СОФИЙСКИ УНИВЕРСИТЕТ
"СВ. КЛИМЕНТ ОХРИДСКИ"
ФАКУЛТЕТ ПО МАТЕМАТИКА И
ИНФОРМАТИКА
КАТЕДРА ИНФОРМАЦИОННИ ТЕХНОЛОГИИ

Разработване на алгоритми за паралелизация на изчисления върху видео карта

Дипломна работа

Научен Ръководител:
доц. д-р Леандър Литов

Дипломант:
Иван Табанлийски

специалност "Разпределени системи и мобилни технологии"
фак. № 22574

СОФИЯ 2009

Съдържание

Списък на фигурите

Глава 1

Увод

GPU (Graphics Processing Unit), с изчислителните си способности, ни дава възможност да видим света през компютъра по нов невиждан начин до сега. Развитието на GPU е свързано с разрешаването на най-големите предизвикателства на изчислимостта. По настояще разновидността на приложенията, които използват GPU е голямо и продължава да расте. CUDA (Compute Unified Device Architecture) технологията ни дава за първи път възможност по-гъвкаво да програмираме GPU с общо предназначение и да го използваме почти като супер компютър. Дори вече съществуват реални супер компютри, използващи CUDA GPUs (TSUBAME supercomputer, Tokyo Tech).

Настоящата дипломна работа е посветена на начини за паралелизация и разработване на нов тип хетерогенни алгоритми за HPC (high-performance computing) скалируемо паралелизиране на симулации на физически процеси при преминаване на заредени частици в газова среда. За целта, ще бъде използван CUDA програмния модел, който е хетерогенен и дефинира новата архитектура наречена SIMT (single-instruction, multiple-thread) разработена от NVIDIA и имплементирана от мултипроцесорите на NVIDIA GPU. Конкретно ще бъде имплементирана симулация и изложени резултати за развитие на електронна лавина в RPCs (Resistive Plate Chambers), на базата на която могат да се проектират симулации и за други подобни задачи. Също така, ще бъдат обсъдени възможности и за по-нататъшно повишаване на ефективността и производителността им. Тъй като, създаването, имплементирането, откриването и отстраняване на грешки е значително по-трудно при паралелните алгоритми, е разработен еквивалентен сериен алгоритъм, изпълняващ се на CPU и служещ за сравнение, и оценка на ускорението осъществимо с CUDA.

Дипломата работа се състои в четири части. В първата е обсъде-

но какво е CUDA, нейното настоящо развитие и как възнамеряваме да я използваме. Във втората са описани накратко възникващите физически процеси при преминаване на заредени частици в газови среди и по-конкретно в RPC, тяхното математическо представяне и тримерният Монте-Карло симулационен модел, който ще използваме. В третата част е описано конкретно разработените сериен CPU алгоритъм и хетероген CUDA алгоритъм с възможните негови модификации. Последната част е посветена изцяло на резултати, коментари и сравнения.

Глава 2

CUDA архитектура

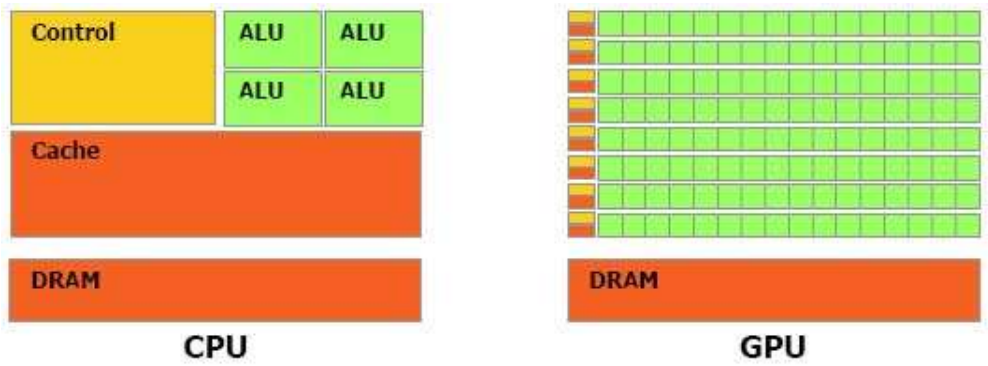
Тази глава въвежда в паралелните изчисления и използването на CUDA, като скалируема програмна среда за GPU масивно паралелни изчисления с общо предназначение.

Паралелно изчисление е едновременно използване на множество изчислителни ресурси за решаването на дадена задача. Таксономията на Флин класифицира мултипроцесорните компютърни архитектури, според това как се изпълнява потокът от инструкции и използват данните, използвани от тях, получавайки се следните модели: SISD[1], SIMD[2], MISD[3], MIMD[4]. Краят на 80-те и началото на 90-те години на миналия век, е съпроводен с бурно развитие на SIMD изчислителни машини с масивно паралелни процесори, които са били за времето си истински супер компютри. Но заради тяхната цена, ограничено разпространение и излизащите на пазара евтини и мощни микропроцесори, тези супер компютри са били изместени от големи клъстери от PCs с микропроцесори. Прави се преход от SIMD масивно паралелни процесори към SPMD (Single Program, Multiple Data, което е под категория на MIMD) разпределени системи. Но през 2000-та година скоростта на нарастване на мощността на микропроцесорите започва сравнително осезаемо да намалява и в резултат построяването на по-мощен клъстер означава използване на повече микропроцесори, което има силно отрицателен ефект върху скалируемостта, общата ефективност и енергономичност на клъстера. Тази ситуация прави масивно паралелните процесори отново популярни.

GPU са масивно паралелни устройства, които в общия случай, грубо погледнато, спадат към SIMD модела поради високата регулярност на

[1]SISD - Single Instruction, Single Data
[2]SIMD - Single Instruction, Multiple Data
[3]MISD - Multiple Instruction, Single Data
[4]MIMD - Multiple Instruction, Multiple Data

данните, с които работят - всяка изчислителна единица изпълнява една и съща инструкция върху различен елемент от данни. По-специално GPUs поддържащи CUDA са SIMT масивно многонишкови много ядрени чипове, които се характеризират с фино гранулиран (за отделни елементи на масив) паралелизъм на ниво данни, терафлопова производителност и хиляди нишки, изпълняващи се едновременно върху стотици скаларни процесори. CUDA дава възможност за имплементиране на алгоритмите, създадени за супер комютрите от 90-те години и прави масово достъпен HPC с GPU. На Фиг. 2.1 е илюстрирано грубо схематично сравнение на GPU с CPU, показващо колко повече транзистора за изчисления (за ALU (Arithmetic logic unit)) са отделени при GPU.

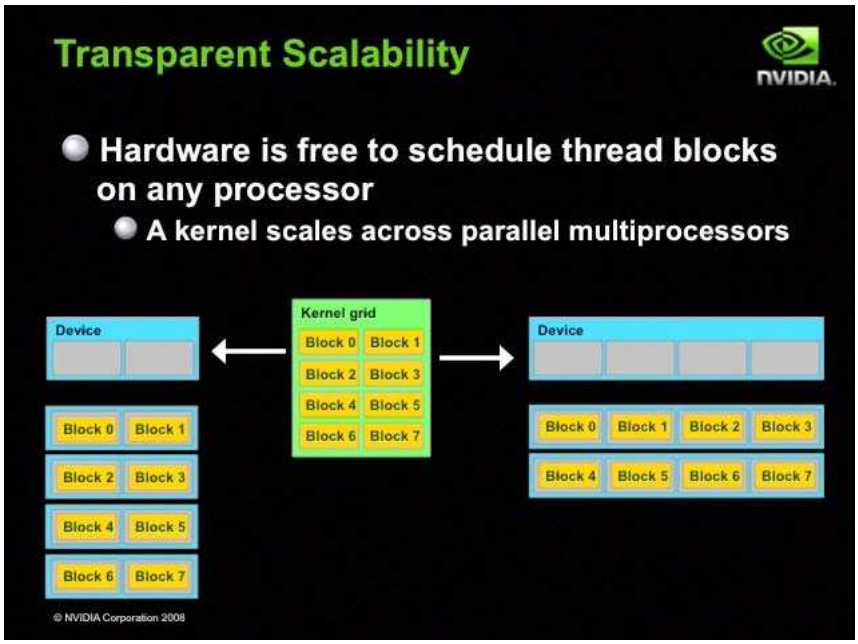


Фигура 2.1: Съпоставка на GPU и CPU [7].

2.1 CUDA програмен модел

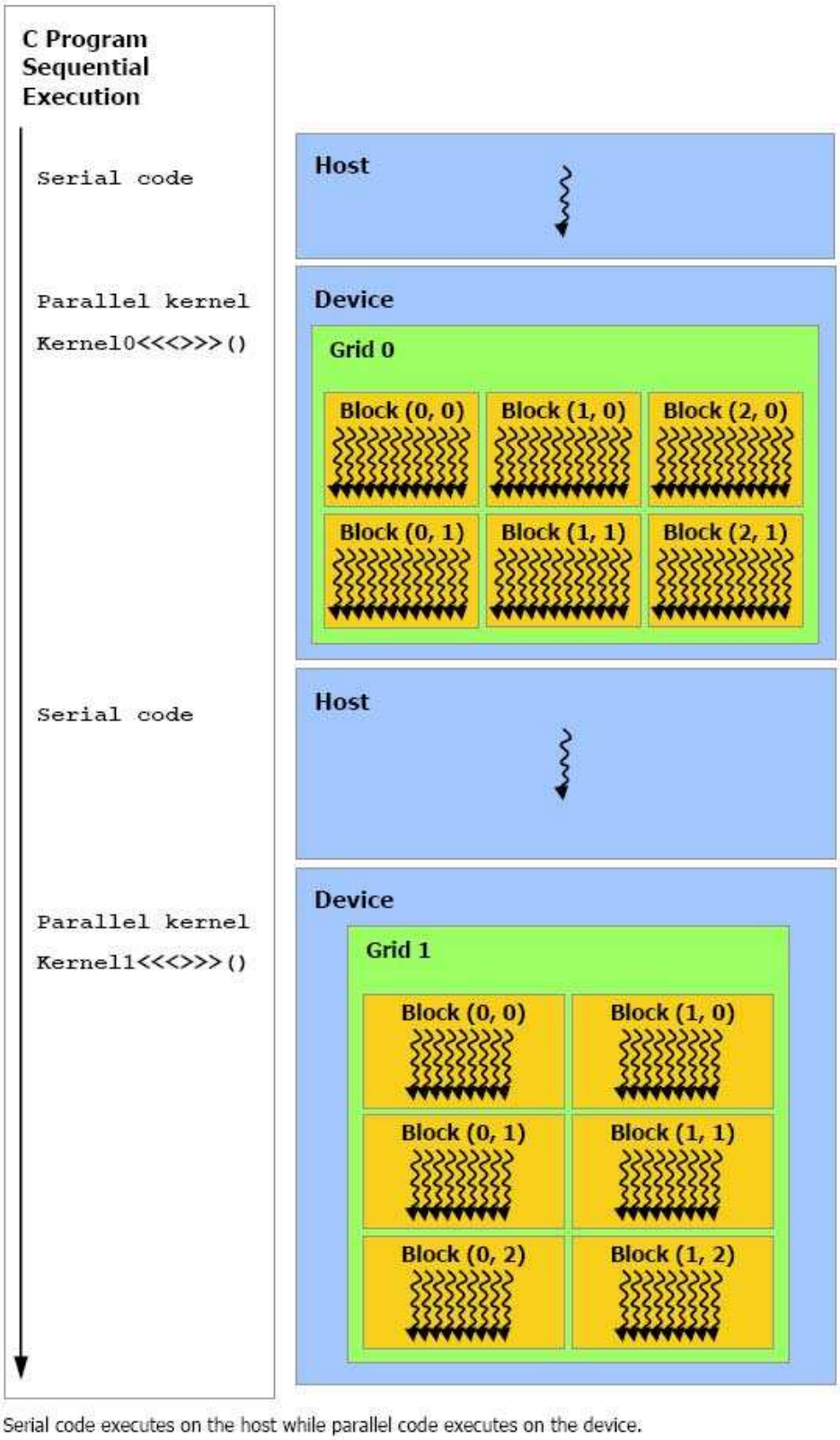
CUDA е паралелен програмен модел и софтуерна среда, която дава възможност за програмиране на GPU с общо предназначение. Абстрактният ѝ паралелен модел се състои в задачов паралелизъм и груб паралелизъм на ниво данни, който от своя страна, се още подразделя на съответно нишков паралелизм и фино гранулиран паралелизъм на ниво данни. Казано по друг начин означава, че за да се постигне максимално експлоатиране на модела е необходимо: първо една задача да се разделя на независими под задачи (в решетка от блокове), които могат да се решат самостоятелно, и на втора стъпка всяка от тези задачи, да се под паралелезира на под задачи (изпълнявани от нишките на блоковете), които могат кооперативно (сътрудничейки си) да се решат. Така описаният модел позволява, при достатъчно на брой под задачи, прозрачно да се

повиши производителността с повишаване на изчислителния ресурс, тъй като независимите под задачи могат да се стартират на свободен или в последствие освободен ресурс - виж Фиг. 2.2.



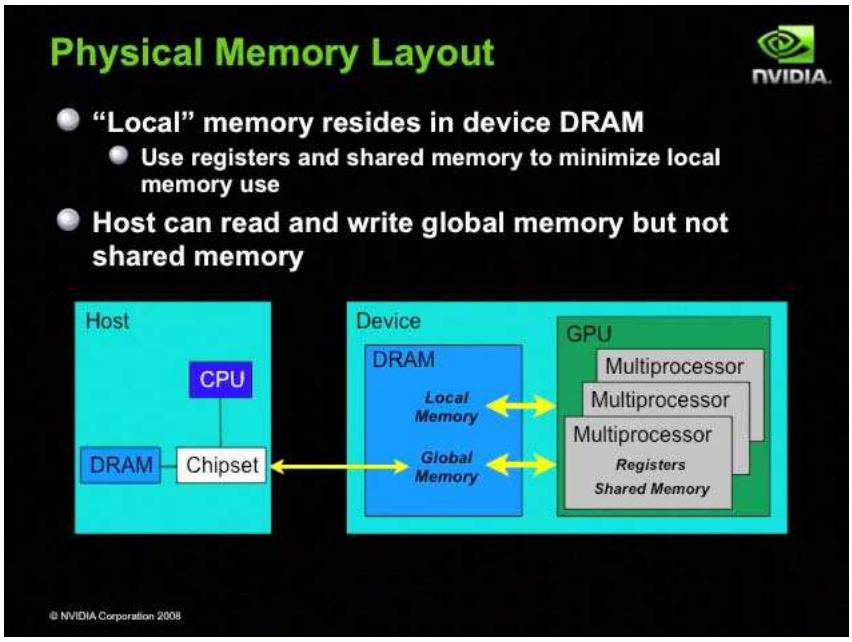
Фигура 2.2: Прозрачна скалируемост [7].

CUDA е разширение на C програмния език и по този начин позволява директна имплементация на паралелен алгоритъм. CUDA е проектирана за хетерогенни изчисления, където серийната част от кода се изпълнява на CPU, а паралелната се стартира на GPU като ядрова (kernal) функция - CUDA функция, стартираща се от CPU и изпълняваща се от всяка GPU нишка. На Фиг. 2.3 е илюстрирана хетерогенността на модела, като имаме предвид, че CPU кода продължава да се изпълнява веднага, след стартиране на ядровата функция и не се изчаква нейното завършване, освен, ако не е извикано експлицитно синхронизиране, чрез CUDA API. По този начин е възможно CUDA код да замести или да се добави към вече съществуващи приложения. GPU нишките са изключително олекотени, в смисъл, техния жизнен цикъл и превключване е изключително не натоварващо и почти прозрачен - хиляди нишки могат да се създадат в няколко такта. Когато една нишка, е принудена да чака, поради някакви причини, като достъп до паметта, тя автоматично се превключва от хардуера и се дава възможност на друга нишка да се изпълнява, без да има деградация на изпълнимостта. В един момент, може да се



Фигура 2.3: Хетерогенно програмиране [7].

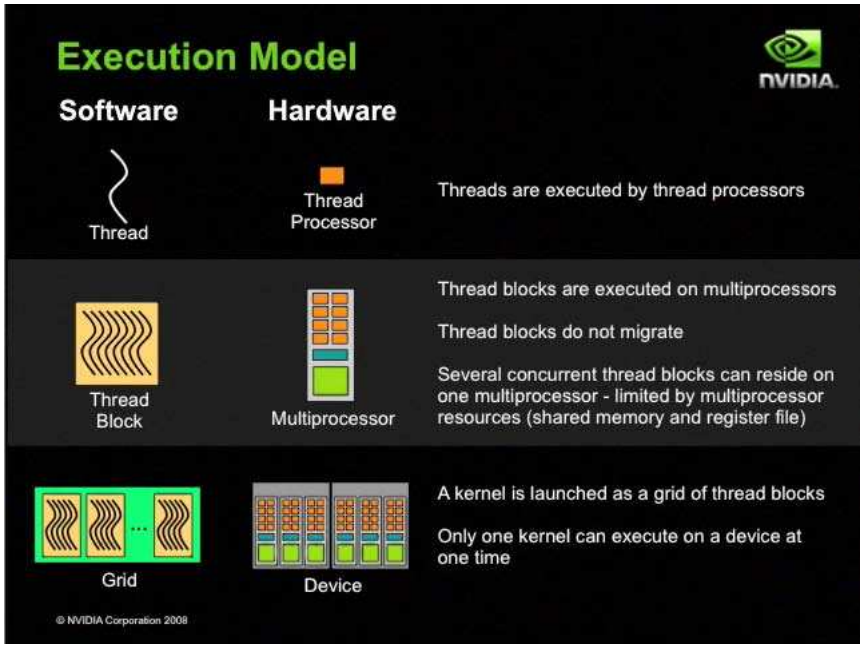
изпълнява само едно ядро и всяко ядро се изпълнява от предварително определен брой нишки, специфичен за съответната ядрова функция. Възможно е, също така да се пускат паралелно CUDA процеси на CPU и да се използват множество GPU от един CPU процес. CUDA третира CPU и GPU като различни устройства с отделна памет. Това позволява едновременно изпълнение на серийната и паралелната част от кода, без да се състезават за използването на паметта. На Фиг. 2.4 е илюстрирано физическото разположение на различните видове памети.



Фигура 2.4: Физическо разположение на различните видове памети [7].

Ядрата се стартират като масиви от паралелни нишки, всека нишка изпълнява един и същи код и си има идентификационен номер (ИН), служещ за неговото отличаване от другите, изчисляване на достъп до паметта и определящ еволюцията му. Нишките могат да кооперират, споделяйки резултати през глобалната памет на GPU, чрез атомарни операции или чрез споделената памет, намираща се на самите мултипроцесори, като от тук нататък ще я наричаме само споделена памет. Последното позволява драстично намаляване на достъпа до глобалната памет и напълно премахване на ограниченията на изпълнимостта свързани с това. Тъй като, достъпът до споделената памет е няколко порядъка по-бърз, всичките нишки стартирани за изпълнението на едно ядро са разделени в блокове, като само нишките от един блок могат да се

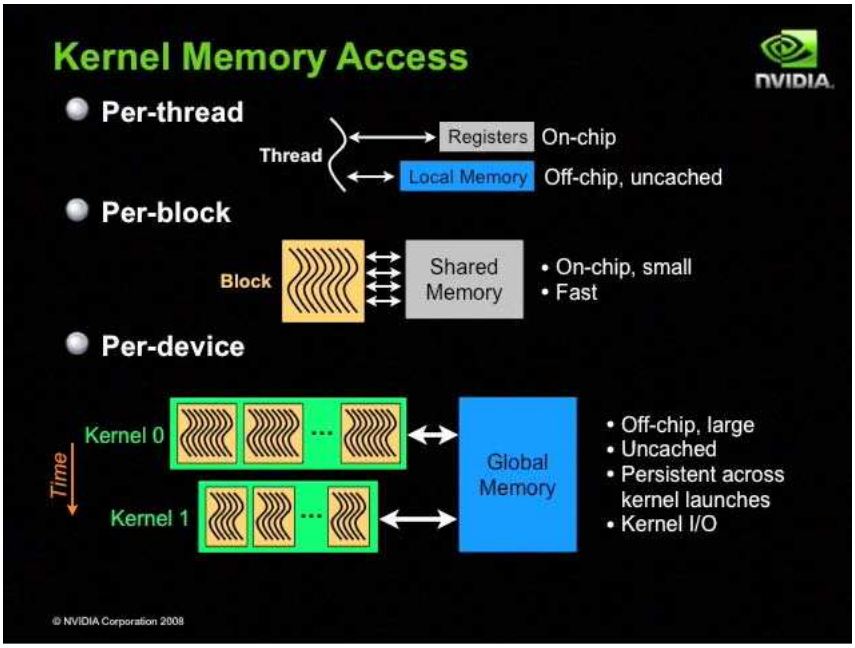
бариерно синхронизират и да кооперират през споделената памет на съответния мултипроцесор, който изпълнява блока. Всеки блок от нишки се изпълнява само от един мултипроцесор, като максималната му големина зависи от модела на GPU. Различните модели GPU имат различни характеристики и тенденцията е към увеличаване на броя на мултипроцесорите, тяхната гъвкавост и техните възможности - споделена памет, брой едновременно стартирани нишки и др. На Фиг. 2.5 е илюстриран модела на изпълнение. Когато се стартира ядрова функция, хардуера



Фигура 2.5: Модел на изпълнение [7].

стартира решетка от блокове от нишки. В един момент се изпълняват толкова блокове, колкото GPU може да поеме. Веднага, след изпълнението на някои от блоковете, на негово място се стартира нов блок и така, докато се изпълнят всичките. Именно тази техника дава възможност за прозрачна скалируемост, имайки предвид, че за максимално използване на GPU е нужно да има достатъчно блокове - минимум броя на мултипроцесорите, и е желателно да няма блок, който да се изпълнява несравнимо по-бавно от всички останали взети заедно. Всеки мултипроцесор, съставен от скаларни (нишкови) процесори, изпълнява изчисленията в SIMT режим, което е подобно на SIMD, с изключение на това, че всяка нишка се изпълнява независимо от другите и има собствен файл с регистри, стек, инструкционен показател и локална памет. Всеки ска-

ларен процесор може да превключва изпълняващата се нишка с един такт. Тази възможност прави SIMT много по-гъвкава, давайки възможност за паралелен код за паралелелизъм на ниво данни с координирани нишки, напълно еквивалентно на SIMD, и напълно независим паралелен код на нишково ниво за независими скаларни нишки, еквивалентно на MIMD. Всяка нишка има достъп до различни видове памет с различни характеристики - виж Фиг. 2.6 и Фиг. 2.7.

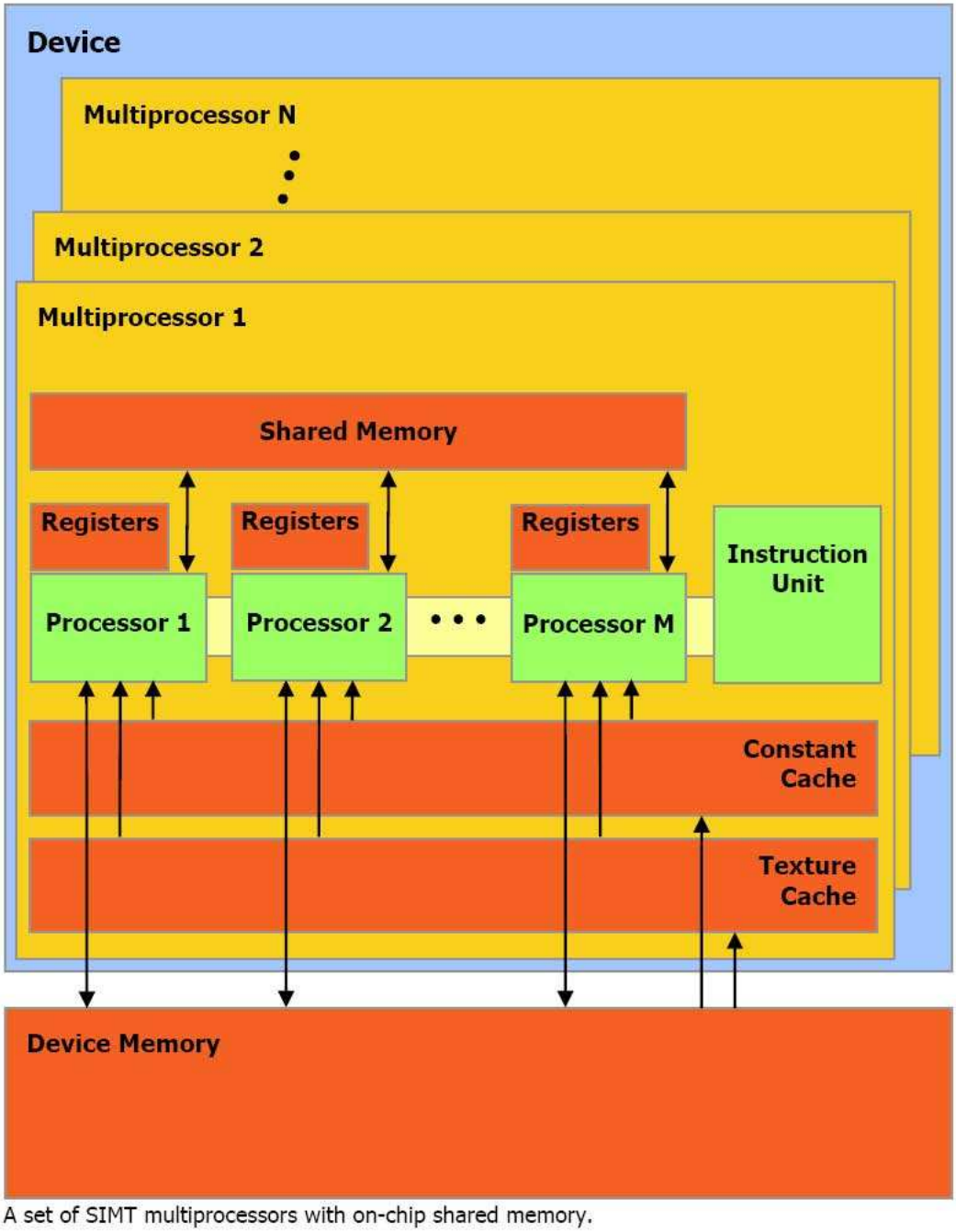


Фигура 2.6: Достъп до памети [7].

2.2 CUDA API

В тази част ще бъде накратко описан използваният програмен интерфейс на CUDA. За по-детайлно описание и справка за CUDA API се обърнете към NVIDIA CUDA Programming Guide [7].

Разширението към езика C включва: разширение на типовете на функциите, разширение на типовете на променливите, добавени са вградени променливи, допълнителни типове данни и части от стандартната C библиотека.



Фигура 2.7: Хардуерен модел [7].

Разширение на типовете на функциите

Ако не е указано друго, функциите по подразбиране са от тип __host__, изпълняващи се на CPU. Спецификаторът __global__ специфицира функцията, като ядрова функция. Този тип функции се изпълняват на GPU от всички нишки, чийто брой е зададен при извикването й в изпълнителната конфигурация. Изпълнителната конфигурация се състои от четири параметъра: два задължителни - големината на решетката от блокове в две измерения и големината на блока в три измерения. Функциите, които в последствие могат да се извикват от GPU нишките са серийни функции и се специфицират с спецификатор __device__. Тези функции са аналог на __host__ функциите, извикващи се от CPU нишката.

Разширение на типовете на променливите

Променливите, които се разполагат в глобалната памет на GPU се специфицират с __device__, а като се добави и __constant__ се разполагат в константната част от паметта. Тези променливи са достъпни през цялото време на изпълнение на приложението от всички GPU нишки и са достъпни от CPU кода през runtime библиотеката. Променливите, разполагащи се в споделената памет на мултипроцесорите, се специфицират с __shared__. Тяхното съществуване продължава докато се изпълнява съответния блок и са достъпни само от нишките на блока.

Добавени вградени променливи

За удобство са добавени дву-,три- и четиримерни векторни типове, съответстващи на стандартните скаларни типове. Те представляват структури с полета x,y,z,w и имат наименования от типа uint4, което означава четиримерен unsigned int. Изключение прави dim3, който е uint3, но чрез конструктора си инициализира полетата си с едно.

Вградени променливи

Вградените променливи се използват за информиране на съответно изпълняващата се GPU нишка, относно конфигурацията на стартираната ядрова функция и за самоидентифициране измежду всички останали. gridDim и blockDim са от тип dim3 и съдържат съответно големината на решетката и на блок в решетката. blockIdx и threadIdx са от тип uint3 и съответно за всяка нишка имат различна стойност идентифицираща нишката. Променливата warpSize е от тип int и е специфична за всяко

GPU. Автоматично се инициализира с големината на групата, на която SIMT мултипроцесора групира нишките в един блок. Нишките от една група стартират изпълнението си от един и същи адрес, но могат да се разклоняват независимо една от друга в изпълнението си. За постигане на максимална производителност е необходимо това да се избегне, ако алгоритъмът го позволява.

Глава 3

Физика и 3D модел за развитие на електронна лавина

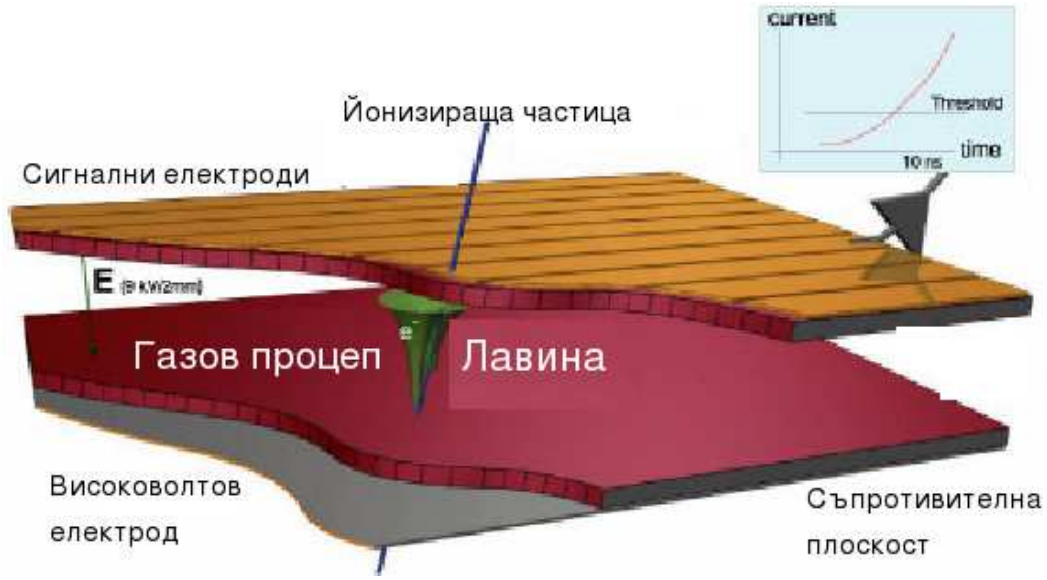
3.1 Обща физическа постановка

3.1.1 Принцип на действие на RPC

Камерите със съпротивителна плоскост са детектори на йонизиращи частици. По бързо действие са сравними със сцинтилаторите и притежават добра пространствена разделителна способност.

Конструктивно са изградени от две паралелни плоскости със сравнително голямо специфично съпротивление (10^{10} - 10^{11} Ωcm), отстоящи на разстояние от порядъка на няколко милиметра една от друга и образуващи газов процеп (фиг. 3.1). От външната страна на плоскостите е нанесен проводящ слой, който формира захранващите високоволтови електроди. Върху високоволтовите електроди е нанесен изолационен слой и върху него са разположени сигналните електроди, изработени най-често от медно или алуминиево фолио. При някои конкретни реализации, с цел намаляване на шума на камерата от вътрешната страна на съпротивителните електроди, е нанесен слой от ленено масло [10].

При преминаване на йонизиращо лъчение, през активния обем на детектора, в работния газ се образуват електрон-йонни двойки. Под действието на приложеното електрично поле ($\sim 45-50\ \frac{kV}{cm}$) първичните електрони започват да дрейфът към анода, предизвиквайки вторична йонизация. Дрейфът на отрицателния заряд към анода индуцира токов импулс върху сигналните електроди.



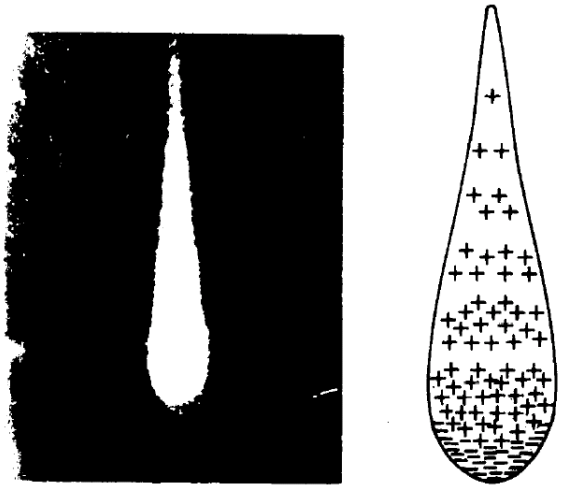
Фигура 3.1: Схема на единичен газов процеп на RPC камера [9]

Развитие на електронна лавина в газове

С цел, въвеждане на някои основни понятия от физиката на газовите разряди, нека да разгледаме поведението на клъстер от n_0 на брой свободни електрона, намиращи се между два безкрайни плоски електрода разположени на разстояние d един от друг. Нека за определеност координатната ос x е перпендикулярна на електродите и катода има координата $x = 0$ в тази система, а клъстерът от електрони и йони има координата x_0.

Под действието на електричното поле, създадено от потенциалната разлика между двата електрода, електроните започват движение към анода, а положителните йони към катода. Ако външното електрично поле е над определена гранична стойност, движещите се електрони започват при сблъсъците с молекулите на газа да избиват електрони, при което се получава лавинен процес. Едновременно с това част от електроните могат да се захванат от атоми.

Поради голямата подвижност на електроните, формата на лавината в пространството наподобява капка течност, в долната част на която са електроните, а в горната - по-бавните положителни йони - Фиг. 3.2.



Фигура 3.2: Снимка на лавина в мъглинна камера (ляво). Вижда се характерната капкоподобна форма на лавината. Разпределение на заряда в лавината - положителните йони изостават от бързия електронен фронт (дясно) [11].

Ако означим с n_e броя на електроните в лавината, а с dn промяната на броя на електроните след преминаване на разстояние dx, имаме в сила следната зависимост:

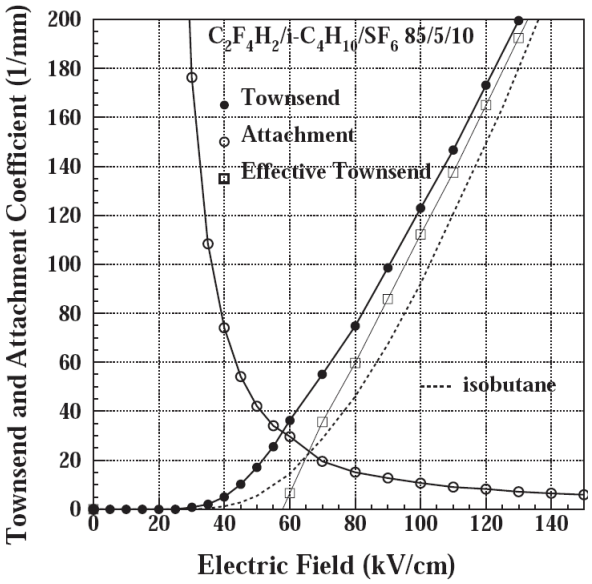
$$dn_e = n_e \eta dx \tag{3.1}$$

$$\eta = \alpha - \beta \tag{3.2}$$

където η е ефективния коефициент на Таунсенд (Townsend), α е коефициента на йонизация (първи коефициент на Таунсенд), β е коефициента на електронно захващане. Първият коефициент на Таунсенд α дава вероятността за йонизация на единица изминато разстояние. Ако Λ е средният свободен пробег на електроните, то $\alpha = 1/\Lambda$.

На фиг. 3.3 и фиг. 3.4 са представени ефективния коефициент на Таунсенд, коефициент за йонизация (първи коефициент на Таунсенд) и коефициента на електронно захващане за газове и газови смеси, често използвани в камерите със съпротивителна плоскост.

Броят на електроните $n_e(x)$ в точка с координати x (за определеност $x > x_0$) се получава, при предположение, че η не зависи от x, като се



Фигура 3.3: Ефективен коефициент на Таунзенд, коефициент за йонизация (първи коефициент на Таунсенд) и коефициент на електронно захващане, пресметнати с програмата IMONTE [12] за температура 296,15 K и налягане 1013 mbar [13].

интегрира (3.1):

$$G = \frac{n_e(x)}{n_0} = e^{\eta(x-x_0)} \tag{3.3}$$

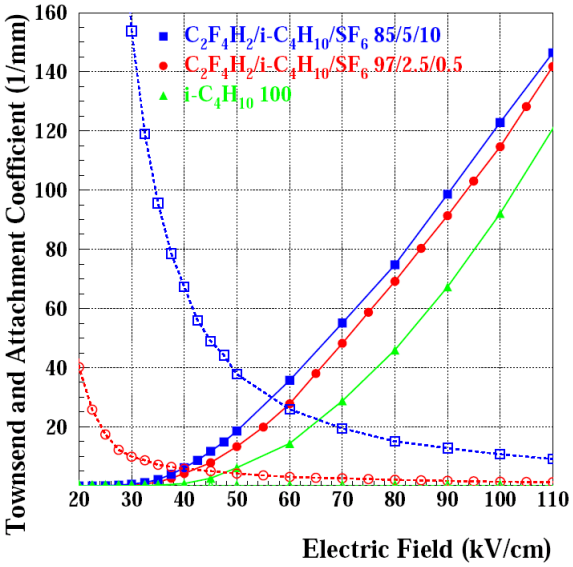
Величината G се нарича газово усилване.
По аналогичен начин за броя на положителните n_+ и отрицателните йони n_- имаме:

$$n_+(x) \approx \frac{\alpha n_0}{\eta} e^{\eta(x-x_0)} \tag{3.4}$$

$$n_-(x) = n_+(x) - n_e(x) \approx \frac{\beta n_0}{\eta} e^{\eta(x-x_0)} \tag{3.5}$$

За пълнота трябва да отбележим, че поради стохастични причини броят на електроните в лавината флуктуира. Вероятността $P(n)$ в лавината да има n електрона, при не големи стойности на редуцирания интензитет на електричното поле[1] E/p се задава от закона на Фъри (Furry) [15] [16], а именно:

[1] редуцираният интензитет на електричното поле E/p по дефиниция е интензитетът на електричното поле разделен на атмосферното налягане



Фигура 3.4: Коефициента на Таунсенд и коефициента на електронно захващане пресметнати с програмата IMONTE за различни газови смеси [12] [14].

$$P(n) = \frac{1}{N} e^{-\frac{n}{N}} \tag{3.6}$$

където $N = n_0 e^{\eta(x-x_0)}$. При големите стойности на E/p, при които оперират камерите със съпротивителна плоскост, вероятността в лавината да има n електрона се задава от разпределение на Поля (Polya):

$$P(n) = [\frac{n}{N}(1+\theta)]^{\theta} e^{-\frac{n}{N}(1+\theta)}, \tag{3.7}$$

където $\theta = 0,5$ [15]. Флуктуациите могат да се отчетат, като се добави допълнителен множител M, към (3.3):

$$n_e(x) = n_0 M e^{\eta(x-x_0)} \tag{3.8}$$

Развитие на лавина в RPC камера

При преминаване на йонизиращо лъчение през активния обем на детектора, в работния газ се образуват клъстери от електрон йонни двойки.

Под действието на приложеното електрично поле ($\sim 50\ kV/cm$), първичните електрони започват да дрейфът към анода, предизвиквайки лавинна вторична йонизация.

Да разгледаме $j-$тия клъстер, образуван в обема на детектора от йонизиращата частица. Нека n_j и x_j са съответно броят на електроните в клъстера и началната позиция на клъстера. Нека, за определеност, клъстерът с индекс $j = 1$ се намира най-близо до катода.

Лесно може да се определи разпределението на зарядите в газа. Зарядът на свободните електрони е:

$$Q_e(x) = Q_j e^{\eta(x-x_j)}, \tag{3.9}$$

където $Q_j = q_{el} n_j$, а q_{el} е зарядът на електрона.

Можем също така лесно да оценим статистическите флуктуации на x_j и на n_j, тъй като се дължат на статистически независими процеси.

Броят на клъстерите се подчинява на Поасонова статистика. Следователно, вероятността да намерим първия клъстер между x и $x + dx$ е разпределена експоненциално:

$$P(x) = \lambda e^{-\lambda x} \tag{3.10}$$

където λ е средната линейна плътност на клъстерите. На фиг. 3.5 е представена средната линейна плътност на клъстерите за някои газове и газови смеси.

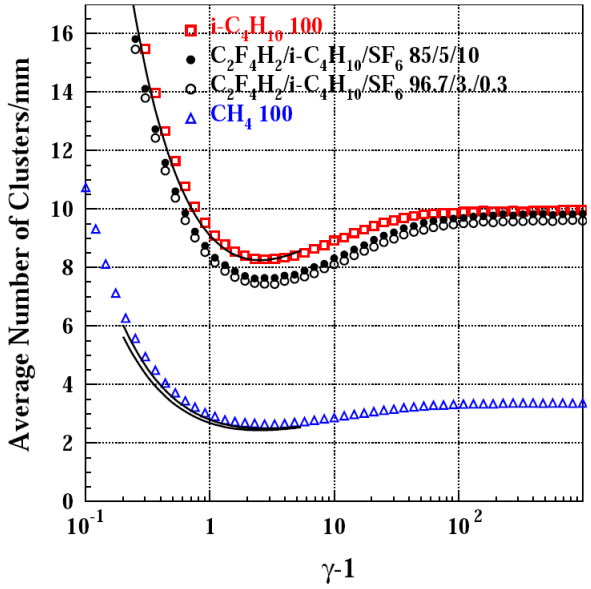
Използвайки Поасонова статистика [11] [19], може да определим, че вероятността $P_{cluster}(j, x)$, $j-$тия клъстер, независимо от останалите клъстери, да се намира между x и $x + dx$ се задава от:

$$P_{cluster}(j, x) = \frac{x^{j-1}\lambda^j}{(j-1)!} e^{-\lambda x}, \tag{3.11}$$

където λ е линейната плътност на клъстерите в газовата смес.

Флуктуации на броя на електроните в даден клъстер[2] може да се оценят, използвайки различни модели [20]. В случая, тъй като вторичните електрони в даден клъстер се отделят след малък брой независими един

[2] средният брой електрони в клъстерите, образувани в газа се нарича размер на клъстера на газа



Фигура 3.5: Среден брой клъстери на mm за различни газове при температура 296,15 K и налягане 1013 mbar в зависимост от кинетичната енергия на частиците, които ги пораждат, получени с програмата Heed. Плътните линии представляват измерванията за метан и изобутан [13] [17] [18].

от друг сблъсъка на първичния електрон с молекулите на газа, достатъчно адекватно описание може да се получи, като се използва Поасонова статистика, т.е. вероятността $P(n)$ да намерим клъстер с брой електрони между n и $n + dn$ е:

$$P(n)dn = \frac{\mu^n}{n!}e^{-\mu}dn, \tag{3.12}$$

където μ е средния брой на електроните в клъстер.

Средният заряд $\langle Q_e(x)\rangle$ на всички електрони в $j-$тия клъстер може да се пресметне, използвайки (3.9), (3.10) и (3.11):

$$\langle Q_e(x)\rangle \approx q_{el}\mu e^{\eta x}(\frac{\lambda}{\lambda+\eta})^j \tag{3.13}$$

Индуциране на сигнал върху електродите

Движението на електроните в електричното поле индуцира токов импулс (сигнал) върху сигналните електроди на камерата със съпротивителна плоскост. Положителните и отрицателните йони, поради по-малката си дрейфова скорост индуцират значително по-малък сигнал, който е пренебрежим в сравнение със сигнала, индуциран от електроните. Индуцирането на сигнал, върху електрод в многоелектродна система, може лесно да се изчисли с помощта на теоремата на Рамо [21] или обобщената теорема на Рамо [22], като се използват така наречените тегловни полета. За изчисляване на тегловните полета се използват различни приближени модели на камерите със съпротивителна плоскост [19] [23] [24].

Индуцираният токов сигнал според теоремата на Рамо е:

$$i(t) = \frac{E_w}{V_w} V_d q_{le} N(t), \tag{3.14}$$

където E_w е тегловното поле. Тегловното поле е електричното поле в газовия процеп, ако на сигналния електрод е подаден потенциал V_w, а всички останали електроди са свързани към земя. $N(t)$ е броя на електроните в момент от време t в лавината. V_d е дрейфовата скорост на електроните. На фиг. 3.6 е представена зависимостта на дрейфовата скорост от електричното поле за често използвани в камерите със съпротивителна плоскост газови смеси.

За да се определи полезният сигнал, който се индуцира, трябва първо да се определи тегловното поле на сигналния електрод. За целта, обикновено се използва опростен модел на камерата със съпротивителна плоскост [19] [23] [24]:

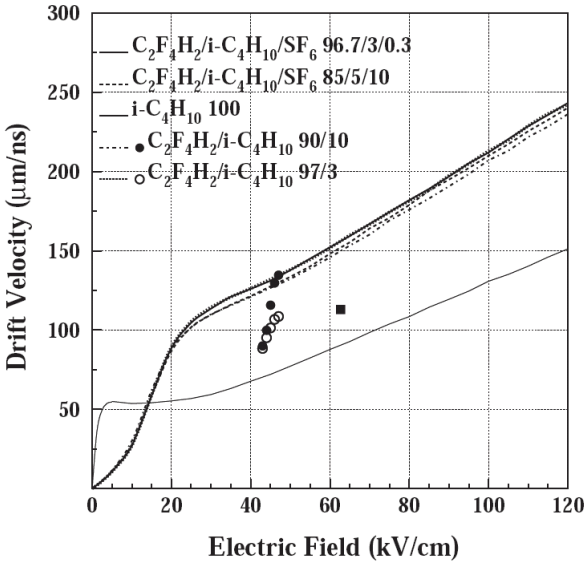
$$\frac{E_w}{V_w} = \frac{\epsilon_r}{\epsilon_r d + 2s}, \tag{3.15}$$

където s е дебелината на бакелитения или стъкления електрод, d дебелината на процепа, а ϵ_r е специфичната му диелектрична проницаемост.

Индуцираният заряд q_{ind} във външната верига може да се изчисли, като се интегрира (3.14) и се използва (3.13):

$$q_{ind} \simeq \frac{E_w}{V_w \eta} \langle Q_e(d) \rangle, \tag{3.16}$$

където $\langle Q_e(d) \rangle$ е сумарният електричен заряд, събран на анода.



Фигура 3.6: Линиите показват дрейфовата скорост в случай на различни газове при температура 296,15 K и налягане 1013 mbar, изчислени с програмата Magboltz [25] [13]. Кръгчетата показват експерименталните стойности [13] [26].

Камерите със съпротивителна плоскост за експеримента CMS се състоят от два газови процепа, между които са разположени сигналните електроди. Поради тази причина, зарядът индуциран върху сигналните електроди е сума от зарядите, които биха се индуцирали поотделно от всеки от газовите процепи [16] [19].

В случай на многопроцепна конфигурация, когато сигналните електроди са разположени върху пакет от n на брой газови процепи, за $\frac{E_w}{V_w}$ имаме:

$$\frac{E_w}{V_w} = \frac{\epsilon_r}{n\epsilon_r d + (n+1)s} \tag{3.17}$$

От тук, ясно се вижда, едно от големите предимства на двупроцепната конфигурация. Ако параметрите на газовите процепи, като ширина, материал и дебелина на бакелитните електроди, газова смес, газово усилване и т.н. са едни и същи за една двупроцепна и една трипроцепна камера, то тогава за камера с три и повече процепа, съгласно формули (3.15) и (3.17) $\frac{E_w}{V_w}$ има по-малка стойност, т.е. даже и трите процепа да са

дали принос, то индуцираният сигнал, в случай на трипроцепна камера ще е по-малък от този на двупроцепна камера.

3.1.2 Видове камери със съпротивителна плоскост

Камерите със съпротивителна плоскост могат да се разделят условно на няколко типа:

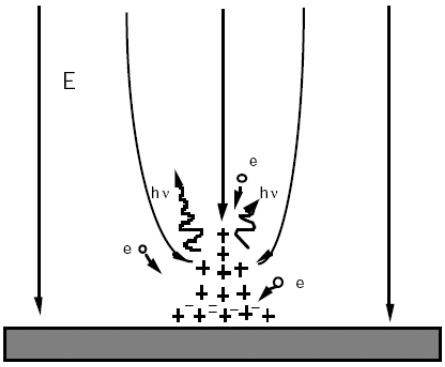
- според **физическите процеси**, които протичат в тях: камери, работещи в стримерен или лавинен режим
- според **предназначението си** : за измерване на време на прелитане (TOF[3]) или за формиране на тригер
- според **броя на газовите процепи** : еднопроцепни, двупроцепни и многопроцепни
- според **материала** : стъклени или бакелитени плоскости
- според **размера на газовия процеп**
- според **размера на сигналните електроди**

Всеки от тези типове има свое приложение, свои предимства и недостатъци.

Стримерен и лавинен режим на работа

От историческа гледна точка, най-напред са разработени камерите със съпротивителна плоскост, работещи в стримерен режим [27]. В стримерен режим, сигналът индуциран върху сигналните електроди е относително голям - между 50 pC [28] и няколко nC [29]. Поради тази причина не е необходим предусилвател и сигналът може да се подаде директно на дискриминатор, който да формира логически сигнал. Следователно електрониката, необходима за снемане на сигнал от камери, работещи в стримерен режим е сравнително проста и евтина [30]. Недостатъкът на този режим на работа е способността на камерите да работят при големи натоварвания, т.е. големи потоци йонизиращи частици, до около няколкостотин Hz/cm^2 е ограничена.

[3] TOF - време на прелитане от англ. Time of flight



Фигура 3.7: Формиране на стример. Лавината при своето разпространение излъчва фотони, те избиват електрони, които дрейфат към лавината [31].

Развитие на електронна лавина, под действието на електрично поле, може да се осъществи практически във всеки газ. Изборът на подходяща газова смес в действителност се ограничава значително от специфични експериментални ограничения [11].

Лавинното усилване настъпва в благородните газове при доста помалки стойности на приложеното електрично поле, отколкото при комплексните молекули. Това е следствие от възможността на комплексните молекули да приемат енергия, без да се йонизират. Възбудените благородни газове, могат да се върнат в основно състояние, само, чрез излъчване на фотон [11].

Многоатомните молекули, особено, ако имат повече от четири атома имат голям брой "неизлъчващи" възбудени състояния (ротационни и вибрационни), което позволява поглъщането на фотони в широк енергетичен интервал. Това е общо свойство на повечето органични съединения във въглеводородните и алкохолните семейства, а също така и на някои неорганични съединения, като фреоните, CO_2, BF_2 и други. Молекулите губят излишната енергия, чрез еластични сблъсъци или чрез дисоциация до по-прости радикали. Вторична емисия е малко вероятна, в случай, когато многоатомна йонизирана молекула се неутрализира на катода. При неутрализацията, радикалите рекомбинират или в по-прости молекули (дисоциация) или формират по-сложни комплекси (полимеризация). Добрата абсорбция на фотони и потискането на вторични емисии, позволява да се достигне голямо газово усилване, преди да е настъпил

разряд [11].

Гасящият ефект на многоатомните газове, по правило се увеличава с броя атоми в молекулата. Изобутанът $iso\text{-}C_4H_{10}$ често се използва за стабилизиране работата на детекторите, при голямо газово усилване [11].

Електронегативните газове, като например фреоните, освен че могат да поглъщат фотони, могат и да захващат свободни електрони, формирайки негативни йони, които от своя страна не могат да инициират лавина [11].

В детектори с метални електроди, значителна част от събрания в детектора заряд, ще се разреди чрез искра. В случай, на камера със съпротивителна плоскост, обаче, локалното зареждане със заряди на съпротивителния електрод, значително ще попречи на преминаването на искра. В случай, на фреон базирана газова смес, електричният заряд освободен от искра е много малък $\sim 1\ pC$ [31].

Камери със съпротивителна плоскост, работещи в лавинен режим, ще се ползват освен от експеримента CMS, също и в мюонната система на ATLAS [32] и в системата за измерване време на прелитане (TOF) на експеримента ALICE [33]. RPC камери, работещи в лавинен режим, са ползвани успешно в TOF системата на HARP [34], [35].

Камерите със съпротивителна плоскост, като тригерни детектори и детектори за измерване на време на прелитане

Във физиката на елементарните части, камерите със съпротивителни плоскости, намират приложение най-вече като специализирани тригерни детектори или като детектори за измерване на време на прелитане.

Поради добрата си времева разделителна способност ($\sim 1\ ns$), камерите със съпротивителна плоскост могат да се използват, като специализирани тригерни детектори на експерименти, провеждани на съвременните ускорители.

Експериментите CMS и ATLAS ще използват камери със съпротивителна плоскост с газови процепи с размер 2 mm за тригерни детектори. Прилаганите високи напрежения достигат до 10 kV и следователно, интензитета на електричното поле достига до 50 kV/cm. Камерите със съпротивителна плоскост, предназначени за тригерни детектори се изработват най-често с бакелитени електроди.

Камерите със съпротивителна плоскост използвани в TOF приложения [36] имат най-често газов процеп с размер от $0,2$ до $0,3$ mm в многопроцепна конфигурация [37]. TOF системите, използващи многопроцепните камерите със съпротивителна плоскост имат сходни характеристики с TOF системите, използващи сцинтилационни детектори, но са по-евтини от тях [38].

За TOF приложения обикновено се използват камери със съпротивителна плоскост със стъклени електроди. Приложеното електрично поле е по-голямо от случая на тригерни RPC и е около 100 kV/cm.

Експериментът ALICE ще използва за своята TOF система многопроцепни камери със съпротивителна плоскост с размер на процепите $0,25$ mm, работещи в лавинен режим [33] [39] и електрично поле около 100 kV/cm, като използваната газова смес е следната: $C_2F_4H_2(90\%)$,iso-$C_4H_{10}(5\%), SF_6(5\%)$. Експериментът HARP също разполага с TOF система, изградена от камери със съпротивителна плоскост [40].

В общия случай, камерите със съпротивителна плоскост за TOF приложения са многопроцепни, ефективността им достига до 99%, а разделителната им способност по време до 50 ps [36][40][41][42].

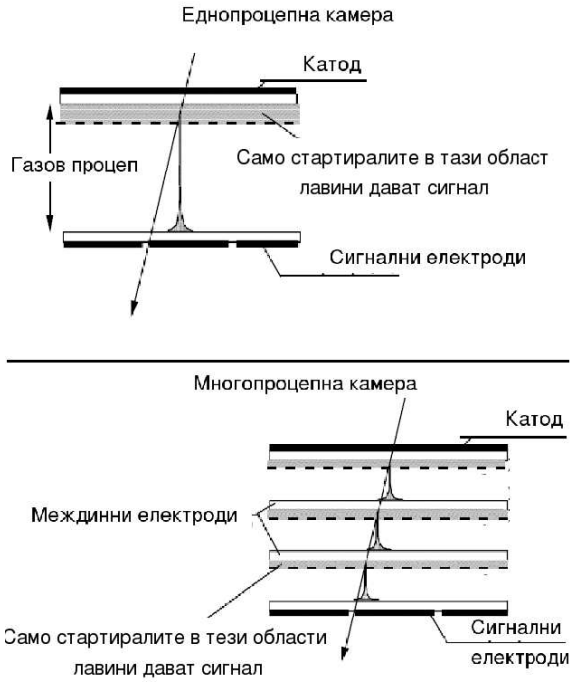
Еднопроцепни, двупроцепни и многопроцепни камери със съпротивителна плоскост

Камерите със съпротивителна плоскост са разработени оригинално, като детектори с една газова междина, т.е. еднопроцепна конструкция. На по-късен етап се разработват камери със съпротивителна плоскост с два и повече газови процепа.

При двупроцепната конструкция [44], сигналните електроди обикновено са разположени между двата газови процепа. Предимството е поголемият сигнал, индуциран върху сигналните електроди и от там поголямата ефективност. Трябва да отбележим, че в случай на двупроцепната конфигурация, ефективността не е просто логическо ИЛИ от сигналите на двата процепа, а е по-голяма. Това се дължи на факта, че може да се случи така, че при преминаване на йонизираща частица през двата процепа, лавините и в единия и в другия процеп по отделно да индуцират заряд върху сигналния електрод под прага на електрониката, но сумата от двата индуцирани заряда да е над този праг. В този

случай, ако работим в еднопроцепна конфигурация, ние няма да успеем да регистрираме йонизиращата частица, но в случай на двупроцепна конфигурация, ще я регистрираме. Този ефект беше наблюдаван при изследването RPC камерите за експеримента CMS [45]. Ефективността на различните RPC конфигурации е дискутирана подробно в [16].

Многопроцепни камери [37] със съпротивителна плоскост са предложени за първи път от членове на проекта LAA [46] през 1996 г. с цел, да се подобри с близо един порядък времевата разделителна способност на камерите със съпротивителна плоскост, като едновременно с това се запази тяхната ефективност.



Фигура 3.8: Сравнение на RPC камера с широк процеп и многопроцепна RPC камера. Виждат се областите, в които първичните клъстери дават принос към сигнала [37].

Газовият обем, в камерите със съпротивителна плоскост, служи едновременно за активен обем, в които се формират, под действието на йонизиращото лъчение, първичните клъстери и за газово усилване.

Както беше показано (3.3), броят на електроните в лавината, а от там

и индуцираният върху сигналните електроди сигнал, зависи от положението на първичния клъстер. Следователно, електрониката на камерата детектира лавини, създадени от първични клъстери, намиращи се в някаква неголяма област около катода.

За газов процеп с ширина 8 mm тази област е ограничена на около $1-1,5$ mm от катода. Тази неопределеност в положението на първичния клъстер води до вариации в предния фронт на сигнала. Ако широкият процеп е разделен на няколко по-тесни (фиг. 3.8), то тогава областта около всеки катод, в която първичните клъстери ще дават принос при формиране на сигнала, ще е много по-малка ($\sim 0,5$ mm) и следователно сигналът, ще варира по-малко във времето [37]. Благодарение на това, времевата разделителна способност се подобрява значително.

Материал: стъклени или бакелитени плоскости

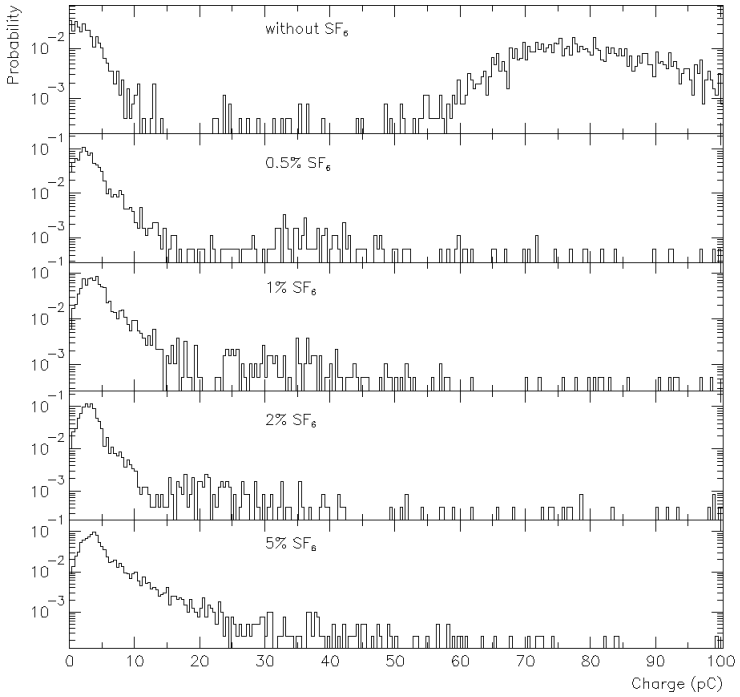
От една страна, съпротивлението на електродите потиска възникването на искри и стримери в обема на камерата, но от друга страна, твърде голямото съпротивление на електродите влошава характеристиките на камерата при работа в условия на голям поток йонизиращи частици. Ако потокът йонизиращи частици върху камерата е много голям, т.е. камерата работи при голямо натоварване, то в този случай, отделеният в камерата заряд е голям. Поради голямото съпротивление, натрупаният в активния обем заряд, не може бързо да се снеме през веригата на високоволтовото захранване, което води до увеличаване на мъртвото време, а от там и до намаляване на ефективността при големи натоварвания на камерата. Поради тази причина, се налага да се постигне компромис при подбора на материал за изработка на съпротивителните електроди. За тригерни приложения, особено за експериментите поставени на LHC, работоспособността при големи потоци от йонизиращи частици е много важна и поради тази причина, електродите се изработват от бакелит, който има по-малко съпротивление от стъклото. За TOF приложения обикновено се използват стъклени електроди, тъй, като очакваният поток през детекторите не е много голям.

Размер на газовия процеп

Според размерът на процепа, камерите със съпротивителна плоскост се делят основно на два типа - камери с тесен и камери с широк процеп [47].

При работа на камери със съпротивителна плоскост в лавинен режим е нежелателно наличието на стримери. Електрониката на камерите със

съпротивителна плоскост, работещи в лавинен режим е снабдена с много чувствителни предусилватели, така, че всеки стример, тъй като генерира голям сигнал, смущава голям участък от камерата. Стримерите могат да се подтиснат по два метода. Единият е, като се използва газова смес с голямо процентно съдържание на газ, абсорбиращ ултравиолетовите лъчи, а също така и електронегативни газове. Това най-често са различни видове фреон [48] [49] или серен хексафлуорид (SF_6) [50] [51] - фиг. 3.9. Другият метод е да се увеличи ширината на газовия процеп [43], което намалява вероятността за възникване на стример.



Фигура 3.9: Разпределение на заряда, получен от сигнален електрод при едно и също високо напрежение, но различна газова смес. Газовата смес съдържа $C_2H_2F_4$, *iso*-C_4H_{10} и SF_6. Процентното съдържание на *iso*-C_4H_{10} е 3%, докато съдържанието на SF_6 е различно за сметка на съдържанието на $C_2H_2F_4$ [51]. Забелязва се, че при по-голямо съдържание на SF_6 в газовата смес, зарядът е по-малък.

Камерите със съпротивителна плоскост, с тесен газов процеп, имат по-добра времева разделителна способност от тези с широк процеп. Камерите, с широк процеп, от своя страна имат по-добра способност за

работа, в случай на голям поток на йонизиращото лъчение. Това е така, поради по-малкия динамичен интервал от заряди на лавината и от там по-малък среден заряд, произведен в газовия обем. Поради тази причина, отделената мощност в газовия обем е доста по-малка, от случая, на камера с тесен процеп. Освен това, камерите с широк процеп, допускат по-голям механичен допуск при изработката.

Размер на сигналните електроди

Обикновено, за тригерни или TOF цели, необходимата пространствена разделителна способност е от порядъка на сантиметри. С цел, разширяване кръга на възможните приложения на камерите със съпротивителна плоскост, са направени успешни опити за подобряване на пространствената им разделителна способност, като за сигнални електроди се използват така наречените микрострипове, които всъщност представляват много тесни сигнални електроди [52]. Достигната е пространствена разделителна способност от 50 μm [53]. Възможно приложение на камерите със съпротивителна плоскост, с подобни характеристики, е конструирането на позитронно-емисионни томографи [4] апарати за медицински и фармакологични изследвания [54] [55].

3.2 3D Монте-Карло лавинна симулация

В тази секция, ще бъде очертана теорията на тримерният модел на развитие на електронна лавина в газ, която е описана по-подробно в [3, 4, 5], а в следващата глава всяка от стъпките, очертани тук, ще бъдат подробно разгледани със съответните им възможности за имплементация.

Вероятността за развитие на лавина до n електрона на разстояние z от началото на развитие, при условие, че α и η са константи, се дава със следната формула

$$\begin{aligned} P(n=0,z) &= k\frac{\bar{n}(z)-1}{\bar{n}(z)-k} \\ P(n>0,z) &= \bar{n}(z)\left(\frac{1-k}{\bar{n}(z)-k}\right)^2\left(\frac{\bar{n}(z)-1}{\bar{n}(z)-k}\right)^{n-1} \end{aligned} \tag{3.18}$$

където $\bar{n}(z) = e^{(\alpha-\eta)z}$ и $k = \eta/\alpha$. За генериране на случайно число n с разпределение 3.18, генерираме равномерно разпределено случайно число s

[4] англ. Positron Emission Tomography - Позитронно-емисионенна томография

в интервала $(0,1)$ и пресмятаме

$$\begin{array}{ll} n=0 & \\ n=1+trunc\left[\frac{1}{\ln\left(\frac{\bar{n}(z)-1}{\bar{n}(z)-k}\right)}\ln\left(\frac{(\bar{n}(z)-k)(s-1)}{(k-1)(\bar{n}(z))}\right)\right] & \begin{array}{l} s<k\frac{\bar{n}(z)-1}{\bar{n}(z)-k} \\ s>k\frac{\bar{n}(z)-1}{\bar{n}(z)-k} \end{array} \end{array} \tag{3.19}$$

За случаите когато $\alpha=0$ или $\alpha=\eta$ разпределение 3.18 става неопределено и вместо него използваме за $\alpha=0$ разпределението

$$\begin{aligned} P(n=0,z) &= 1-\exp(-\eta z) \\ P(n=1,z) &= \exp(-\eta z) \end{aligned} \tag{3.20}$$

За генериране на случайно число n с разпределение 3.20, генерираме равномерно разпределено случайно число s в интервала $(0,1)$ и пресмятаме

$$\begin{array}{ll} n=0 & s>\exp(-\eta z) \\ n=1 & s<\exp(-\eta z) \end{array} \tag{3.21}$$

За случаите когато $\alpha=\eta$ използваме разпределението

$$\begin{aligned} P(n=0,z) &= \frac{\alpha z}{1+\alpha z} \\ P(n>0,z) &= \frac{1}{(1+\alpha z)^2}\left(\frac{\alpha z}{1+\alpha z}\right)^{n-1} \end{aligned} \tag{3.22}$$

За генериране на случайно число n с разпределение 3.22, генерираме равномерно разпределено случайно число s в интервала $(0,1)$ и пресмятаме

$$\begin{array}{ll} n=0 & \\ n=1+trunc\left[\frac{1}{\ln\left(\frac{\alpha z}{1+\alpha z}\right)}\ln((1-s)(1+\alpha z))\right] & \begin{array}{l} s<\frac{\alpha z}{1+\alpha z} \\ s>\frac{\alpha z}{1+\alpha z} \end{array} \end{array} \tag{3.23}$$

Тези формули ще ги използваме за пресмятане на мултиплицирането на всеки електрон на разстояние dz на всяка стъпка от еволюцията на лавината За облекчаване на пресмятанията, без ограничаване на общността, може да разгледаме случая без съпротивителни плоскости, тъй като при него изразите за пресмятане на електричното поле $\vec{E}$ са значително по-прости и предотвратява значително възможността от натрупване на грешка от закръгления, което може да окаже значителен негативен ефект върху крайния резултат. Използваната формула за $\vec{E}$ е

$$\vec{E} = \frac{q(\vec{r}-\vec{r}')}{4\pi\varepsilon_0|\vec{r}-\vec{r}'|^3} \tag{3.24}$$

където $\overrightarrow{r}'$ задава местоположението на заряда, а $\overrightarrow{r}$ задава точката на наблюдение.

Използваният тримерен модел на развитие на лавина в газова среда се състои в следния алгоритъм по-общо изложен в [5].

1. В пространството, в което се намира газа въвеждаме Декартова координатна система и разделяме пространството на елементарни кубични обеми с страна dz. Времевата стъпка на симулацията dt намираме като разделим dz на дрейфовата скорост $\upsilon_D(E_0, p)$ на електроните при началното електрично поле E_0 и налягане p, което ще считаме за постоянно и от него няма да зависи симулацията. Как $\upsilon_D(E_0, p)$ зависи от E_0 изчисляваме чрез програмата Magboltz [25]. Така получаваме $dt = dz/\upsilon_D(E_0, p)$.

2. Поставяме един или повече електрони в обема на газа - в зависимост от физическия процес възбуждащ лавината, като се има предвид, че когато имаме повече от един електрон на разстояние от порядъка на елементарната дължина на дискретизиране на пространството ще се развие повече от една лавина. Едно от основните преимущества на 3D модела е, че не налага ограничение на броя на стартираните лавини и тяхното разположение. Единственото ограничение са възможностите на изчислителните ресурси

3. Пресмята се електричното поле във всеки елементарен обем, където има електрон и съответно за всеки елементарен обем се изчисляват: дрейфовата скорост $\upsilon_D(|\overrightarrow{E}|/p)$, коефициентът на йонизация $\alpha(|\overrightarrow{E}|/p)$, коефициент на електронно захващане $\eta(|\overrightarrow{E}|/p)$, напречният дифузен коефициент $D_T(|\overrightarrow{E}|/p)$ и надлъжният дифузен коефициент $D_L(|\overrightarrow{E}|/p)$.

4. За всеки един електрон на съответната стъпка се пресмята неговото мултиплициране на разстояние dz от текущото му местоположение, използвайки формули 3.19, 3.21, 3.23. За всеки нов електрон (в това число влиза и пораждащият мултиплицирането, който сега се премества) се пресмята относителното му преместване спрямо текущото му положение $(\overrightarrow{x}', \overrightarrow{y}', \overrightarrow{z}')$ в координатна система определена от $-E_x, -E_y, -E_z$, като се има предвид, че υ_D е колинеарен с обратна посока на $\overrightarrow{E}$, а D_L и D_T са съответно колинеарни и перпендикулярни на $\overrightarrow{E}$. Координатите x' и y' намираме чрез генериране на две Гаусово разпределени случайни числа съответно за x' и y' с параметри на Гаусовото разпределение $\mu = 0$ и $\sigma = D_T\sqrt{|\upsilon_D|dt}$, а

z' намираме по същия начин, но с параметри на Гаусовото разпределение $\mu = |\upsilon_D|dt$ и $\sigma = D_L\sqrt{|\upsilon_D|dt}$. Преходът от координатната система образувана от $(-E_x, -E_y, -E_z)$ или (x', y', z') в основната координатна система (x, y, z) се извършава чрез матрицата на въртене (Фиг. 3.10)

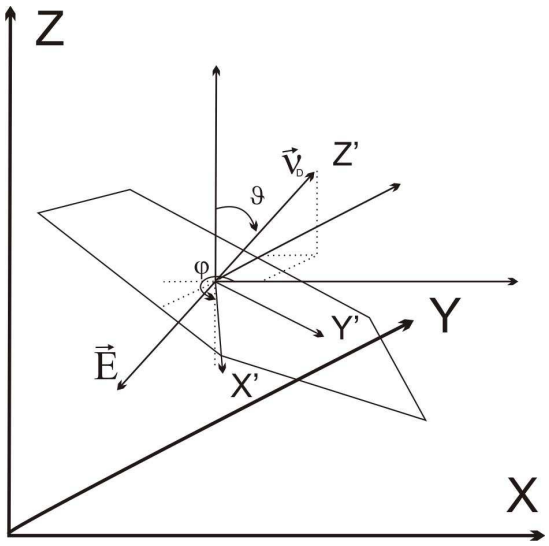
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi\cos\theta & -\sin\varphi\sin\theta \\ \sin\varphi & \cos\varphi\cos\theta & \cos\varphi\sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \quad (3.25)$$

където φ и θ можем да определим чрез отношенията

$$\tan\varphi = \frac{E_y}{E_x} \quad \tan\theta = \frac{\sqrt{E_x^2 + E_y^2}}{E_z} \quad (3.26)$$

и варианта на $\arctan(x)$, $\arctan 2(x, y)$, която връща резултат в интервала $(-\pi, \pi]$, който може да трансформираме в $[0, 2\pi)$ чрез добавяне на 2π към отрицателните стойности.

5. Стъпки 3 и 4 се повтарят докато електроните напуснат газа или по-нататъшното развитие на лавината не представлява интерес.



Фигура 3.10: Смяна на координатната система

Глава 4

CUDA имплементация на 3D модела на електронна лавина

Най-често, първа стъпка към създаването на паралелен алгоритъм, е създаването на съответната му серийна версия. Това помага за изясняване на естеството на задачата и изясняване на кои части или изцяло е най-ефективно да се паралелезират. Също, съпоставянето на резултати, изчислени по два начина, потвърждава коректността на имплементацията. Това е особено важно при числените експериментиране, тъй като неизбежната грешка от закръгления се натрупва по различен начин, в зависимост от последователността на изчислителните операции и тяхната точност. В нашия случай е дори още по-важно, тъй като представянето на числа, с плаваща запетая и точността на основните операции с тях при настоящите GPU не съвпада с това при CPU. Затова, в първата част от тази глава ще обърнем внимание на CPU имплементацията на 3D модела на развитие на електронна лавина, описано в предишната глава.

4.1 CPU сериен алгоритъм

Най-естествено е да разглеждаме всеки електрон като точков заряд, но тъй като всяко устройство за изчисления представя реалните числа с определена точност, то се налага да дискретизираме пространството на елементарни обеми, като моделираме заряд от един или повече електрона намиращи се в един елементарен обем като пространствено разпределен в него. Следователно колкото е по-малък елементарният обем, толкова по-точно можем да интегрираме по обема при пресмятането на $\overrightarrow{E}$ и съответно отчитането на ефекта от пространственото разположение на зарядите върху еволюцията на лавината. Но колкото с по-голяма често-

та дискретизираме обема, толкова повече памет и изчислителна мощ е необходима. Затова основният въпрос, който възниква е как да представим обема на газа. Една възможност е да се използва кубичен масив, адресиращ всеки елементарен обем в декартови координати. Ако дискретизираме всяко от измеренията на N части, то необходимата памет за съхранението на масива е $O(N^3)$, броя на итерациите необходими за изчисляването на резултатното електрично поле $\overrightarrow{E}$ във всяка клетка породено от всички останали клетки е $O((N^3)^2)$. Ако за развитието на една лавина са необходими M стъпки, където $M \geq N$, то само за пресмятането на електричното полето са необходими $O(M(N^3)^2)$ итерации. Става ясно, че такъв подход е крайно неефективен и дори да го реализираме би било още по-неефективно изследването на неговата точност чрез увеличаване на честотата на дискретизация.

За да се преодолее изложената по-горе неефективност, вместо масив се използва динамичната структура от данни - двусвързан нареден списък (ДНС), която ще съдържа информация само за елементарните обеми, в които има заряди. При детайлен анализ, този избор може да се окаже не най-оптимален, но тъй като CPU вариант не е основна цел, а само за сравнение, то относителната простота и задоволителната ефективност на ДНС го прави подходящ за целта. За всеки елементарен обем, в който има електрони и/или йони има съответен елемент в ДНС, който съдържа информация за координатите на елементарния обем, броя на електроните и броя на положителните и отрицателни йони. Елементите се нареждат спрямо координатите на елементарният обем, който описват. Наредбата е необходима за да се намират вече включени елементарни обеми и инкрементират техните полета описващи лавината, вместо да се добавят нови елементи. Това намалява значително големината на списъка и повишава ефективността в последващото му използване, въпреки, че добавянето не е за константно време.

CPU алгоритъмът е следния:

1. Създай списък L_1 с един или повече елемента инициализирани с началните координати на един или повече електрона/електрони. Създай празен списък L_2 и две променливи $numEls$ и $numElsFin$ и инициализирани с 0, съответно за отчитане на броя на електроните текущо в лавината и на броя на вече достигналите края на газовия процеп.

2. Ако L_1 не е празен, се итерира по-всички негови елементи, като на всяка итерация се прави следното с всеки един от елементите му:

   (a) Ако има положителни йони и/или отрицателни йони, добавят

се в L_2.

(б) Ако има електрони, сметни $\overrightarrow{E}$ в тази клетка

(в) Пресметни аналогично на точка 3 и 4 от секция 3.2 еволюцията на всеки един от електроните от текущия елемент и добави новите (като ако текущите продължават да се движат се разглеждат като нови) електрони в L_2, като се инкрементират подходящо $numEls$ и $numElsFin$.

3. Изтрий списък L_1 и размени L_1 с L_2, така вече L_1 е L_2, а L_2 е L_1.

4. Ако $numEls > numElsFin$ премини към стъпка 2, иначе лавината/те е/са напуснала/ли газовия процеп, като или е/са стигнала/ли отвъдния край или е изгаснала.

5. Ако искате повече от един опит преминете към едно, инициализирайки всички необходими променливи наново.

4.2 GPU алгоритъм

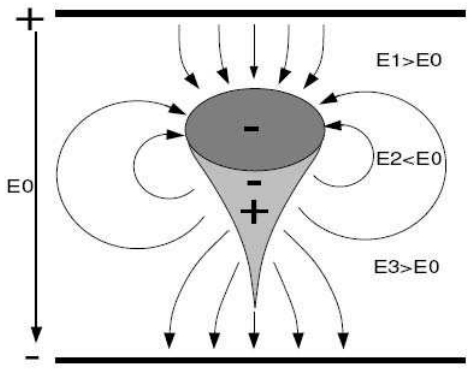
В тази част, първо накратко ще дискутираме произходът на разработеният хетерогенен CUDA алгоритъмът, а след това ще разгледаме съставните му части, като самостоятелни под задачи. Ще бъдат обяснени в детайли ядровите функции, техните параметри и съответните device функции, които те използват. След като вече сме се запознали с основните съставляващи, ще бъде изложено цялостното решение и ще бъде коментирано върху възможни подобрения. За информация относно конкретната имплементация е най-уместно да се прегледа директно кода, където има коментари.

При съставянето на паралелен алгоритъм, първият въпрос възникващ е, кои части от изчисленията биха могли да се извършват едновременно. Нашият случай има две главни под задачи, които са паралелни сами по себе си - пресмятането на сумарното електрично поле във всеки елементарен обем, където има електрони и пресмятането на еволюцията на всеки от електроните. След като сме открили съществуващият естествен паралелизъм, следващият въпрос е как и колко ефективно бихме могли да го експлоатираме при реализацията с разглежданата паралелна архитектура и нейните налични ресурси. За първата под задача разполагаме със SIMT архитектура, което ни навежда на мисълта да използваме по една нишка за всеки елементарен обем, където е необходимо да се пресметне $\overrightarrow{E}$. За втората под задача, тъй като CUDA е масивно нишкова архитектура, ако няма други съображения, изглежда, че е

най-ефективно еволюцията на всеки електрон да се пресмята от отделна нишка. Но както ще видим, не можем да използваме този подход, защото GPU не разполага с истински генератор на случайни числа. Следващият въпрос който възниква е как да съхраняваме лавината в паметта. Производителността на алгоритъма зависи както от самия него, така и от представянето на данните върху които работи. GPU първоначално е бил проектиран за ускоряване на работата с компютърна графика и все още е силно неефективен за работа с динамични структури от данни. Затова основна трудност е как да представим лавината, съобразявайки се с възможностите на GPU, която не е със статичен брой заряди, заема около 20% при квадратен газов обем, процесът ѝ на развитие и положението ѝ в обема на газа е с вероятностна природа и силно зависи от началните условия и конкретната постановка на задачата. Този най-основен въпрос е решен, чрез избора на две едновременни представяния, съответно за ефективно имплементиране на двете естествено паралелизиращи се под задачи, като на всяка стъпка от еволюцията се прави конвертиране от едното в другото. За пресмятане на $\overrightarrow{E}$ съответната ядрова функция взима като входен параметър масив от тип int4, описващ местоположението на всеки не свободен елементарен обем и неговото съдържание, а като изход връща масиви съдържащи необходимите данни за мултиплициране на електроните. Ядровата функция, мултиплицираща електроните, взима като вход тези данни и връща като резултат три масива – съответно по един за електроните, положителните и отрицателните йони. Тези три масива, описват съдържанието на целия обем на газа и информативното им съдържание се конвертира в един масив на следващият цикъл, служещ за изчисляването на $\overrightarrow{E}$.

4.2.1 Пресмятане на електричното поле

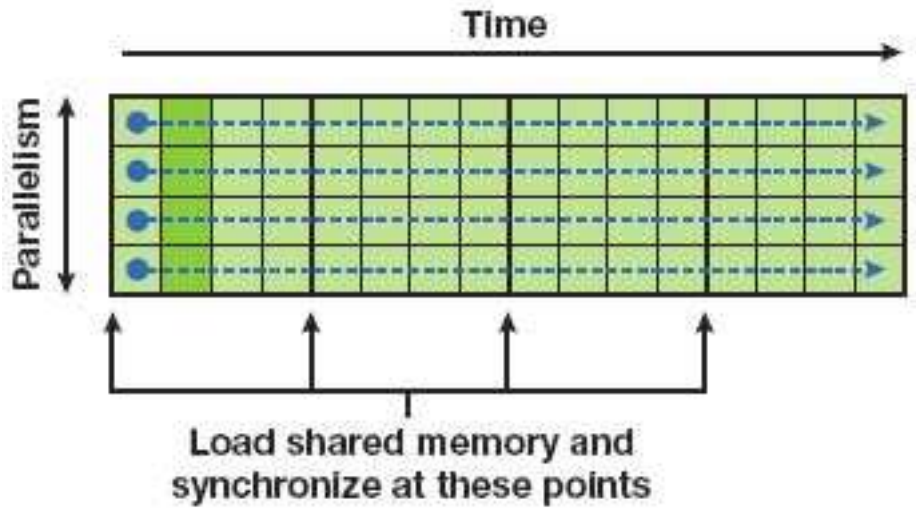
Електричното поле във всеки елементарен обем, където има електрони в разглеждания модел е сума от основното електрично поле, полето образувано от положителните и отрицателните йони и полето на електроните от останалите елементарни обеми. Схема на лавината и деформация на полето в пространството на газа е илюстрирано на Фиг. 4.1. За конкретното пресмятане ще използваме частен случай на алгоритъм, които може да се приложи за всяка задача с модел, в който има взаимно взаимодействие (всеки с всеки) между всичките ѝ елементи. Нас ни интересува, да пресметнем електричното поле в елементарните обеми, където има електрони, а не във всички ЕО, където има заряди. В общия случай сложността по време е $O(N^2)$ и в много случаи се търсят по-бързи приближения. Този тип задача поради огромното ѝ приложение в различни



Фигура 4.1: Електронна лавина и деформираното от нея поле

области и ресурсоемкостта си, се разглежда за паралелизиране още от самото зараждане на паралелните изчисления.

За да се възползваме максимално от CUDA, тук се предлага следният подход. Подреждаме ЕО съдържащи заряди в масив arr1, в следния ред – електрони, положителни йони, отрицателни йони, като един ЕО може да съдържа повече от един тип заряд и следователно да се появи повече от веднъж в този масив. За пресмятането на $\overrightarrow{E}$ във всеки ЕО, съдържащ електрон се стартира по една нишка, която серийно пресмята въздействието на всички останали върху него. По този начин теоретично постигаме $O(N)$ паралелизъм. Нишките се стартират в блокове, като всяка нишка копира данните в стил SIMD от един елемент на масив arr1 от глобалната памет в споделената памет на мултипроцесор, който го изпълнява. И затова, големината на arr1, трябва да се дели на големината на блока от нишки, защото всеки блок чете големината на блока елементи паралелно. След бариерна синхронизация, всяка една от нишките в блока пресмята на базата на данните в споделената памет. Така се получава фино гранулиран паралелизъм на ниво данни. Следва втора бариерна синхронизация, за да е сигурно, че всяка нишка е вече приключила с използването на всичките данни от споделената памет, и цикълът се повтаря докато всички елементи на arr1 се обработят - Фиг. 4.2. По този начин силно се облекчава четенето от глобалната памет, което е на порядък по-бавно и позволява да се постигне пикова производителност. Големината на блока се съобразява с възможностите на GPU и по-специално с размера на споделената памет, която трябва да събере данните, копирани от нишките на блока. Ако споделената памет е недостатъчна блокът не се стартира.

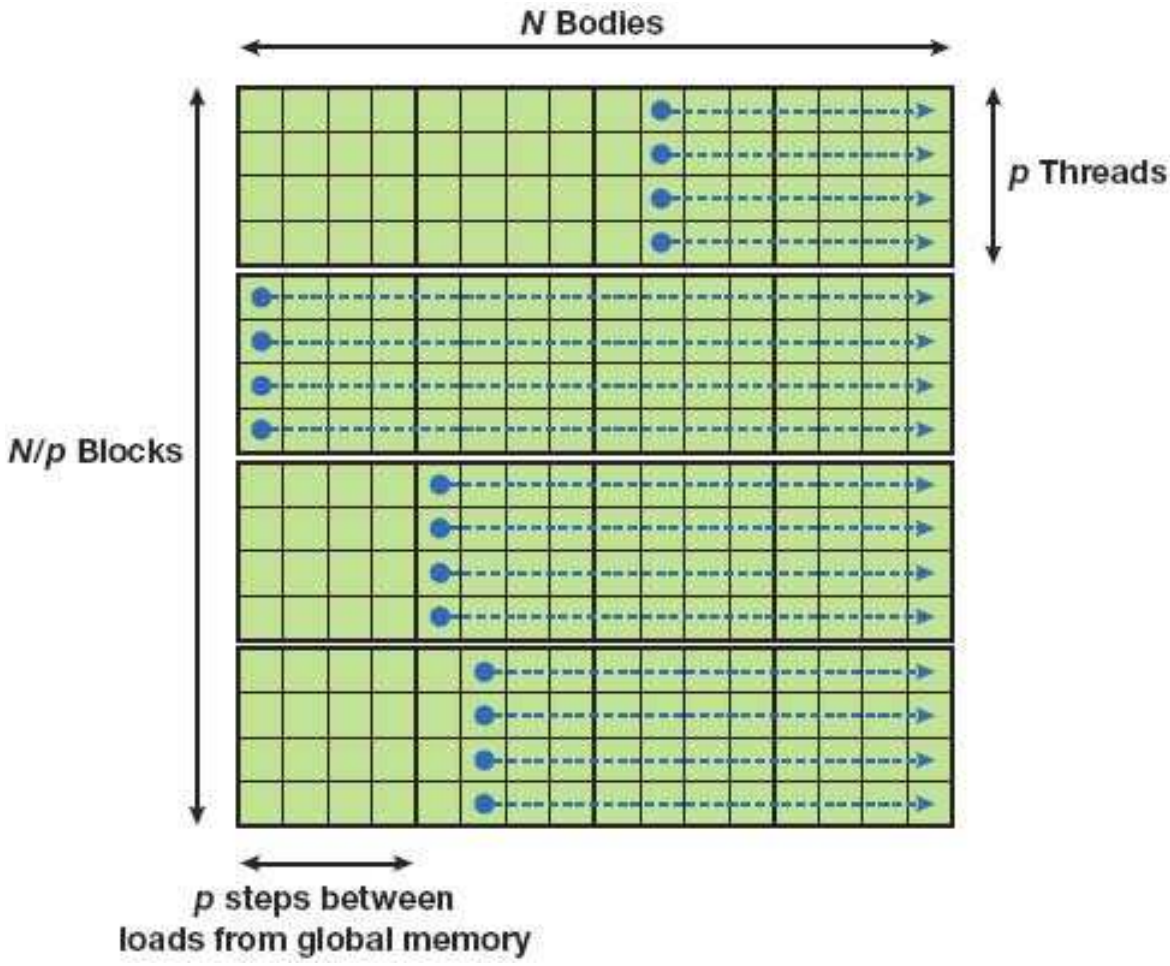


Фигура 4.2: Кооперация между нишките на блок [8].

Блоковете се стартират в решетка. За нашата цел е достатъчно решетката да бъде едномерна, но ако максималната големина на едно измерение не е достатъчна, лесно може да се представи двумерена решетка като едномерна, чрез трансформацията $blockID = blockIdx.x + blockIdx.y * gridDim.x$. Нека имаме N ЕО с електрони и големината на блока е p. Тогава големината на решетката ще бъде N/p. Фиг. 4.3 показва еволюцията на нишките в решетката.

4.2.2 Мултиплициране на електроните

Паралелното мултиплициране на електроните на всяка стъпка се извършва от напълно независими нишки. Всяка нишка мултиплицира серийно електроните от един ЕО, като итерира по всеки от тях и прилага аналогичен на CPU алгоритъма от секция 4.1 с основната разлика, че вместо двусвързан списък сега еволюцията им се записва чрез атомарни операции в глобалната памет на GPU, където са разположени три масива съответно за електрони, положителни и отрицателни йони и обхваща целият обем на газа. Всяка една от нишките се нуждае от генератор на равномерно разпределени случайни числа и е изключително важно между тези числа да няма корелация. Как се постига това ще бъде разгледано в секция 4.2.4. Алгоритъмът, определящ как се стартира и изпълнява ядровата функция на всяка стъпка от развитието на лавината е комп-



Фигура 4.3: Еволюция на нишките от решетката [8].

лициран, поради необходимостта от балансиране на следните фактори: минималното количество електрони, което всяка нишка трябва да обработи, за да се постигне задоволителна независимост при генерирането на псевдо случайни числа, определящи еволюцията на лавината; големината на решетката и големината на блоковете в нея, определящи заетостта и ефективността на GPU мултипроцесорите и прозрачността на скалируемост при следващите поколения GPU; максималният брой на нишки или максималният брой на псевдо генераторите на случайни числа (ПГСЧ), балансиращ нивото на корелация и степента на паралелизъм. Определянето на посочените фактори е най-добре да се прави ръчно, за постигане на максимална производителност за конкретен GPU. В разглежданата имплементация имаме входни параметри:

- максимален брой на ПГСЧ - maxNumOfRNG
- големината на блока в решетката - blockSize
- минималният брой блокове за обработка, които трябва да достигне всяка нишка преди да се добави нова нишка - minNumOfBlocksPerThs

и за конкретност точния откъс от кода изчисляващ големината на решетката gridXSizeK2 е

```
int currNumOfThs = coutElsCells / minNumOfBlocksPerThs + 1;
if (currNumOfThs > maxNumOfRNG) currNumOfThs = maxNumOfRNG;
sp_h.numCellsPerThread = coutElsCells%currNumOfThs == 0 ?
    coutElsCells/currNumOfThs : coutElsCells/currNumOfThs + 1;
sp_h.numThreads = coutElsCells % sp_h.numCellsPerThread == 0 ?
    coutElsCells / sp_h.numCellsPerThread :
    coutElsCells / sp_h.numCellsPerThread + 1 ;
gridXSizeK2 =  sp_h.numThreads % blockSizeK2 == 0 ?
    sp_h.numThreads  / blockSizeK2 : sp_h.numThreads  / blockSizeK2 + 1;
```

където sp_h е структура, която се копира в паметта на GPU и служи за предаване на параметри. Ядровата функция се извиква чрез израза

```
dim3 threads(blockSizeK2,1,1); dim3 grid(gridXSizeK2,1, 1);
mulEls2<<<grid, threads>>>();
```

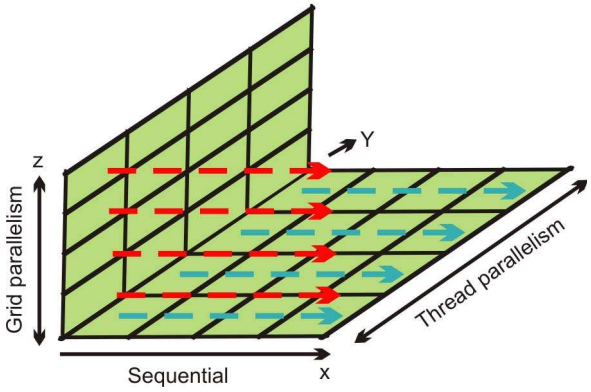
Нейното представяне частично в псевдокод за сбитост е следното

```
__global__ void mulEls2()
{
    int tIDth = __mul24(blockIdx.x , blockDim.x) + threadIdx.x;
    //Всички тредове с идентификатори по-малки от нужния брой нишки ще изпълнят
    ядровата функция
    if (tIDth < sp_d.numThreads){

    // Зареждне на текущото състояние на Mersenne Twister ПГСЧ
    mt_struct_stripped config = bp_d.MT_d[tIDth]; int iState = bp_d.iState_d[tIDth];
    unsigned int mt[MT_NN];
    for(int i = 0; i < MT_NN; i++) mt[i] = bp_d.mt_state_d[tIDth*MT_NN + i];
```

```
    //Стартира се цикъл по всяка от клетките, за която нишката е отговорна
    for(int cells=0; cells < sp_d.numCellsPerThread; cells++){

    //Изчислява номера на клетката в масива получен след конвертиране
    int tID = __mul24(__mul24(blockIdx.x , blockDim.x) +
                threadIdx.x, sp_d.numCellsPerThread ) + cells;

    //Зареждане информация за клетка от глобалната памет в локалната на мултипроцесора
    int4 pos_s = (int4) sp_d.pos_d[tID];

    //Ако в клетката има електрони и не е гранична на анода на камерата (обема на газа)
    if ( pos_s.w < 0 && pos_s.z < (bp_d.dimZ - 1) ) {

    //Дефиниране на необходими локални променливи
    ..............................................
    //Извиква се атомарно изваждане, което изважда броя на електроните от разглеждания
    обем от кубичният масива описващ електроните в обема на газа
    atomicSub(&bp_d.numEls_d[P(pos_s.x,pos_s.y,pos_s.z)], abs(pos_s.w));

    //Зареждане в локалната памет от глобалната на необходимите данни за съответния ЕО
    ..............................................
    //В зависимост от стойностите на коефициента на захващане и йонизация се използва
    съответното разпределение за мултиплициране на електрон, прилагайки се алгоритъмът
    описан в 3.2 и записвайки еволюцията чрез атомарни операции в съответните масиви
    - bp_d.numIons_d, bp_d.numEls_d, bp_d.numNegIons_d
    ..............................................
    }
    }
    // Запазване на текущото състояние на Mersenne Twister ПГСЧ в глобалната памет
    }
}
```

4.2.3 Намиране на лавината в обема

След мултиплициране, лавината се описва чрез трите масива: bp_d.numIons_d, bp_d.numEls_d, bp_d.numNegIons_d. За ускоряване конвертирането на представянето на лавината, което се извършва от CPU е съставена ядрова функция, която намира минималният информативен правоъгълен паралелепипед, чрез двата му ръба - най-близкия и най-далечния, мерено от центъра на координатната система. По този начин, тройният цикъл не проверява целия обем на газа за заряди, а само паралелепипеда и получаваме ускорение $O(N^3)$. На Фиг. 4.4 е илюстрирано разпаралелването на задачата по нишки. Всяка нишка използва атомарни операции - atomicMin и atomicMax, за да осъвремени съответните променливи, намиращи се в глобалната памет на GPU и описващи ръбовете на паралелепипеда. Със съвсем малко допълнение към кода, без да е необходимо структурно изменение, можем да намерим големината на различните типове натрупан заряд и паралелепипедите, които го съдържат. За по-нататъшна конкретизация е най-ефективно да се погледне сорс кода.



Фигура 4.4: Паралелизъм при намиране на лавината в газа

4.2.4 Паралелен генератор на случайни числа

Най-основната част от Монте-Карло симулация, определяща правилността ѝ и точността ѝ е използваният генератор на случайни числа. Най-добре би било да използваме физически (хардуерен) генератор, но с такъв понастоящем NVIDIA GPU не разполага, нито пък изглежда ефективно предварително да генерираме случайни числа, да ги разполагаме в глобалната памет на GPU, която е относително по-малко и по-скъпа и да ги използваме наготово. Евтина алтернатива на физически генератор, е висококачествен добре апроксимиращ математически ПГСЧ . Съществуват немного всепризнати различни методи за серийно генериране на псевдо СЧ, които обикновено се задават чрез рекурсивно уравнение, определящо се от параметри и инициализиращо се с начално състояние наречено семе ("seed"). С настъпването на разпределените изчисления произлиза необходимостта от паралелен генератор, характеризиращ се с следните свойства:

- Желателна никаква корелация между ПРСЧ генерирани от всяка изчислителна единица.
- Задоволително случайно разпределение (нерегулярност) с максимализиран период на повтаряемост.
- За ефективна и скалируема генерация е необходимо ефективност на генератора на всяка от изчислителните единици и минималност на комуникацията между тях, така че да могат да ескалират до задоволителни количества.

- Желателно е, при едно и също стартово състояние, да има възпроизвеждане на една и съща последователност от СЧ, което улеснява сравняването на резултати и откриването на грешки

Има различни подходи при направата на паралелен генератор, всеки със своите предимства и недостатъци. За симулацията се използва Mersenne Twister (MT) [1], защото е един от най-добрите серийни генератори - голям период $2^{19937} - 1$, добро разпределение на генерираните СЧ, висока ефективност и може да се паралелизира директно и ефективно върху CUDA програмен модел. MT е итеративен, което прави трудно разпаралелването му, особено на масивно паралелни архитектури, защото всички нишки трябва да обновяват текущото състояние на генератора. Най-директното и лесно решение на този проблем е просто да използваме толкова различни MT, колкото нишки пускаме. Така всяка нишка ще обновява различно състояние и може да се постигне пълен реален паралелелизъм. Но тъй като, променянето на стартовото състояние на MT или произволно променяне на параметрите не прави два MT некорелирани, то е специално разработена отделна библиотека "dcmt"от създателите на MT, която генерира параметри за MT, така, че всеки два MT задоволително да не корелират и да имат голям период. Използваната версия 0.4 дава възможност за създаване до 2^{16} различни 32 битови MT. Смисъла на "различни"е дефиниран в [2]. Единственият недостатък на този подход е, че динамичното генериране на пареметрите с dcmt може да отнеме значително време и затова се прави отделно самостоятелно, като данните се записват в файл, които след това само се зареждат от програмата, на която са нужни.

В нашия случай първо се зареждат параметрите за 2^{16} MT от файл в глобалната памет на GPU. Второ, генератор Г1, чието стартовото състояние задаваме директно и е едно естествено число, генерира 2^{16} стартови състояния за втори генератор Г2, който инициализира всеки MT. Така за повтаряемост е необходимо само да инициализираме Г1 с едно и също стартово състояние. За повишаване на скоростта тази операция се извършва от ядрова функция, която се изпълнява от 2^{16} нишки, всеки от който инициализира по един MT използвайки Г2. Така реално разполагаме с 2^{16} MT, но с цел изследване и оптимизация максималния брой използвани MT е оставен да бъде входен параметър на симулацията.

За преобразуване на равномерно разпределените случайни числа в Гаусово разпределени, се използва бърза, без цикли, добре апроксимираща трансформация, която превръща всяко генерирано равномерно разпределено число в Гусово разпределено [6].

4.2.5 Цялостно изложение на алгоритъма

С оглед да се акцентира структурата и избегне ненужната утежняваща конкретизация, алгоритъма е даден в по-описателен стил. Желаещите да разберат точната имплементация е най-удачно да се обърнат към кода.

1. Определяме входните параметри: g големината на процепа; E_0 основното поле; N дискретизация по-всяко от измеренията; dimX, dimY, dimZ големината на масивите, описващи зарядите в обема на газа съответно по трите измерения; maxNumOfRNG- максималния брой на MT, които ще се използват; minNumOfBlocksPerThs - минимално количество ЕО, което трябва да се достигне за орбаботка от всяка нишка преди да се добави нова, ако текущия им общ брой е по-малък от maxNumOfRNG; blockSizeK2 - големината на блока в решетката при изпълнение на ядровата функция мултиплицираща всеки електрон mulEls; зависимостта на параметрите на газа υ_D, D_L, D_T, α и η от $\vec{E}$ се изчислява предварително с програмата Magboltz, записват се във файл и се зарежда в глобалната памет на GPU.

2. Инициализират се основните параметри на задачата : $dx = dy = dz = g/dimZ$ определят големината на елементарния обем; $dt = dz/\upsilon_D(|\vec{E}_0|)$ стъпката по времето на симулацията. Резервира се памет за основните масиви: numIons, numEls, numNegIons. Зареждат се и се инициализират MT в глобалната памет на GPU. Основните параметри на задачата се описват чрез структурата bp, която се копира в ГП на GPU. Параметрите, които се обновяват на всяка стъпка се описват чрез структурата sp.

3. Инициализира се обема на газа numIons, numEls, numNegIons като свободен от заряди. Поставя се началното количество заряд в numEls

4. Извиква се процедура convertArray за конвертиране на numIons, numEls, numNegIons в един масив pos използван по-късно от clacElsInter по начин описан в секция 4.2.1

   (a) convertArray извиква ядровата функция за намиране на лавината в обема и нейното съдържание, необходимо за оразмеряването на pos и определяне на blockSizeK1 - големината на блока в решетката при изпълнение на ядровата функция clacElsInter пресмятаща полето върху електроните, както е описано в 4.2.3

5. Ако лавината е изгаснала или всички електрони са напуснали обема на газа, то или преминаваме към стъпка 2 за нов опит или приключваме изпълнението на програмата.

6. Инициализиране на параметрите необходими за текущата стъпка в структурата sp

7. Стартира се clacElsInter и след това mulEls. Копира се резултата от ГП на GPU в паметта на CPU и се преминава към стъпка 4 за пресмятане на следващата стъпка.

За визуализация на лавината е използван OpenGL. За по-добра производителност е необходимо не само да не се пуска с визуализация, но трябва графичната среда на компютъра да е спряна. Под Linux, X сървъра заема GPU ресурсите и дори е възможно, симулацията изведнъж да спре поради липса на ресурси.

4.2.6 Възможни оптимизации

Оптимизациите могат да се класифицират в различни типове и тук се разглеждат в отделни параграфи.

Подобряването на производителността или ефективността може да стане най-лесно, чрез компилационни опции на компилатора, чрез развиване на цикли (посочени с директива на компилатора), подбиране на големината на решетката и неговите блокове специфично за приложението и даден GPU, използване на хардуерно имплементирани основни математически операции, дори и те да са с намалена точност.

На следваща стъпка е възможно кода да може да се съкрати за сметка на нагледността му или частично, или напълно да се модифицира така, че да използва възможностите на по-нова версия на CUDA. Например, използването на атомарни операции, работещи върху споделената памет, позволява на нишките от блоковете да кооперират и през нея, които иначе трябва да кооперират само през глобалната памет. Това може да доведе до повишаване на производителността, при условие, че ограничаването на производителността идва от забавяне на достъпа до глобалната памет. Необходимо е специализирано изследване за конкретно използваното GPU, за да се определи дали това е така.

Друга възможна оптимизация е да се измисли по-оптимален алгоритъм, което често означава и по-труден за имплементиране и дебъгване. Специално за разглежданата симулация би могло да се помисли, как вместо на всяка стъпка да търсим цялата лавина, в целия обем на газа, да търсим само променената част от нея и то в по-малък обем,

който задоволително апроксимира целия обем. Това би облекчило трафика между GPU и CPU и би ускорило процедурата convertArray. Друго подобрение би било convertArray да се изпълнява на GPU, където трансфера на данни до глобалната му памет е с порядък по-висок. По този начин почти напълно ще облекчим CPU, но записването на контролни данни, като моментна снимка на лавината, би се затруднило и ще води до забавяне.

По принцип, за постигане на максимално ефективен HPC алгоритъм практиката показва, че трябва тясно да се специализира и оптимизира за конкретна задача, за съответната HPC архитектура, върху която ще се изпълнява. Добавянето на съвсем малка на вид функционалност, разширяваща модела на задачата, може да му намали производителността и да се нуждае от преработка. Разработената симулация е направена така, че да се изпълнява ефективно за широк спектър от постановки и начални условия на задачи, и автоматично производителността ѝ да ескалира с настъпването на по-нови GPU. Входните параметри на симулацията, касаещи конфигурациите за изпълнение на ядровите функции са именно за допълнителна фина настройка на производителността, без да е необходимо да се модифицира кода (Фиг. 5.8).

Когато се появи нова архитектура, каквато е CUDA понастоящем, най-обсъждан е въпросът около предлаганото ускорение. За коректност, винаги е необходимо да се посочва какво точно се сравнява и на каква база.

Глава 5

Резултати

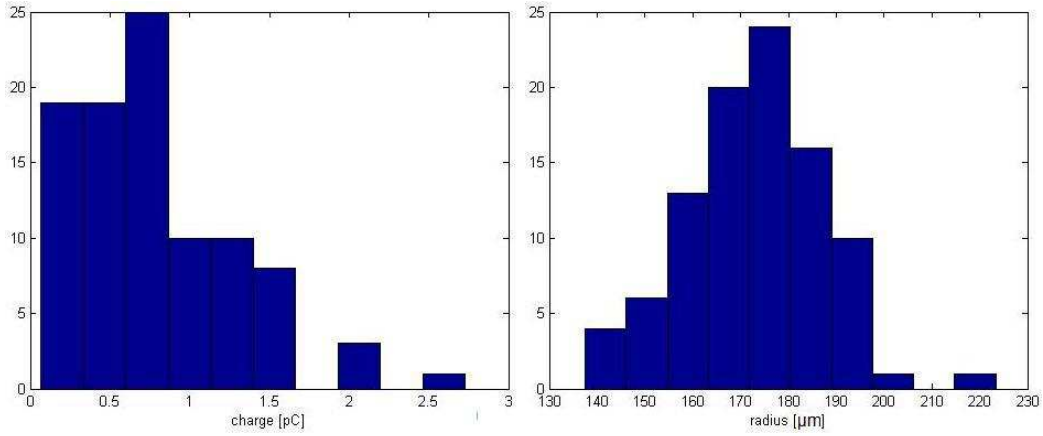
В тази част са показани резултати от симулацията за газовата смес $C_2F_4H_2(96.2\%)$, $iso\text{-}C_4H_{10}(3.5\%), SF_6(0.3\%)$ и различни стойности на основното поле $\vec{E}_0$, големината на обема на газа и честота на дискретизация. При числените експерименти основен показател за правилността на изпълнението му са адекватните резултати, които най-добре се виждат графично изобразени. За оценяване на ускорението ще бъде направено сравнение на производителността за различни стойности на параметрите, определящи изпълнението на ядровите функции.

Тъй като, се прави Монте-Карло симулация, то за различни инициализиращи параметри на генератора на случайни числа и различни параметри на симулацията се получават различни лавини. За това е необходимо построяването на хистограми, показващи разпределението на даден параметър от множество симулации на лавини. Колкото повече отделни симулации участват в построяване на хистограмата, толкова поточна ще бъде тя. Но в нашия случай, тъй като се разполага с ограничени ресурси и ограничено време са направени хистограми за различни параметри на лавината с 100 лавини, което очертава тяхното разпределение и доказва, че симулацията се извършва правилно. Следващите резултати, са получени от симулацията само на една лавина и не представлява трудност да се направят произволно точни хистограми, при наличност на ресурси.

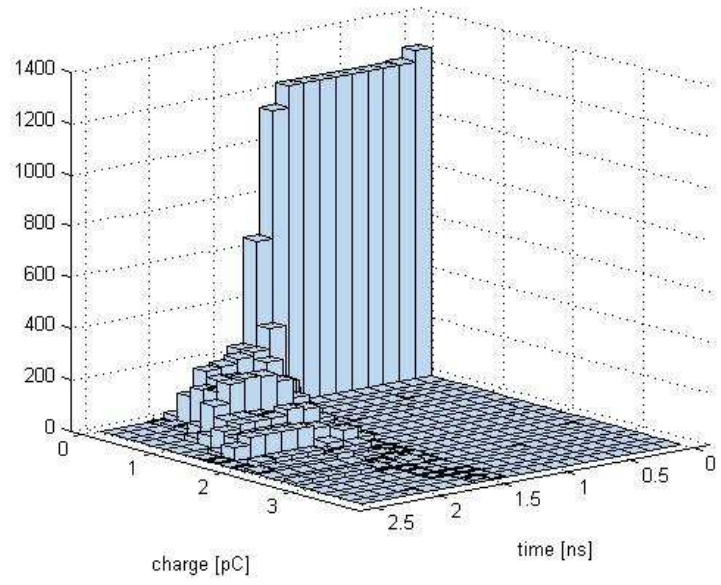
5.1 Лавини при $\vec{E} = 75kV/cm$ и $g = 250\mu$

На следващите фигури Фиг. 5.1 и Фиг. 5.2 са показани хистограми резултат от симулацията на 100 лавини при $\vec{E} = 75\text{kV/cm}$ и големина на процепа $g = 250\mu$. Пет от лавините са изгаснали преди да достигнат

анода. На Фиг. 5.3 се вижда разпределението на електроните натрупали

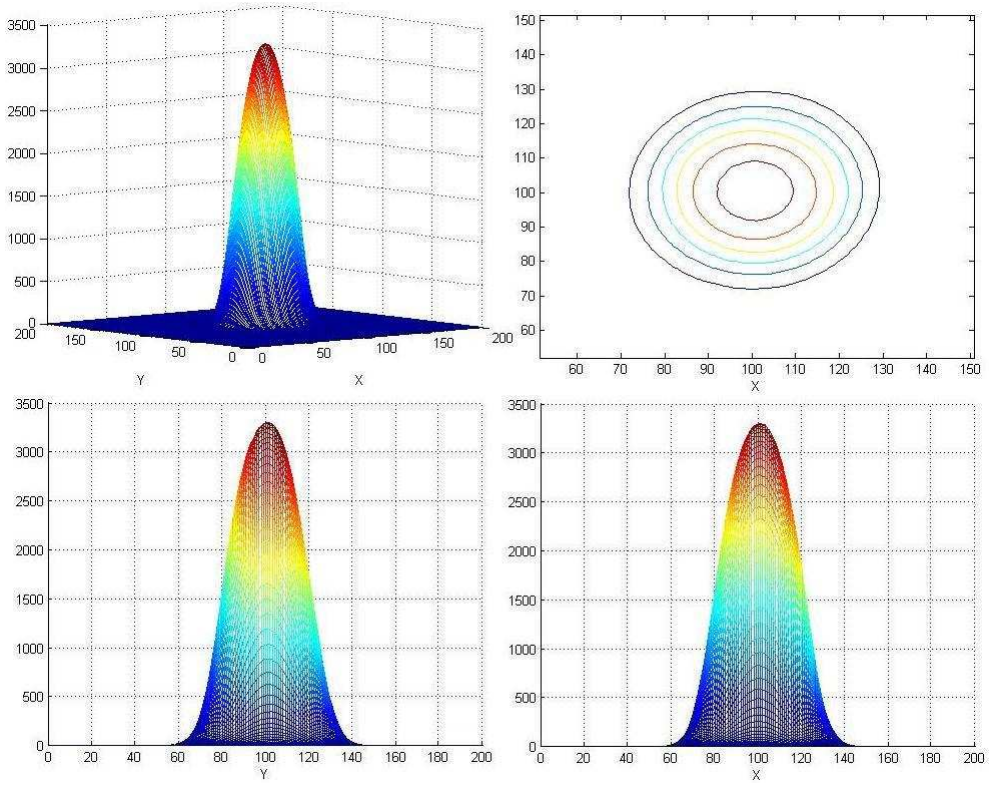


Фигура 5.1: Разпределение на електронния заряд в ляво и на радиуса в дясно на 100 лавини



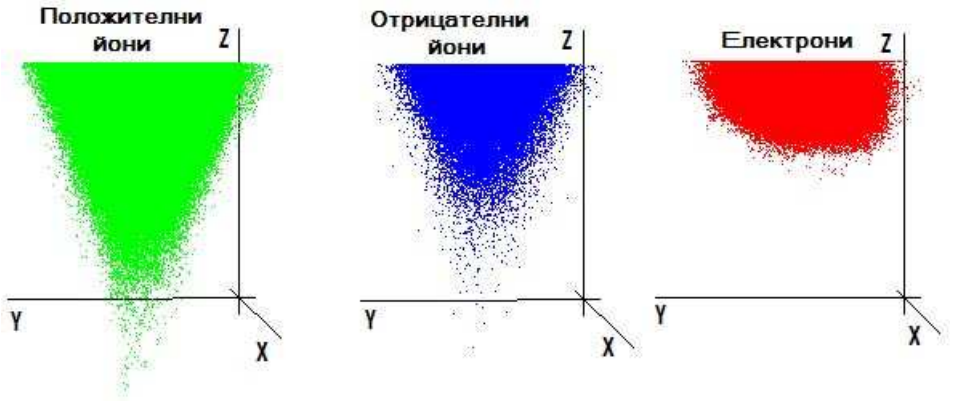
Фигура 5.2: Разпределение на електронния заряд по времето на 100 лавини

се на анода, което е нормално и е силен аргумент, че симулацията на индивидуална лавина е успешно.



Фигура 5.3: Разпределение на електронния заряд на 100 лавини върху равнината XY

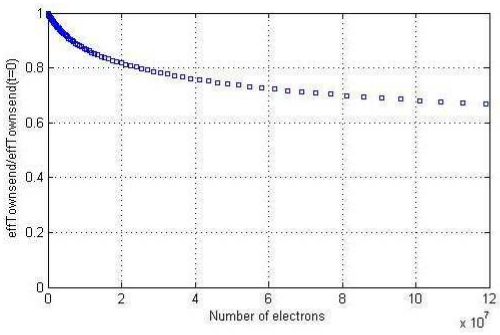
На Фиг. 5.4 най-отляво и в средата е показана снимка, съответно на положителните и отрицателните йони след напускане на всички електрони от обема на газа. Най-отдясно е показа моментна снимка от развитието на лавината, в която част от електроните са достигнали анода.



Фигура 5.4: Най-отляво и в средата е показана снимка съответно на положителните и отрицателните йони след напускане на всички електрони от обема на газа. Най-отдясно е показа моментна снимка от развитието на лавината, в която част от електроните са достигнали анода.

5.2 Лавини при $\vec{E} = 48kV/cm$ и $g = 3mm$

На Фиг. 5.5 е показан претеглено усреднения, ефективен коефициент на Таунсенд. Пресмята се, чрез усредняване по-всички елементарни обеми, заети от електрони и претеглен, съответно по броя на електроните в тях. Сходството между двете графики на Фиг. 5.5, показва коректността на симулацията. Не е необходимо двете графики да съвпадат, тъй като симулацията я правим при подобна обстановка. На Фиг. 5.6 е показано как електричният заряд се развива в лавината. Пиковата стойност се достига при достигане на анода от първия електрон от лавината. След това зарядът започва да намалява, защото електроните, достигнали анода не се мултиплицират повече и само се взема под внимание полето, което те създават върху останалите. На Фиг. 5.7 е показано разпределение на електроните върху анода, забелязва се, че при увеличаване на честотата на дискретизация клони към нормално разпределение.



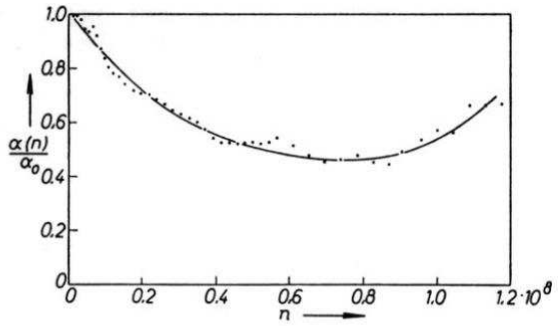
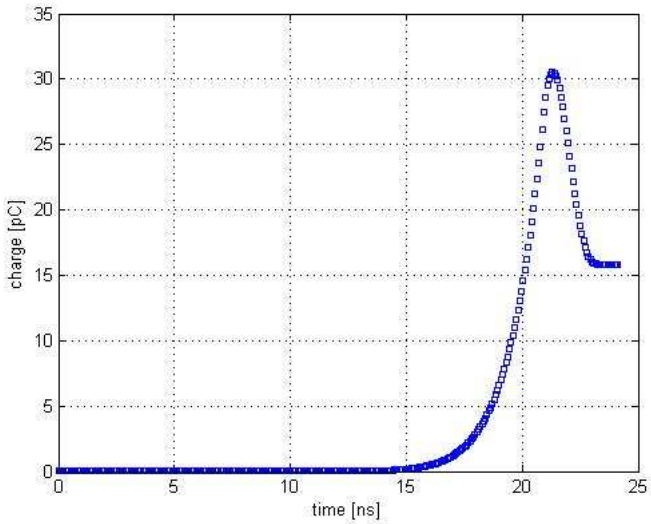
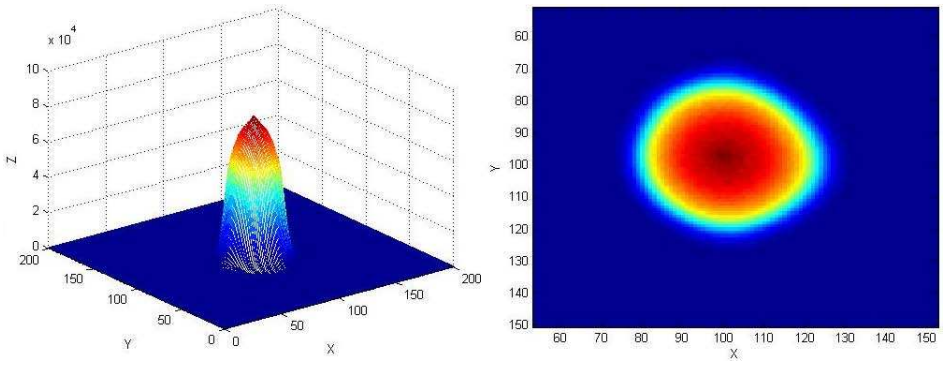


Abb. 6. Durch Raumladung geschwächter Ionisationskoeffizient $\alpha(n)$. Ermittelt aus einer Statistik in Methylal nach Schlumbohm [8] (Versuchsdaten im Text).

Фигура 5.5: В ляво е показан претеглено усредненият ефективен коефициент на Таунсенд, а от дясно е експериментално измерен [3].



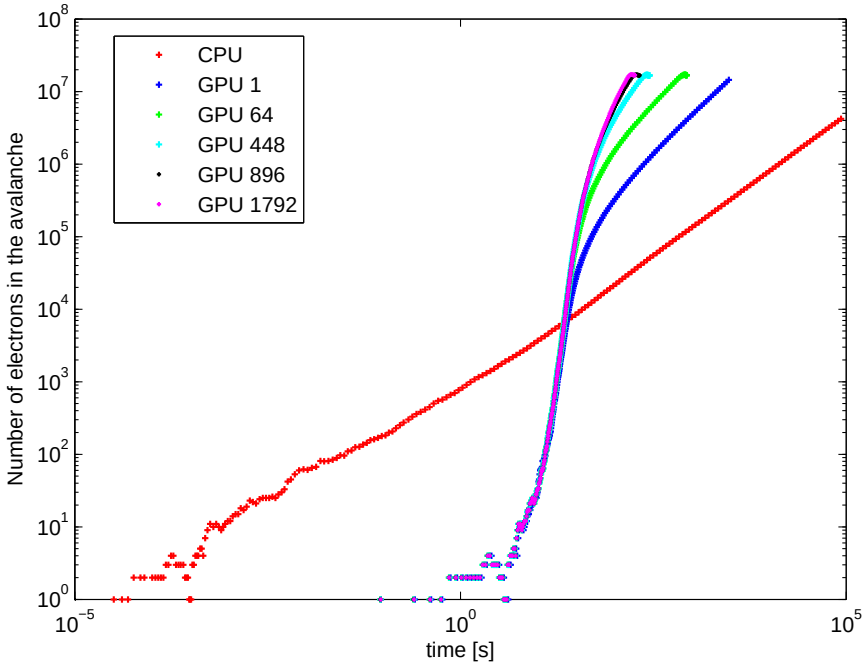
Фигура 5.6: Развитие на електричния заряд на лавината с времето



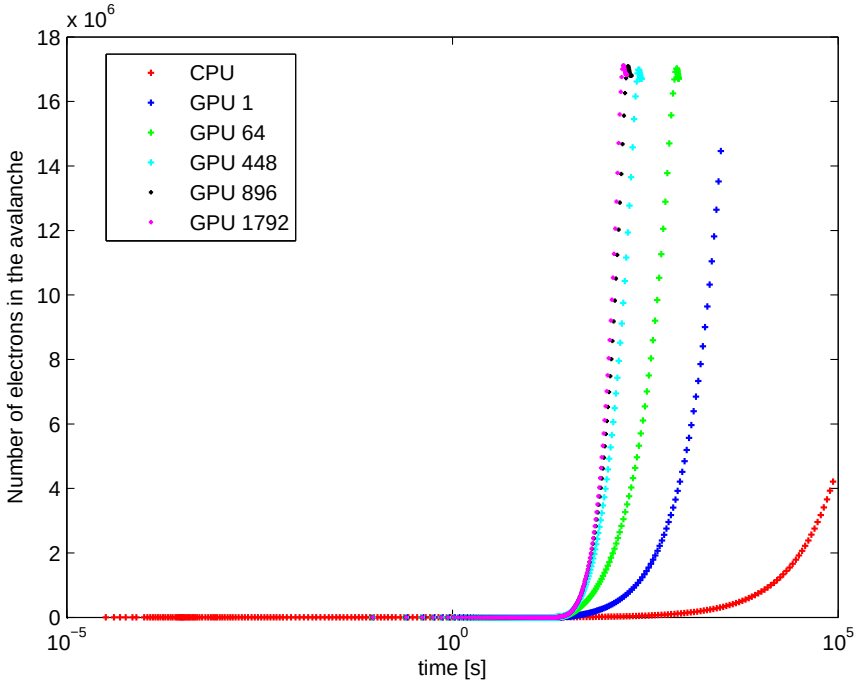
Фигура 5.7: Разпределение на електроните върху анода

5.3 Сравнение на производителностите на GPU с CPU

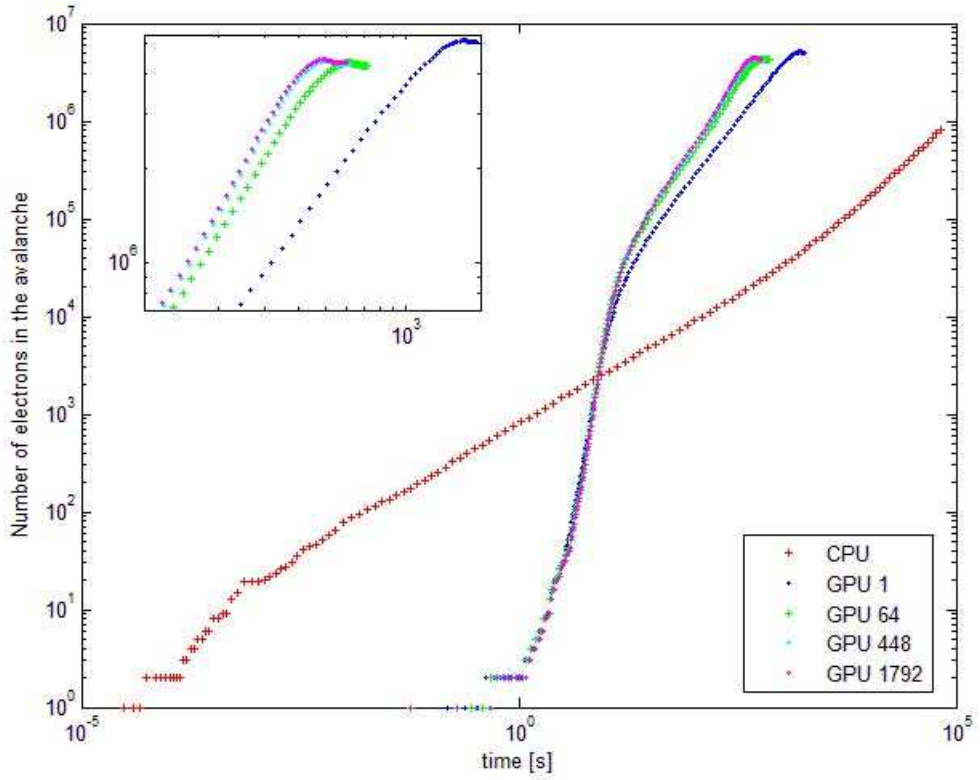
На Фиг. 5.8 и Фиг. 5.9 е изобразено времето необходимо за симулиране на електронна лавина, използвайки CUDA GPU и само CPU при $\vec{E} = 47kV/cm$, $g = 3mm$ и $dz = 12\mu$. Получава се, че за различна стойност на параметъра maxNumOfRNG - за различни конфигурации, стартиращи ядровата функция, мултиплицираща електроните, се получават различни криви клонящи към една, показваща максималното ускорение, което може да се постигне. Фиг. 5.10 е аналогична, но при $\vec{E} = 75kV/cm$, $g = 250\mu$ и $dz = 1.6\mu$. Тези резултати могат да се обосноват теоретично.



Фигура 5.8: Времето необходимо за симулиране на електронна лавина използвайки CUDA GPU и само CPU при $\vec{E} = 47kV/cm$, $g = 3mm$ и $dz = 12\mu$, за различни стойности на maxNumOfRNG - за различни конфигурации, стартиращи ядровата функция мултиплицираща електроните.



Фигура 5.9: Времето необходимо за симулиране на електронна лавина използвайки CUDA GPU и само CPU при $\vec{E} = 47kV/cm$, $g = 3mm$ и $dz = 12\mu$, за различни стойности на maxNumOfRNG - за различни конфигурации, стартиращи ядровата функция мултиплицираща електроните.



Фигура 5.10: Времето необходимо за симулиране на електронна лавина използвайки CUDA GPU и само CPU при $\vec{E} = 75kV/cm$, $g = 250\mu$ и $dz = 1.6\mu$, за различни стойности на maxNumOfRNG - за различни конфигурации, стартиращи ядровата функция мултиплицираща електроните.

Глава 6

Заключение

В дипломната работа бяха разработени алгоритми за паралелизации на изчисления върху видео карта. Бе обстойно изучена GPU архитектура, поддържаща NVIDIA CUDA, съответния CUDA програмен модел и иновативните възможности, които CUDA предлага за HPC. Беше разработен основен хетерогенен алгоритъм за ускоряване, чрез паралелизация, на симулации на лавинни процеси. Като пример, бе разработена симулация на развитие на електронна лавина в газова среда и поспециализирано в камера със съпротивителна плоскост. Примерната симулация може да служи като основен модел за разработване на симулации на лавинни процеси в по-общ смисъл. Това обстоятелство, прави разработения алгоритъм с широк спектър на приложение, последващо развитие и внедряване. Най-основният обобщаващ принципен резултат от тази дипломна работа, е доказателството, че GPU вече могат да се използват за ускоряване на алгоритми с общо предназначение. Полученото ускорение силно зависи от конкретната задача и варира от един до няколко порядъка. След няколко години няма да има сериозен алгоритъм или програма, която да не е разпаралелена и дори да не е хетерогенна. Тази дипломна работа допринесе за овладяването и адаптацията на GPU и показва бъдещия стил на програмиране.

Библиография

[1] M.Matsumoto, T.Nishimura “Mersenne Twister. A 623-dimensionally equidistributed uniform pseudorandom number generator.“, ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, pp.3-30 (1998) DOI:10.1145/272991.272995

[2] M.Matsumoto, T.Nishimura “Dynamic Creation of Pseudorandom Number Generators.“, Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer, 2000, pp 56–69

[3] W. Legler. “Die Statistik der Elektronenlawinen in elektronegativen Gasen, bei hohen FeldstËarken und bei hoher GasverstËarkung (the statistics of electron avalanches in electronegative gases at high electric field strengths and at large gas gain).“ Z.Naturforschung, 16a:253–261, 1961.

[4] W. Riegler, R. Veenhof, and C. Lippmann. “Simulation of resistive plate chambers.“ CERN-EP-2002-046, 2002. Nucl. Instr. Meth. A.

[5] L. Christian “Simulation of Space Charge Effects and Induced Signals in Resistive Plate Chambers“, Doctoral Thesis, `http://www-linux.gsi.de/~lippmann/files/Dissertation.pdf`

[6] M. Abramowitz, A. Stegun, “Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables“, Dover Publications, 1965 (originally published by the National Bureau of Standards, 1964)

[7] NVIDIA “NVIDIA CUDA Compute Unified Device Architecture Programming Guide & CUDA Education“, `http://www.nvidia.com/cuda`

[8] H. Nguyen “GPU Gems 3“, Addison Wesley Professional, 2007

[9] The Compact Muon Solenoid, Technical Proposal, CERN/LHCC 94-38, 1994, `http://cmsdoc.cern.ch/TPref/TP.html`

[10] M. Abbrescia et al. "Effect of the linseed oilnext term surface treatment on the performance of resistive plate chambers", NIM A 394 (1997) 13

[11] F. Sauli, "Principles of operation of wultiwire proportional and drift cahmbers", 1977, CERN 77-09

[12] S. Biagi, IMONTE, програма за изчисляване на транспортните параметри на газа

[13] W. Riegler and C. Lippmann, "The physics of Resistive Plate Chambers", Nuclear Instruments and Methods in Physics Research A 518 (2004) 86-90

[14] W. Riegler et al. "Detector physics and simulation of resistive plate chambers", NIM A , Volume 500, Issues 1-3, 11 March 2003, Pages 144-162

[15] H. Genz, "Single electron detection in proportional gas counters", Nucl. Instr. and Meth. 112 (1973) 83

[16] M. Abbrescia et al., "The simulation of resistive plate chambers in avalanche mode: charge spectra and efficiency", Nuclear Instruments and Methods in Physics Research A 431 (1999) 413-427

[17] Igor Smirnov, HEED, program to compute energy loss of fast particles in gases, Version 1.01, CERN.

[18] F. Rieke, W. Prepejchal, "Ionization Cross Sections of Gaseous Atoms and Molecules for High-Energy Electrons and Positrons", Phys. Rev. A 6 (1972) 1507-1519

[19] M. Abbrescia et al. "Properties of C_2H_2F-4-based gas mixture for avalancehe mode operation of resistive plate chambers", NIM A 398 (1997) 173-179,

[20] W. Blum, G. Rolandi, "Particle Detection with Drift Chambers", Springer, Berlin, 1993

[21] S. Ramo, "Current induced in electron motion", PROC. IRE 27 (1939), 584

[22] E. Gatti et al., "Signal evaluation in multielectrode radiation detectors by means of a time dependent weighting vector", NIM 193 (1982), 651-627

[23] W. Riegler, "Induced signals in resistive plate chambers", NIM A 491 (2002) 258271

[24] B. Schnizer et al. "Simple models for RPC weighting fields and potentials", NIM A 535 (2004) 554557

[25] S. Biagi, MAGBOLTZ, програма за изчисляване на транспортните параметри на газа, CERN.

[26] Eduardo Gorini, "Measurements of drift velocity and amplification coefficient in C2H2F4-isobutane mixtures for avalanche operated RPC", Fourth International Workshop on Resistive Plate Chambers and Related Detectors, Napoli, October 1997.

[27] R. Santonico and R. Cardarelli, "Development of Resistive Plate Counters", NIM 187 (1981) 377-380

[28] R. Arnaldi et al. "Study of resistive plate chambers for the ALICE dimuon spectrometer". Nucl. Phys. B - Proc. Suppl., 78:8489, 1999.

[29] C. Bacci et al. "High altitude test of RPCs for the ARGO YBJ experiment". Nucl. Instr. Meth., A 443:342350, 2000.

[30] C. Bacci et al. "A hodoscope made of resistive plate chambers to identify muons in a fixed targed beauty hadroproduction experiment", Nucl. Instr. Meth., A 324:83-92, 1993.

[31] P. Fonte, V. Peskov, and B.D. Ramsey. "Streamers in MSGC's and other gaseous detectors", ICFA instrumentation bulletin, Fall 1997, (SLAC-PUB-7718, SLAC-JOURNAL-ICFA-15), `www.slac.stanford.edu/pubs/icfa/fall97/paper1/paper1.pdf`

[32] ATLAS muon spectrometer technical design report. CERN LHCC-97-22, ATLAS TDR 10, CERN, 1997.

[33] ALICE TOF - Time-of-flight technical design report. CERN/LHCC 2000-012, ALICE TDR 8, 2000.

[34] M. Bogomilov et al. "The RPC time-of-flight system of the HARP experiment", presented at the 'RPC 2001' 6th Workshop on Resistive Plate Chambers and Related Detectors, 26-27 November 2001, Coimbra, Portugal.

[35] M. Bogomilov et al. "The HARP RPC time-of-flight system", Nuclear Instruments and Methods A 508 (2003) 152-158

[36] A. Akindinov et al. "The multigap resistive plate chamber as a time-of-flight detector", Nucl. Instr. Meth., A 456:16-22, 2000.

[37] Cerron Zeballos, "A new type of resistive plate chamber: The multigap RPC", Nuclear Instruments and Methods in Physics Research A 374 (1996) 132-135; CERN PPE/95-166; CERN/LAA-MC 95-23

[38] P. Fonte et al., "A new high-resolution TOF technology", NIM A 443 (2000) 201-204, CERN-EP/99-68

[39] Addendum to the ALICE TOF - Time-of-flight technical design report. CERN/LHCC 2002-016, 2002.

[40] The HARP collaboration. Proposal for an RPC TOF system. 2000. Available:

[41] P. Fonte et al. "High resolution RPCs for large TOF systems", Nucl. Instr. and

[42] M.C.S. Williams et al. "The multigap RPC : The time-of-flight detector for the ALICE experiment", Nucl. Instr. Meth., A 478:183-186, 2002.

[43] I. Crotty et al., "The non-spark mode and high rate operation of resistive parallel plate chambers", Nucl. Instr. and Meth. A 337 (1994)370

[44] Cardarelli et al. "Progress in resistive plate counters", Nuclear Instruments and Methods A 263 (1988) 20-25

[45] M. Abbrescia et al. "Cosmic ray tests of double-gap resistive plate chambers for the CMS experiment" NIM A 550 (2005) 116-126

[46] A. Zichichi, "The LAA Project", CERN-EP/87-122, 1987

[47] E. Cerron Zeballos et al. "Comparison of the wide gap and narrow gap resistive plate chamber", Nucl.Instrum.Meth. A 373 (1996) 35-42 ; CERN-PPE-95-146; CERN-LAA-MC-95-21

[48] R. Cardarelli, A. Di Caiccio, and R. Santonico. "Performance of a resistive plate chamber operating with pure CF3Br", Nucl. Instr. Meth., A 333:399B403, 1993.

[49] I. Duerdoth et al., "The transition from proportional to streamer mode in a resistive plate chamber", Nucl. Instr. and Meth. A 348 (1994) 303

[50] P. Camarri et al., "Streamer Suppression with SF_6 in RPCs Operated in Avalanche Mode", NIM A414 pp. 317-324, 1998

[51] V.Ammosov et al., "Study of SF_6 addition influence on narrow gap RPC performnace", IHEP 99-53

[52] E. Ceron Zeballos et al. "Resistive plate chambers with secondary electron emitters and microstrip readout", Nuclear Instruments and Methods in Physics Research A 392 (1997) 15O-154

[53] I. Crotty et al. "High-rate, high-positionnext term resolution microgap RPCs for X-ray imaging applications", Nuclear Instruments and Methods A 505 (2003) 203-206

[54] A. Blanco et al. "Perspectives for positron emission tomography with RPC", Nuclear Instruments and Methods A 508, (2003), 88-93

[55] A. Blanco et al. "Very high position resolution gamma imaging with resistive plate chambers", Nuclear Instruments and Methods A 567 (2006) 96-99