие на програмите. Тук ще се спра над типовете компютърни архитектури, най-общата представа за тях ще ви даде повече знания за това кога и за какво можете да ги ползвате. Масово използваните днес процесорни архитектури са Харвардска и Фон Нойман. Първият модел, разпределя отделни блокове памет за съхранение на инструкциите и данните, а вторият тип позволяват на паметта да съдържа едновременно програмни инструкции и данни (Фиг. 1). Освен това процесорите могат да се различават по начина по който обработват инструкциите и данните, като съществуват основно четири модела:

- **SIMD** (Single Instruction, Multiple Data),
- **SIMT** (Single Instruction Multiple Threads) CUDA
- **MISD** (Multiple Instruction, Single Data),
- **SISD** (Single Instruction, Single Data) и
- **MIMD** (Multiple Instruction stream, Multiple Data stream).

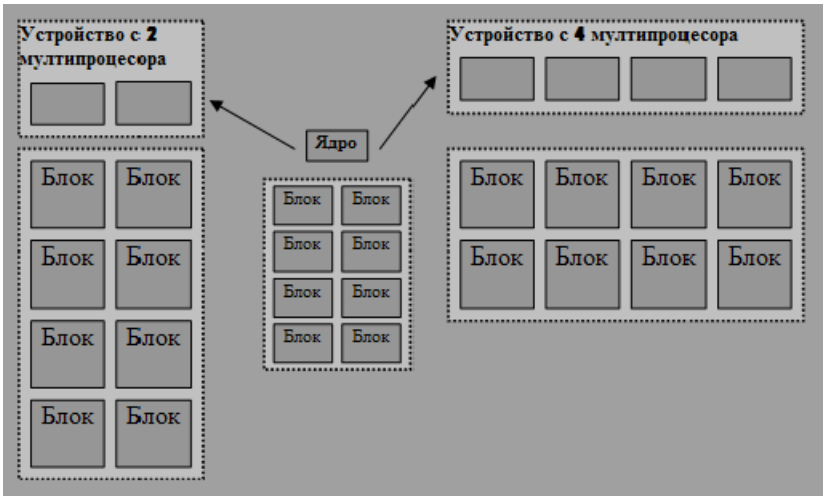
SIMD предполага използването на множество АЛУ работещи едновременно над съседни области памет с еднакви инструкции, каквито са съвременните DSP процесори. CUDA е модификация на SIMD наричана SIMT. В тази технология всеки процес обработва данните върху свои собствени регистри, докато при SIMD програмистът предварително трябва да пакетира данните.
MISD позволява обработката на една област памет едновременно от множество АЛУ. SISD се ползва при микроконтролери и микропроцесори от нисък клас, като при тях е налично едно АЛУ можещо да обработва една данна в даден момент от време, това са обикновено миниатюрни вградени приложения с ниска консумация на електроенергия.



Фиг. 1. Архитектурни особености на GPU и CPU, (изт. NVIDIA).

MIMD позволява мащабна асинхронна обработка на данни, при която всеки отделен процесор обработва данни намиращи се в различни области на паметта, каквито са съвременните много ядрените и многопроцесорни системи. Подобни системи са подходящи за разпределени офис и интернет приложения, уеб услуги, където всеки потребител работи над различно приложение, което обработва различни данни постъпващи в асинхронен режим. Спецификата на CUDA я доближава много до съвременните DSP процесори, като при това запазва гъвкавостта и възможностите за лесната промяна на програмната конфигурация характерна за съвременните персонални компютри. Характерна разлика обаче е ползването на паметта, при DSP за постигане на бърза сигнална обработка има специализирани преместващи регистри, които са скъпи и това ограничава практическото внедряване на подобни системи. Въпреки това в практиката при реализацията на комплексен изчислителен център често се ползват разнообразни сървърна технологии: CPU, DSP, FPGA и CUDA базирани. Тяхната цел е да позволят поточното въвеждане на данните, които ще се обработват и обработката им върху възможно най-подходящата архитектура. Това ги оскъпява, а програмирането им е трудно и изисква ползването на различни софтуерни инструменти. Но основната разлика с DSP, е че при тях не може да правите съществени изменения в програмния код. Съществуват процесори за линейна обработка на данни и такива за нелинейна, тоест един DSP чип има строго специфична област на приложение. При тях постигането на по-високо бързодействие изисква добавяне на още чипове. От друга страна драйверите на CUDA разпределят автоматично отделните изчисления над пълния набор налични мултипроцесори в системата, което дава възможност една част групирани в блокове задачи да бъде осъществима на карта с по-малък брой мултипроцесори или такава с по-голям брой без да се налага програмистът да мисли за разпределянето на ресурсите – фиг. 2. Това е непосилна задача за класическите C програми работещи върху DSP. Новите версии на драйверите позволяват едно масивно изчисление да се разпредели едновременно върху няколко графични карти, и при това без програмистът да трябва да учи да ползва драйверът на видеокартите. Това позволява скалируемост и мултиплатформеност на приложенията, например ако вашата програма работи върху мобилен телефон с CUDA съвместим хардуер тя ще консумира по-малко енергия за сметка на по-бавно изпълнение, а ако работи върху супер компютър ще се изпълнява по-бързо. И най-важното потребителят няма да е ангажиран с допълнителни действия освен да пусне своята програма. Друга особеност, която би ви накарала да изберете CUDA при реализация на комплексни програми за обработка на сигнали и изображения вместо DSP, е че вие можете да ползвате десетки готови математически функции от CUDA SDK, с които да извършвате 1D, 2D и дори 3D Фурие трансформации и матрични операции без да се налага да пишете специфичен софтуер оптимизиран за специфични DSP чипове. С две

думи, ако вашето приложение не е вградено, няма съществени изисквания към безотказността му, няма да се ползва в самолети, автомобили и военни съоръжения, непременно следва да изберете CUDA.



Фиг. 2. Изпълнение на една и съща изчислителна задача заемаща 8 блока върху различни архитектури имащи 2 и 4 мултипроцесора отнема различно време, но резултатът ще бъде еднакъв.

В най-общия случай за да извършим някакви изчисления паралелно ще се стартират n на брой процеса. Данните и съответно инструкциите от тези процеси се обособяват в т.н. блокове, които от своя страна се обособяват в т.н. рамки (клъстери). По този начин може да се оптимизират ресурсите на видео картата, като глобална памет и брой мултипроцесори. Това позволява на драйвера да ползва различни хардуерни архитектури, зареждайки в тях повече или по-малко на брой блокове, респективно сумарен броя едновременно изпълнявани процеси. Това става прозрачно за програмиста и позволява ползването на хардуер с различни възможности без да се налага повторно компилиране на софтуера. При повечето случаи с цел оптимизиране на броят обръщения към глобалната памет, а това са над 95% от реалните примери, се налага ползването на допълнителни механизми за синхронизация на процесите преди да се прехвърли управлението върху друга група процеси от следващ блок. Версиите на CUDA позволяват използването и на специфични операции, наричани атомарни. При тях например версия 1.1 на хардуера има вградени команди за извършване на изчисления над дадена клетка от глобалната памет. Версия 1.2 на хардуера позволява такива действие върху споделената памет във всеки един мултипроцесор. Тези действия разрешават постигането на високо бързодействие. Следва да се каже, че ползването на мултипроцесорите за обработка на елементи намиращи се в глобалната памет е многократно по-бавно, това налага ползването на споделена и константна памет в зависимост от ситуацията. Малки порции данни, респективно процеси се зареждат в споделената памет на всеки мултипроцесор. След като всички процеси в обработвания блок приключат системата приема за обработка друг блок. Данните от междинните резултати се съхраняват в споделената памет, като след приключване на всеки един блок те се записват в глобалната памет. Това позволява постигането на високо бързодействие, т.к. обръщенията към глобалната памет се намаляват. При това всеки мултипроцесор работи с локални копия на данните и микропрограмите. Архитектурата предоставя многоядрени процесори, обединени в т.н. мултипроцесорен блок, всеки от които има отделни АЛУ за изчисления на целочислени аргументи и отделно АЛУ за изчисления на числа с плаваща запетая. Не всички изчислителни версии на хардуера поддържат еднакъв тип глобални и локални атомарни

операции над различни типове данни. Начинаещите програмисти следва да обърнат внимание на следните основни термини и тяхното значение:

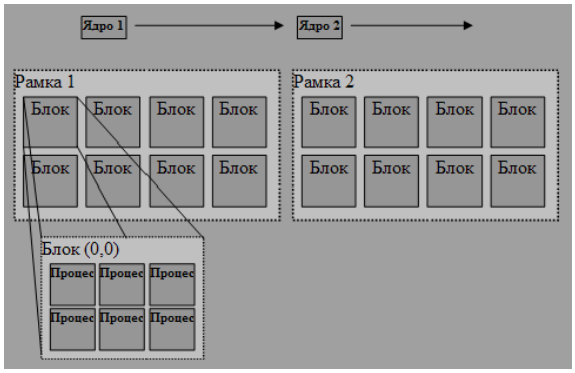
Half-Warp – това е група от 16 процеса чакащи в опашката за последователно изпълнение. Half-warp процесите се изпълняват едновременно. Например процесите от 0->15 ще се намират в един и същи под блок, 16->31 в друг и т.н., но е възможно по-нови версии на хардуера да поддържат друго деление.

Warp – това е група от 32 съседни процеса, те принципно се изпълняват в паралел. Това налага специфични методи за синхронизация, целящи изчакване на завършването на всички процеси на дадена итерация от изчисленията преди да бъде приет следващия блок за обработка.

Block – блокът е набор от процеси, които правят едно и също нещо над различни елементи от масив с данни или едни и същи споделени данни. Поради технически причини минимум 192 процеса са нужни в един блок за оптимизация на закъсненията между тях. Един типичен блок има 256 максимум 512 процеса. Блокът може да има 1D, 2D и 3D структура, като сумарния брой процеси не бива да надвишава 512. Следва да се има предвид, че процесите в един блок се синхронизират по-бързо и така по-лесно обменят данни помежду си чрез споделената памет. Следва да се има предвид, че 8 конкурентни блока могат да се изпълняват на един мултипроцесор.

Grid – “решетката или рамка” позволява да се създават макро блокове (клъстери) от обособени процесни блокове. Отделните блокове се синхронизират трудно, процесите в един блок също не могат да се синхронизират с процесите принадлежащи на друг блок. Отделни "grid"масиви се генерират за всяка специфична изчислителна задача (наричана kernel ядро), обхващащи определени данни, като при това всеки различен грид може да се създава за нови и вече записани в паметта на картата данни (фиг. 3). Дименсиите на изчислителния клъстер са 1D, 2D и 3D, като максималния брой блокове в него не може да надвиши 65536 във всяка една посока.

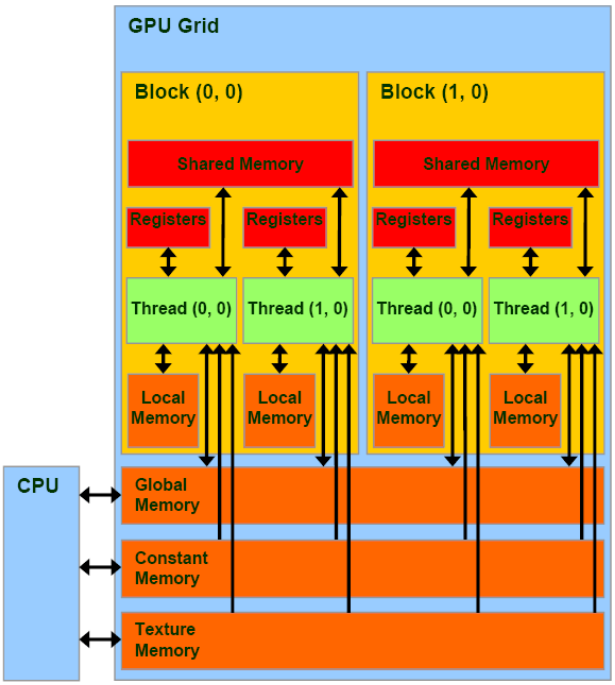
Kernel – ”ядро” това е глобална функция (_global_), която най-общо казано се изпълнява върху CUDA съвместимия хардуер, като тези функции могат да бъдат викани от основната програма или да се викат от други функции работещи върху CUDA. Ядрото обикновенно се свързва с даден набор функции, които имат за цел да извършат еднотипна обработка на всички данни записани в една изчислителна рамка. Ядрото може да вика функции за обработка на данни дефинирани за видими само в рамките на видеокартата (device_).



Фиг. 3. Разпределение на отделните процеси в блокове и ползване на различни рамки от процеси за различни изчислителни задачи.

2. Типове памет

За CUDA програмистите от съществено значение да правят разлика между типовете памет интегрирани във видео картата и съответно предимствата и недостатъците от нейното използване – фиг. 4:



Фиг. 4. Видове памети използвани в CUDA.

Глобална памет – това е инсталираната физическа памет на графичната карта, чийто обем може да варира между 128 и 4000 MB. Организацията й е линейна, в тази памет, както отделните мултипроцесори с изпълняваните върху тях kernel, а също така и драйверите на видеокартата работещи на хост компютъра могат да пишат и четат данни. Всички процеси могат да пишат и четат данни в глобалната памет, също така процеси стартирани на хост компютъра могат да пишат и четат от и в тази памет чрез функциите предвидени от драйвера на видеокартата. Четенето и записа в нея са най-бързи при четене и запис на данни в поредни адреси. За постигане на точни резултати се налага ползването на функции за синхронизация между процесите изпълнявани в отделни изчислителни блокове, така че да се гарантира, че даденият изчислителен блок ще приключи изпълнението на всички предвидени за изпълнение процеси преди да бъде зареден следващ изчислителен блок. Когато много процеси четат и записват данни в тази обща глобална памет програмата става бавна. Освен това когато два или повече процеса четат и записват данни от междинни изчисление в едни и същи адреси на глобалната памет се налага ползването на допълнителна синхронизация или т.н. атомарни операции, които ни гарантират, че никой следващ процес няма да получи монополен достъп до дадения адрес от глобалната памет преди предходния да приключи успешно. Най-високо бързодействие се постига при последователно четене и запис на 16 байтови блокове данни, имайте това предвид, когато реализирате kernel функции обработващи големи масиви данни. Понякога е удобно данните да се пакетират на 16 байтови групи, като самият kernel адресира отделните байтове по отделно, за да бъде изпълнена дадена изчислителна операция над тях.

Локална памет – това е памет в която може да чете и записва данни само един процес. Обикновено се ползва за съхраняване на междинни резултати от изчисления.

Споделена памет – всеки един мултипроцесор има малък обем от памет наричан споделен, с размер 16KB. Тази памет се използва от отделните процеси в рамките на този блок за бързо четене и запис на данни. Важно е да се знае, че споделената памет в един блок е видима само и единствено за процесите вътре в него. Като пример ще кажем, че е възможно да имаме няколко блока работещи едновременно върху един и същи мултипроцесор. Тази памет е регистрова и е в пъти по-бърза от глобалната памет на картата.

Памет за константи – GPU притежава и малък обем памет 8kb, която е по-бърза от глобалната памет, а достъпът до нея позволява по-висока степен на паралелизъм.

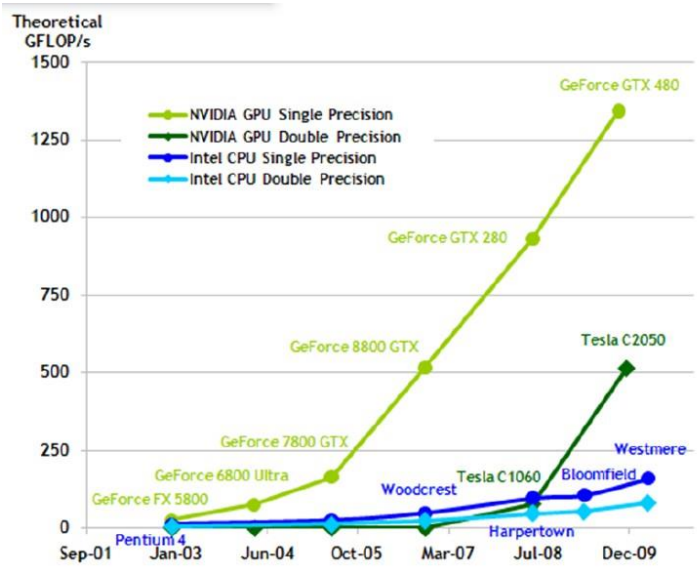
Памет за текстури – GPU притежава и текстурна памет, която може да се ползва при определени обстоятелства, като например за линейна филтрация на данни. Структурата на тази памет позволява запис на 2D матрици. Тази памет се кешира и е принципно само за четене.

3. Развитие на технологията

Проектът CUDA излиза на бял свят заедно с видео карта G80 (ноември 2006г.), като първата SDK е пусната февруари 2007г. Първата версия с номер 1.0 се поддържа до излизането на професионалните клъстерни карти Tesla (юни 2007г.) и е съвместима с процесорите G80. Тя е предназначена за пазара на изчислителни системи. Версия 1.1 въвежда функции за ползването на NVIDIA драйверите, като са поддържани карти след GeForce 8 и по-новите версия. Това е ключов момент за програмистите, тъй като прави възможно ползването на CUDA върху всички видове видеокарти, а също така и като симулатор върху стандартен компютърен процесор. Следва да се отбележи, че много от преимуществата са достъпни само за 64-битовата версия на Windows. Версия CUDA 2.0 е готова заедно с картите GeForce GTX 200, като бета версията се въвежда през началото на 2008. Следващата версия поддържа: изчисления върху нецелочислени променливи с двойна прецизност (float, double), като хардуерно това е постижимо само от модел GT200. Системата има поддръжка за Windows Vista (32и 64-битова версия), Mac OS X и Linux. Разработените с всяка следваща версия приложения могат да работят само със същата и по-нови версии на CUDA SDK, като не могат да бъдат поддържани от драйверите на по-старата версия. На фиг. 5 е дадено сравнение на производителността със съвременни Intel процесори излезли от производство в същия интервал от време. Наличната към момента стабилна версия е 4.0 CUDA TOOLKIT.

Тъй като това са видео карти, нормално е да се очаква, че ползваните в тях технологии за обмен на данните ще надвишават многократно скоростите на трансфер на данни между CPU и RAM на компютрите. Едновременно с това следва да се отчете, че NVIDIA забавят излизането на пазара на продукти с вградени атомични операции за работа с глобалната памет и споделените ресурси [4]. Вместо това се представят нови платки имащи увеличена тактова честота и потребление на енергоконсумацията. В действителност за пазара на игри това не е нужно, но именно този клас платки са масово достъпни като цени за повечето програмисти. За разлика например, една карта TESLA има до 4-5 пъти по-висока цена за същата производителност както една GeForce карта. Особената разлика, че TESLA

поддържа всички атомарни операции и има вградени АЛУ за аритметични операции с двойна прецизност [5].



Фиг. 5. Сравнение на производителността между поредни GeForce продукти и процесори Intel [6].

Освен тези общи особености всеки модел видео карти притежава специфични отличителни черти. Например всички карти имат съвместими функции позволяващи четене и запис на данни в глобалната и споделената памет, четене и запис на данни в текстурите и константната памет. Но подобно на различните технологии и поколения процесори и различните поколения мултипроцесори имат допълнителни функции. Едни от тези важни за бързодействието функции са тъй наречените атомарни операции. Важно е да се знае, че действия с атомарни операции над глобалната памет са достъпни в архитектура 1.1, а действия с атомарни операции над споделената памет са достъпни от архитектура 1.2. Същевременно програми написани за 1.0 са изпълними върху следващото поколение карти, без поддръжка на обратна съвместимост. Естествено ако смятате да ползвате технологията за инженерни изчисления и симулации се ориентирайте към версии 1.3 или 2.X.

ЛИТЕРАТУРНИ ИЗТОЧНИЦИ:

[1]. FARBER, R. CUDA, Supercomputing for the Masses: Part 1. *Dr. Dobb's.: The world of software development* [online]. April 15, 2008 [viewed 12 March 2015]. Available from: http://www.drdobbs.com

[2] SANDERS, J. and E. KANDROT. *CUDA by Example: An Introduct to General-Purpose GPU Programming*. Adisson-Wesley, 2010. ISBN 978-0-13-138768-3.

[3] CARDOSO, N. and P. BICUDO. *CUDA Memory Model, Física Computational (FC5)*. [online]. [viewed 10 March 2015]. Available from: https://fenix.tecnico.ulisboa.pt/downloadFile/563568428758049/CUDA_3.pdf

[4] BUCK, I. The Evolution of GPUs for General Purpose Computing. San Jose Convention Center, CA, September 20–23, 2010. *Nvivdia* [online]. [viewed 12 March 2015]. Available from: https://www.nvidia.com

[5] BOSAKOVA-ARDENSKA, A. and L. BOSAKOV. Parallel image processing with mean filter. In: *International scientific conference, Gabrovo, 16–17 November 2012*. 2012, pp. I-362-I-365 [online]. ResearchGate [viewed 14 March 2015]. Available from: https://www.researchgate.net

[6] *Nvidia* [online]. [viewed 12 March 2015]. Available from: https://www.nvidia.com

За контакти:
Гл. ас. д-р инж. Георги Петров, Департамент ”Телекомуникации” при НБУ, ул. Монтевидео № 21, 2-609, Тел.: 02 8110609, e-mail: gpetrov@nbu.bg

Годишник Телекомуникации 2015, том 2, с. 9 - 17

Yearbook Telecommunications 2015, vol. 2, p. 9 - 17

eISSN 2534-854X https://telecommunications.nbu.bg/bg/godishnik-telekomunikacii

Contacts:
Assist. Prof. Georgi Petrov, PhD, Department Telecommunications, New Bulgarian University, 21 Montevideo St., Tel.: 359 2 8110609, e-mail: gpetrov@nbu.bg

Дата на постъпване на ръкописа (Date of receipt of the manuscript): 07.05.2015.
Дата на приемане за публикуване (Date of adoption for publication): 01.09.2015.