**Національний університет кораблебудування**

**імені адмірала Макарова**

Навчально-науковий інститут автоматики і електротехніки

Кафедра комп’ютеризованих систем управління

**РЕФЕРАТ**

на тему: «Графічні процесори»

Студента 5 курсу групи 5341м

спеціальності 151 «Автоматизація та

комп’ютерно-інтегровані технології»

Іванова С. Ю.

Керівник: Черно О. О.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_ Оцінка ECTS \_\_

Миколаїв – 2021

**Зміст**

[Вступ 3](#_heading=h.3dy6vkm)

[1 Загальні відомості графічних прискорювачів 5](#_heading=h.1t3h5sf)

[2 Особливості архітектура GPU 8](#_heading=h.4d34og8)

[2.1 Конвеєр GPU 8](#_heading=h.2s8eyo1)

[2.2 Введення в модель програмування на GPU 9](#_heading=h.17dp8vu)

[2.3 Модель пам'яті GPU 11](#_heading=h.3rdcrjn)

[2.4 Архітектура графічних адаптерів Nvidia 12](#_heading=h.26in1rg)

[3 Особливості програмування GPGPU (OpenCL) 17](#_heading=h.lnxbz9)

[Висновок 20](#_heading=h.2u6wntf)

[Список літератури 21](#_heading=h.19c6y18)

**Вступ**

Розвиток сучасних графічних прискорювачів або Graphic Processor Unit (надалі GPU) привело до появи графічних прискорювачів з програмованим конвеєром. Якщо раніше програміст графічних додатків міг використовувати тільки фіксовану функціональність GPU, то тепер 2 стадії обробки інформації в конвеєрі GPU стали програмованими. Більш точно програмованою стала обробка вершин, що дозволило створювати, зокрема, більш різноманітні моделі «освітлення», а також обробка фрагментів. Корінна зміна архітектури GPU призвела до можливості використання GPU не тільки для цілей комп'ютерної графіки, але і для завдань, які раніше вирішувалися на CPU, таким чином, з'явилася Технологія Обчислень Спільного Призначення на графічні прискорювачі або General Purpose computation on Graphic Processor Unit (надалі просто GPGPU).

Ідея застосування спеціалізованих арифметичних прискорювачів при побудові суперкомп'ютерних систем за останній 5-10 років стала досить популярною завдяки можливості істотного підвищення продуктивності при збереженні рівня енергоспоживання і кількості обчислювальних вузлів. Якщо простежити за еволюцією списку найшвидших суперкомп'ютерів світу Top500, то легко помітити, що перехід до гетерогенних архітектур не раз дозволяв відповідним обчислювальним системам займати перші місця з істотним відривом від «класичних» кластерів.

Графічний прискорювач - це пристрій, який перетворює графічний образ, що зберігається як вміст пам'яті комп'ютера (або самого адаптера), у форму, придатну для подальшого виведення на екран монітора. Перші монітори, побудовані на електронно-променевих трубках, працювали по телевізійному принципом сканування екрану електронним променем, і для відображення був потрібний відеосигнал, що генерується відеокартою.

Застосувати технології, відпрацьовані при створенні потужних ігрових відеокарт для паралельних обчислень, вперше вдалося NVIDIA, не залишився в стороні і AMD зі своїм прискорювачем FireStream. Використання графічних прискорювачів (GPGPU) дозволяє отримати значну обчислювальну потужність в десятки разів дешевше в ціні та споживчій потужності. GPGPU - єдина доступна по ціні можливість отримати «персональний суперкомп'ютер» терафлопсної потужності в звичайному настільному корпусі. Однак не все так просто, програмування для графічних прискорювачів не найпростіше заняття. Також виникають питання, що вибрати - дорогу спеціалізовану Tesla або топову відеокарту, яка швидше і дешевше? У будь-якому випадку, альтернативи безлічі «легких» ядер поки не проглядається, а значить, доводиться програмувати все більш паралельно. Зараз вибір ПО, призначеного для паралельної обробки даних з використанням не тільки CPU вельми і вельми великий.

# 1 Загальні відомості графічних прискорювачів

Сучасна відеокарта складається з наступних частин:

* графічний процесор (Graphics processing unit - графічний процесорний пристрій) - займається розрахунками зображення, що виводиться, звільняючи від цього обов'язку центральний процесор, проводить розрахунки для обробки команд тривимірної графіки. Відеоконтролер - відповідає за формування зображення в відеопам'яті, дає команди RAMDAC на формування сигналів розгортки для монітора і здійснює обробку запитів центрального процесора. Крім цього, зазвичай присутні контролер зовнішньої шини даних (наприклад, PCI або AGP), контроллер внутрішньої шини даних і контроллер відеопам'яті.
* відеопам'ять - виконує роль кадрового буфера, в якому зберігається зображення, що генерується і постійно змінюється графічним процесором і що виводиться на екран монітора (або декількох моніторів). Цифро-аналоговий перетворювач (ЦАП, RAMDAC - Random Access Memory Digital-to-Analog Converter) - служить для перетворення зображення, сформованого відеоконтролером, в рівні інтенсивності кольору, що подаються на аналоговий монітор.
* відео-ПЗУ (Video ROM) - постійний запам'ятовуючий пристрій, в яке записано відео-BIOS, екранні шрифти, службові таблиці і т.п. ПЗУ не використовується відеоконтролером безпосередньо - до нього звертається тільки центральний процесор.
* система охолодження - призначена для збереження температурного режиму відеопроцесора і відеопам'яті в допустимих межах.

Сучасні GPU мають високу швидкість доступу до своєї власної оперативної пам'яті, яка зазвичай іменується текстурною пам'яттю, і мають високу обчислювальну потужність.

Наприклад наведемо деякі цифри: Якщо у Intel 3.0 GHz Pentium 4 пікова обчислювальна потужність оцінюється як 12 GFLOPs, то у відеокарти ATI Radeon X1800XT 120 GFLOPs, для цього ж обладнання пікова швидкість обміну даними між процесором і відповідної пам'яттю становить відповідно 5.96 GB / s у CPU проти 42 GB / s у GPU1.

На рис.1 показана продуктивність відеокарт різних поколінь з роздільною здатністю 1920х1080:

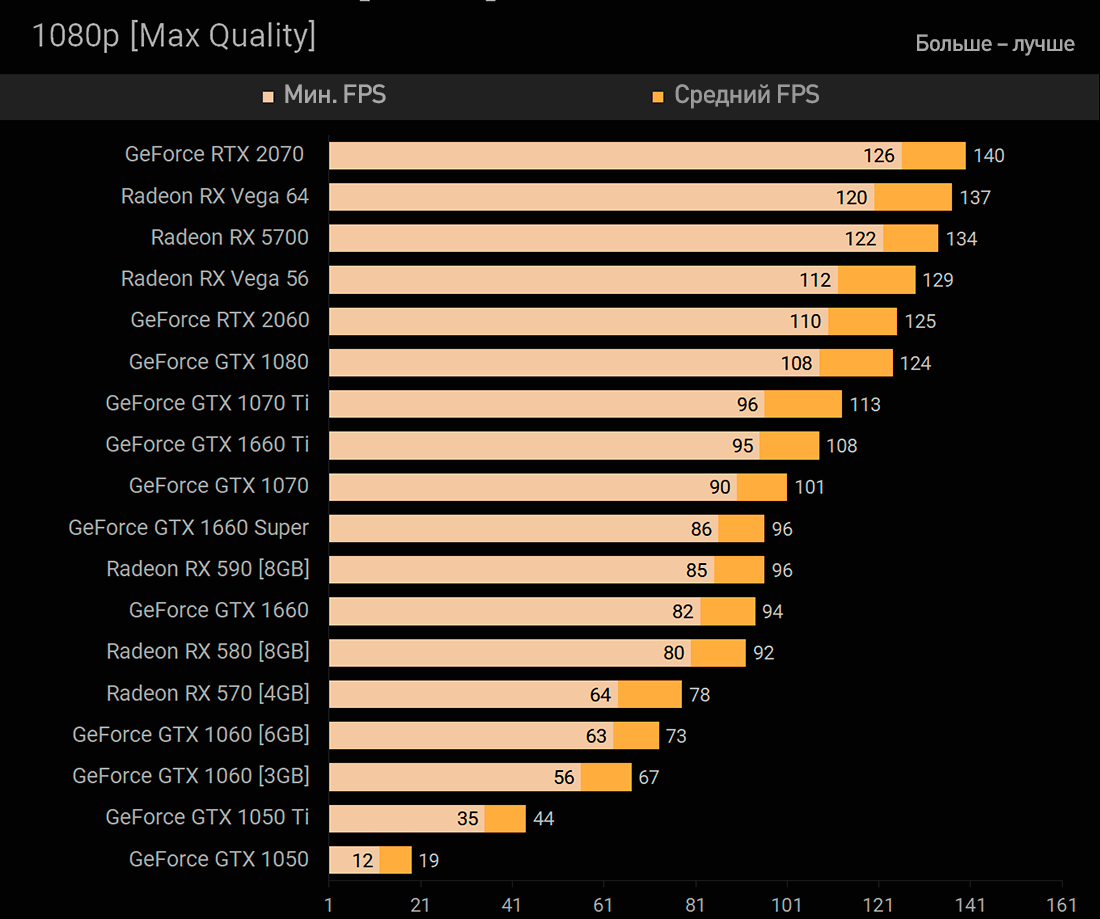


Рис.1 Порівняння відеокарт різних поколінь в benchmark

Але GPU не тільки швидкі і потужні, але і що важливо, їх потужності ростуть дуже швидкими темпами.

Вершинний процесор - програмований модуль, який виконує операції над вхідними значеннями вершин та інші пов'язані з ним дані. Вершинний процесор призначений для виконання наступних традиційних операцій з графікою:

* перетворення вершин;
* перетворення нормалі, нормалізації;
* генерування текстурних координат;
* перетворення текстурних координат;
* налаштування освітлення;
* накладення кольору матеріалу.

Фрагментний процесор - програмований модуль, який виконує операції над фрагментами та іншими пов'язані з ним дані. Фрагментний процесор може виконувати наступні стандартні графічні операції:

* операції над інтерпольованого значеннями;
* доступ до текстур;
* накладення текстур;
* створення ефекту димки;
* накладення квітів.

Якщо врахувати при цьому відносно низьку вартість відеокарт в порівнянні з рівнопотужними CPU, то стане зрозуміло навіщо використовувати відеокарти для обчислень загального призначення.

**2 Особливості архітектура GPU**

**2.1 Конвеєр GPU**

Архітектура GPU визначилася завдяки специфічним потребам комп'ютерної графіки. Саме це визначило високий паралелізм архітектури та її специфіку, завдяки специфічній архітектурі GPU виконують поставлені перед ними завдання набагато ефективніше, ніж класичні процесори.

Структура проведення операцій на всіх сучасних GPU може бути представлена ​​конвеєром (graphic pipeline), який влаштований таким чином, щоб забезпечити максимальну ефективність виконання завдань комп'ютерної графіки за умови потокової паралельної архітектури.

На вхід подається набір вершин, що надходить з програми потім вершини, обробляються в вершинних процесорах, що класифікуються як MIMD, програма для вершинних процесорів називається вершинним шейдером. Після цього результат роботи вершинного шейдера надходить на збірку примітивів, таких як полігони або лінії, далі йдуть тести видимості відсікання та інші стандартні операції комп'ютерної графіки, після них дані надходять на растеризування. Тут об'ємне зображення проектується на плоский екран, і отримана картинка масштабується відповідно до параметрів вікна програми. Результатом цієї операції є картинка, текстура представляє з себе прямокутний масив так званих пікселів. Піксель може бути представлений 1-4 числами, наприклад, в форматі RGBA, ті red, green, blue, alpha (коефіцент прозорості).

Ця текстура обробляється в піксельних процесорах, які за класифікацією багатопроцесорних систем можуть бути класифіковані як SIMD. Потім слід ще кілька операцій, і дані надходять в буфер кадру, який власне і є тією картинкою, яку споживач бачить на екрані. Нижче наведена схема цього конвеєра, звичайно, дане тут опис не претендує на повноту, але для цілей GPGPU цього цілком достатньо.

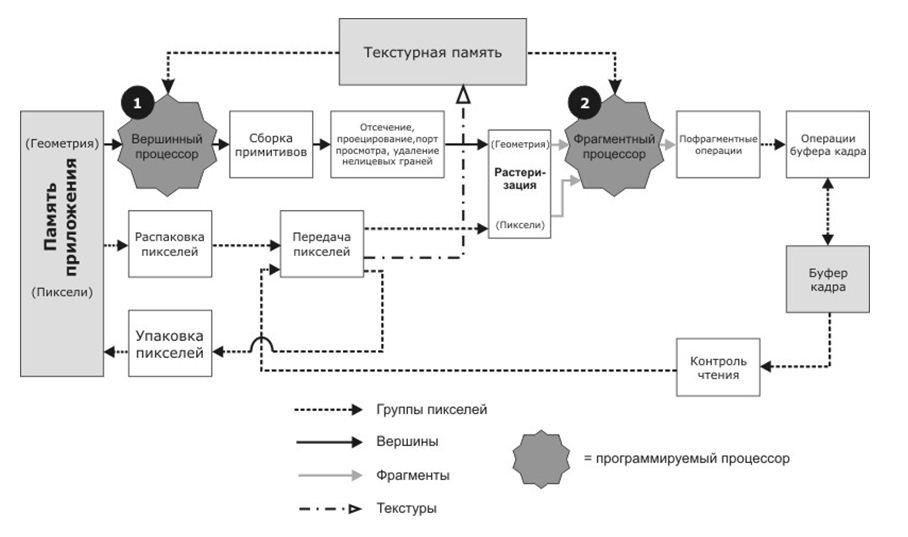


Рис.2 Схематичне зображення роботи GPU-конвеєра

І вершинні і фрагментного процесори мають загальну пам'ять, яка на відеокарті іменується текстурною пам'яттю. У шейдера розробник має доступ на читання текстурной пам'яті, але не на безпосередній запис.

**2.2 Введення в модель програмування на GPU**

Програми, що працюють з тривимірною графікою і відео (гри, GIS, CAD, CAM і ін.), використовують шейдери для визначення параметрів геометричних об'єктів або зображення, для зміни зображення (для створення ефектів зсуву, відображення, заломлення, затемнення з урахуванням заданих параметрів поглинання і розсіювання світла, для накладення текстур на геометричні об'єкти і ін.).

Для вирішення проблеми в відеокарти стали додавати (апаратно) алгоритми, затребувані розробниками. Незабаром стало ясно, що реалізувати всі алгоритми неможливо і недоцільно; вирішили дати розробникам доступ до відеокарти - дозволити збирати блоки графічного процесора в довільні конвеєри, що реалізують різні алгоритми. Програми, призначені для виконання на процесорах відеокарти, отримали назву «шейдери». Були розроблені спеціальні мови для складання шейдеров. Тепер в відеокарти завантажуються не тільки дані про геометричні об'єкти «геометрія», текстури та інші дані, необхідні для малювання (формуванні зображення), а й інструкції для GPU.

Розділяють два види пристроїв – те, яке управляє загальною логікою - host, і те, яке вміє швидко виконати певний набір інструкцій над великим об'ємом даних - device.

У ролі хоста зазвичай виступає центральний процесор (CPU). У ролі обчислювального пристрою - відеокарта (GPU). Відеокарта містить Compute Units - процесорні ядра. Плутанину вводять і виробники NVidia називає свої Streaming Multiprocessor unit або SMX, а ATI - SIMD Engine або Vector Processor. В сучасних ігрових відкритих - їх 8-32.

Процесорні ядра можуть виконувати кілька потоків за рахунок того, що в кожному міститься кілька (8-16) потокових процесорів (Stream Cores або Stream Processor). Для карт NVidia - обчислення проводяться безпосередньо на потокових процесорах, але ATI ввели ще один рівень абстракції - кожен потоковий процесор, складається з processing elements - PE (іноді званих ALU - arithmetic and logic unit) - і обчислення відбуваються на них.

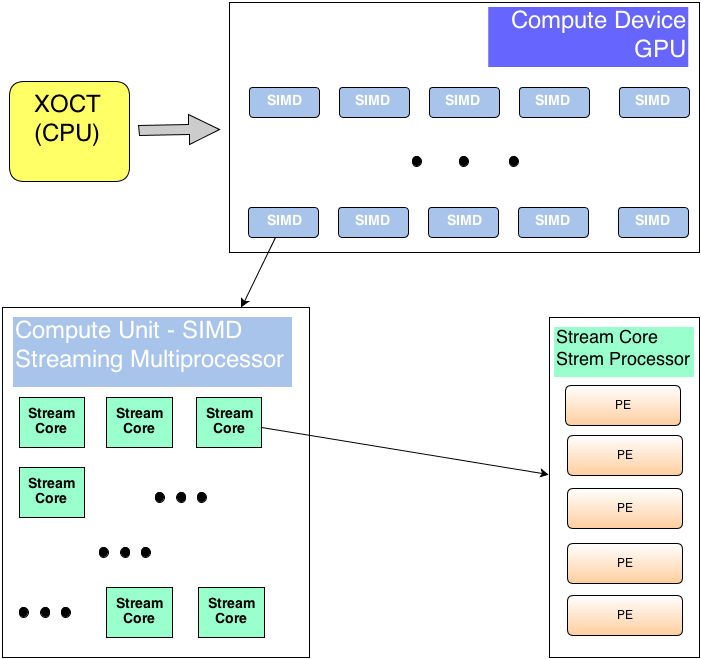


Рис.3 Архітектура моделі GPU

**2.3 Модель пам'яті GPU**

Графічні процесори мають свою власну ієрархію пам'яті, багато в чому схожу з вживаною на звичайних процесорах, але все відмінну від звичайної, архітектура пам'яті GPU створювалася з розрахунку на максимальне прискорення роботи алгоритмів комп'ютерної графіки.

На GPU / CUDA виділяють 6 видів пам'яті:

* Реєстрова;
* Колективна;
* Локальна;
* Глобальна;
* Константна;
* Текстурна.

GPU так само як CPU має свої власні регістри і кеш, для прискорення доступу до даних під час обчислень. Крім цього, GPU має свою власну основну пам'ять (графічна пам'ять), тому для того, щоб програміст міг виконувати обчислення на GPU він повинен попередньо передати дані з оперативної пам'яті CPU в пам'ять GPU. Ця операція традиційно є досить дорогою з точки зору продуктивності через відносно не високій швидкості передачі даних між пам'яттю програми та пам'яттю відеокарти, хоча сучасні шини PCI Express і спеціальні чіпи на материнській платі (такі як NV3 і NV4) помітно прискорили цей процес.

На відміну від пам'яті CPU пам'ять GPU має деякі серйозні обмеження, і доступ до неї може бути здійснений тільки за допомогою деяких абстракцій графічного програмного інтерфейсу. Кожна така абстракція може вважатися як тип потоку зі своїм власним набором правил доступу. Таких потоків безпосередньо доступних для програміста може бути виділено 4 - потоки вершин потоки фрагментів потоки буфера кадрів потоки текстур.

Модель пам'яті в CUDA відрізняється можливістю побайтно адресації, підтримкою як gather, так і scatter. Доступно досить велика кількість регістрів на кожен потоковий процесор, до 1024 штук. Доступ до них дуже швидкий, зберігати в них можна 32-бітові цілі або числа з плаваючою крапкою.

**2.4 Архітектура графічних адаптерів Nvidia**

Одним з найбільш поширених видів гібридних машин є гібридна машина на основі графічних адаптерів компанії Nvidia. Для того щоб ефективно вміти програмувати під такого роду машини необхідно детально розуміти пристрій графічного адаптера з апаратної точки зору.

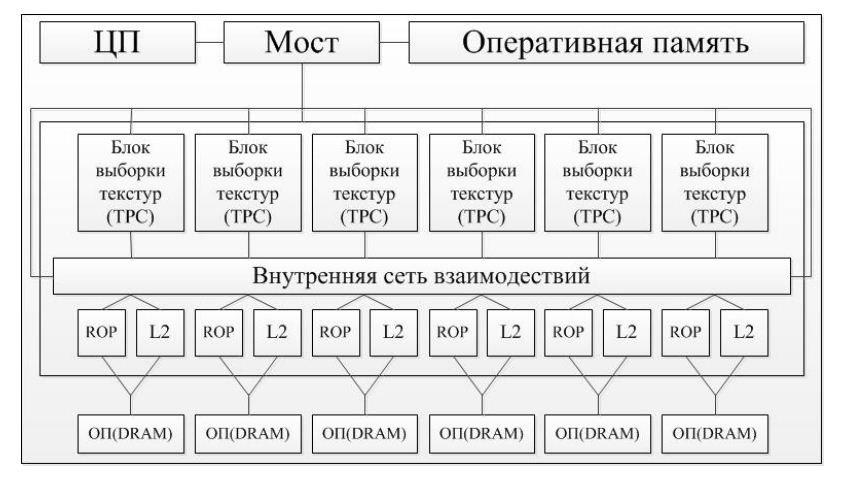


Рис.4 Схематичне зображення відеокарти

Першою відеокартою на основі графічного процесора GK104 стала GeForce GTX 680. Вона замінила перестав випускатися GTX 580 на базі GF110. З одного боку, принцип найменування відеокарт NVIDIA начебто не змінився, топова модель отримала зміна першої цифри індексу. З іншого - навіть судячи по кодовому імені чіпа GK104, він спочатку навряд чи планувався в ролі саме топового рішення.

Цілком ймовірно, спочатку це мала бути GTX 670 (приставка Ti за смаком) або щось в цьому роді, але потім вирішили почекати з цим топовим чіпом, раз у TSMC з новим техпроцесом досі справи неідеальні, і випустити в якості верхньої моделі менш потужний чіп. Втім, ми не раз говорили про те, що найменування відеокарт завжди є маркетинговим рішенням, яке не дуже впливає на технічні характеристики.

Правда, воно сильно впливає на роздрібну ціну рішень. За всіма зовнішніми ознаками (складність чіпа, складність друкованої плати, енергоспоживання, та й собівартість, швидше за все), GTX 680 більше схожа на рішення з верхнього середнього діапазону. На це ж натякають і кодові імена чіпів: GF104 - GK104. Втім, після виходу відеоплат AMD за високими цінами, які не дуже сильно обігнали GeForce GTX 580, у NVIDIA з'явився велика спокуса підняти GK104 у верхній ціновий діапазон, заробивши самим і даючи заробити своїм партнерам по випуску відеокарт.

Всі обчислювальні ядра на графічний адаптер об'єднані в незалежні блоки TPC (Texture process cluster) кількість яких залежить як від версії чіпа (G80 - максимум 8, G200 - максимум 10), так і просто від конкретної відеокарти всередині чіпа (GeForce 220GT - 2, GeForce 275 - 10). Так само, як від відеокарти до відеокарти може змінюватися кількість TPC, так само може змінюватися і кількість DRAM партіцій і відповід повідно загальний обсяг оперативної пам'яті. DRAM партіціі мають кеш другого рівня і об'єднані між собою комунікаційною мережею, в яку, так само підключені всі TPC. Будь-яка відеокарта підключається до CPU через міст, який може бути, як інтегрований в CPU (Intel Core i7), так і дискретним (Intel Core 2 Duo). Від чіпа до чіпу (від G80 до G200) змінювалися деталі в TPC, а загальна архітектура залишалася однаковою. У новому чіпі Fermi відбулися зміни і в загальній архітектурі, тому про нього мова піде окремо.

Архітектура чіпа G80

Будь-який з 128 потокових процесорів G80 являє собою звичайне обчислювальний пристрій, здатне працювати з даними в форматі з плаваючою комою. Отже, він не тільки здатний обробляти шейдери будь-якого типу - вершинні, піксельні або геометричні, але і використовуватися для прорахунку фізичної моделі або інших розрахунків, в рамках концепції Compute Unified Device Architecture (CUDA), причому, незалежно від інших процесорів. Іншими словами, одна частина GeForce 8800 може займатися будь-якими розрахунками, а інша, наприклад, візуалізацією їх результатів, благо, потокова архітектура дозволяє використовувати результати, отримані на виході одного з процесорів в якості джерела даних для іншого.

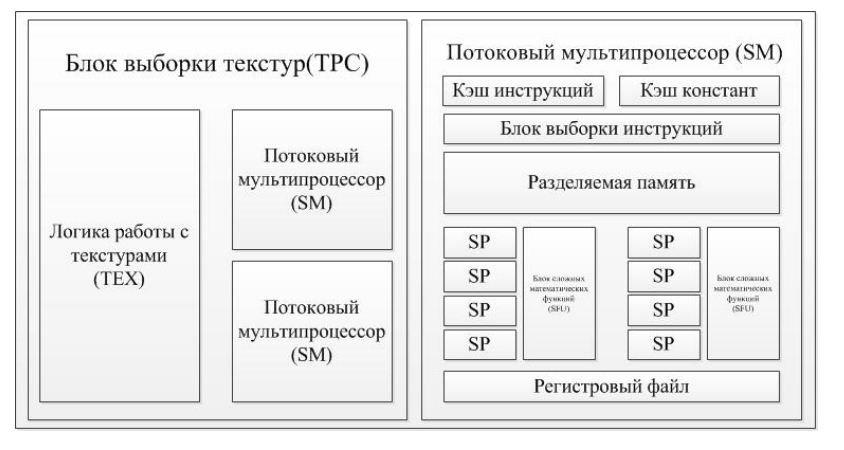


Рис.5 Схема чіпа G80

Створюючи GeForce 8800, NVIDIA дбала не тільки про продуктивність, але і про якість зображення - властивості, в якому попередні розробки компанії часто поступалися виробам ATI Technologies. Зміни та поліпшення торкнулися як повноекранного згладжування, так і анізотропної фільтрації.

Основні складові TPC:

* TEX - логіка роботи з текстурами, містить в собі ділянки конвеєра, призначені для обробки особливої ​​текстури пам'яті про яку йтиметься нижче.
* SM - потоковий мультипроцессор, самостійний обчислювальний модуль, саме на ньому здійснюється виконання блоку. В архітектурі чіпа G80 в одному TPC знаходиться 2 SM.

Основні складові SM:

* SP - потоковий процесор, безпосередньо обчислювальний модуль, здатний здійснювати арифметичні операції з цілочисельними операндами і з операндами з плаваючою точкою (одинарна точність). Чи не є самостійною одиницею, управляється SM.
* SFU - модуль складних математичних функцій. Проводить обчислення складних математичних функцій (exp, sqr, log). Використовує обчислювальні потужності SP.
* Регістровий файл - єдиний банк регістрів, на кожному SM є 32Кб. Найшвидший тип пам'яті на графічний адаптер.
* Пам'ять, - спеціальний тип пам'яті, призначений для спільного використання даних тредов з одного блоку. На кожному SM - 16Кб розділяється пам'яті.
* Кеш констант - місце кешування особливого типу пам'яті (константної). Про особливості застосування йтиметься нижче.
* Кеш інструкцій, блок вибірки інструкцій - керуюча система SM. Не грає ролі при програмуванні.

Разом, чіп G80 має максимально 128 обчислювальних модулів (SP) здатних виконувати обчислення з цілими числами і числами з плаваючою точкою з одинарної точністю. Такого функціоналу було недостатньо для багатьох наукових завдань, була потрібна подвійна точність.

**3 Особливості програмування GPGPU (OpenCL)**

OpenCL (Open Computing Language) - відкритий стандарт паралельного програмування для гетерогенних платформ (зокрема, для GPGPU), що включають центральні, графічні процесори і інші дискретні обчислювальні пристрої. OpenCL дозволив використовувати потужності GPU на різних програмних і апаратних платформах.

Модель платформи (platform model) дає високорівневе опис гетерогенної системи. Центральним елементом даної моделі виступає поняття хоста (host) - первинного пристрою, який управляє OpenCL-обчисленнями і здійснює всі взаємодії з користувачем. Хост завжди представлений в єдиному екземплярі, в той час як OpenCL-пристрої (devices), на яких виконуються OpenCL-інструкції можуть бути представлені у множині. OpenCL-пристроєм може бути CPU, GPU, DSP або будь-який інший процесор в системі, що підтримується встановленими в системі OpenCL-драйверами. OpenCL-пристрої логічно діляться моделлю на обчислювальні модулі (compute units), які в свою чергу діляться на обробні елементи (processing elements). Обчислення на OpenCL-пристроях в дійсності відбуваються на обробних елементах.

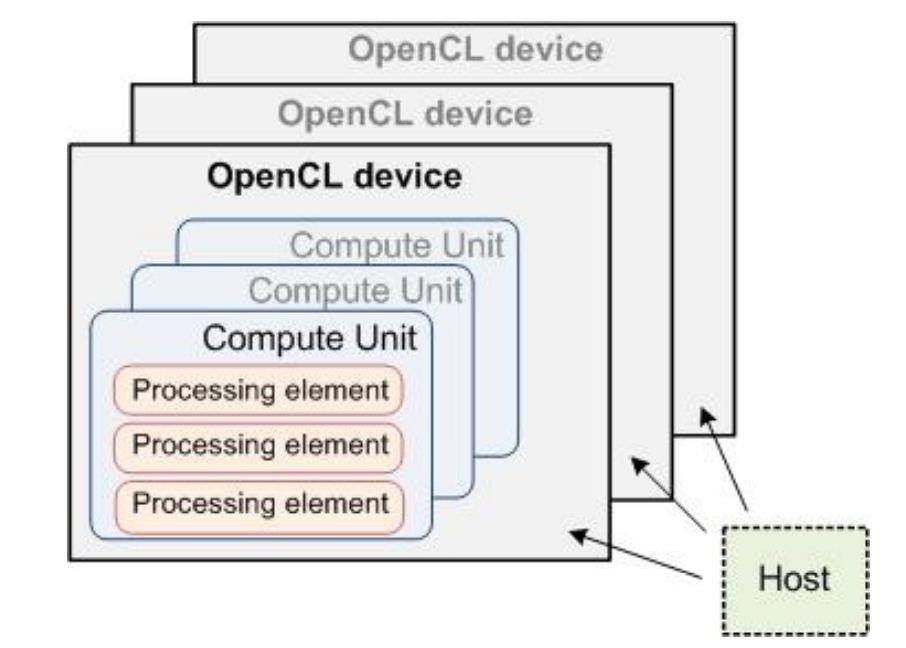


Рис.6 Схематичне зображення OpenCL-платформи з 3-х пристроїв

OpenCL спочатку був розроблений в компанії Apple Inc. Apple внесла пропозиції щодо розробки специфікації в комітет Khronos. Незабаром компанія AMD вирішила підтримати розробку OpenCL (і DirectX 11), який повинен замінити фреймворк Close to Metal.

**Програмна модель OpenCL**

Ядро (kernel) - функція, що виконується пристроєм. Має в описі специфікацію \_\_kernel.

Програма (program) - набір ядер, а також, можливо, допоміжних функцій, що викликаються ними, і константних даних.

Додаток (application) - комбінація програм, що працюють на керуючому вузлі і обчислювальних пристроях.

Команда (command) - операція OpenCL, призначена для виконання (виконання ядра на пристрої, маніпуляції з пам'яттю і т. д.)

**Об'єкти OpenCL**

Об'єкт (object) - абстрактне уявлення ресурсу, керованого OpenCL API (об'єкт ядра, пам'яті і т. Д.)

Дескриптор (handle) - непрозорий тип, який посилається на об'єкт, що виділяється OpenCL. Будь-яка операція з об'єктом виконується через дескриптор.

Черга команд (command-queue) - об'єкт, що містить команди для виконання на пристрої.

Об'єкт ядра (kernel object) - зберігає окрему функцію ядра програми разом зі значеннями аргументів.

Об'єкт події (event object) - зберігає стан команди. Призначений для синхронізації.

Об'єкт буфера (buffer object) - послідовний набір байт. Доступний з ядра через покажчик і з керуючого вузла за допомогою викликів API.

Об'єкт пам'яті (memory object) - посилається на область глобальної пам'яті.

**Пристрої OpenCL**

Робочий елемент (work-item) - набір паралельно виконуваних ядер на пристрої, викликаних за допомогою команди.

Робоча група (work-group) - набір взаємодіючих робочих елементів, що виконуються на одному пристрої.

Виконують один і той же ядро, поділяють локальну пам'ять і бар'єри робочої групи.

Обробляє елемент (processing element) - віртуальний скалярний процесор. Робочий елемент може виконуватися на одному або декількох обробних елементах.

Обчислювальний вузол (compute unit) - виконує одну робочу групу. Пристрій може складатися з одного або декількох обчислювальних елементів.

**Контекст**

Контекст (context) - середовище, в якій виконуються ядра, а також область визначення синхронізації і управління пам'яттю. включає:

* набір пристроїв;
* пам'ять, доступну пристроїв;
* властивості пам'яті;
* одну або кілька черг команд.

Об'єкт програми (program object) - включає:

* посилання на пов'язаний контекст;
* вихідний текст або двійкове подання;
* останній вдало зібраний код, список пристроїв, для яких він зібраний, настройки і журнал збірки;
* набір поточних пов'язаних ядер.

# Висновок

У данному рефераті я постарався розкрити особливості архітектури і програмування графічних процесів, зібрав і проаналізував відомості про архітектуру графічних прискорювачів, а також способи їх програмування.

Розглянув наступні питання:

* Розвиток графічних прискорювачів;
* Переваги GPGPU;
* Архітектура GPGPU;
* Технології програмування GPGPU: CUDA, OpenCL, Direct X.

# Список літератури

1. Mike Houston. General Purpose Computation on Graphics Processors.
2. David Luebke. General-Purpose Computation on Graphics Hardware. University of Virginia.
3. Кривов М.А. «Оптимизация приложений для гетерогенных архитектур. Проблемы и варианты решения».
4. Всё о суперкомпьютерах [Электронный ресурс] // Блог о суперкомпьютерах. – Режим доступа: <https://habr.com/en/post/116733/>
5. О проекте OpenCL [Электронный ресурс] // Официальный сайт OpenCL. – Режим доступа: <http://opencl.ru/about>
6. OpenCL 1.0 Reference Pages [Электронный ресурс] // Официальный сайт комитета Khronos. – Режим доступа: <https://www.khronos.org/registry/OpenCL/sdk/1.0/docs/man/xhtml/>