Висновки

Розглянуті методи оцінювання якості зображень досить просто реалізувати в системах додрукарської підготовки [3, 6]. Наявність людини-спостерігача утрудняє проведення досліджень і породжує суб'єктивні помилки оцінювання якості зображення. Очевидно, що вдосконалення процедури оцінки якості зображень повинно йти шляхом формалізації показника, тобто заміни спостерігача його математичною моделлю, а також вираження цього критерію через такі характеристики зображення, які можуть бути виміряні об'єктивно по довільних реальних зображеннях.

1. *Тимченко О. В., Гавриш Б. М., Лях І. М.* Якість поліграфічної продукції, відтвореної цифровим способом // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Вип.74. – К.: 2015. – С.95-107.
2. *Гренандер У.* Лекции по теории образов: Анализ образов 2. - М.: Мир, 1981. - 448 с.
3. *Гавриш Б., Тимченко О.* Методи визначення якості зображення // Комп'ютерні технології друкарства. Зб. наук. пр. – Вип. 31. – Львів: УАД. – 2014. – 146 с. – С.102-107.
4. *Даджион Д., Мерсеро Р.* Цифровая обработка многомерных сигналов. - М.: Мир, 1988. – 488 с.
5. *Гавриш Б.М., Дурняк Б.В., Тимченко О.В., Ющик О.В.* Відтворення зображень растровими скануючими пристроями. / Львів: Вид. УАД, 2016. – 180 С.
6. *Гавриш Б.М., Тимченко О.В., Левицька Г.Н., Поліщук М.Б.* Інформаційна технологія формування та опрацювання зображень у видавничих вивідних системах // Наукові записки УАД. Вип. № 2 (53). – Львів, УАД, 2016. – С.93-104.
7. *Гавриш Б.М., Тимченко О.В.* Методи опрацювання потоку цифрових даних в процесорах растрових перетворень // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Вип.71. – К.: 2014. – С.142-147.

Поступила 5.09.2016р.

УДК 004.89

І.Г. Цмоць, д.т.н., проф., Національний університет "Львівська політехніка"
О.М. Березький, д.т.н., проф., І. В. Ігнатєв, Тернопільський національний економічний університет

ШЛЯХИ ПІДВИЩЕННЯ ШВИДКОДІЇ ОБРОКИ ЗОБРАЖЕНЬ В КОМП'ЮТЕРНІ СИСТЕМІ З ГРАФІЧНИМ ПРОЦЕСОРОМ

Анотація. Показано, що для зменшення часу обміну між центральний і графічним процесорами доцільно використовувати багатопортову пам'ять, розроблено орієнтований на графічні процесори метод просторово-часового

відображення алгоритмів обробки зображень та запропоновано для програмної реалізації алгоритмів обробки зображень використовувати технологію CUDA.

Abstract. The feasibility of using multiport memory to reduce the time of exchange between the CPU and GPU has been shown. The method of space-time mapping for GPU-based image processing algorithms has been designed. Using of CUDA technology for software implementation of algorithms for image processing has been offered.

Постановка задачі. Основними шляхами підвищення швидкодії обробки зображень у комп'ютерних системах є: розпаралелювання процесу оброки, використання масово-паралельних обчислюваних засобів із великим обсягом пам'яті (графічних процесорів GPU – Graphics Processing Unit) та зменшення часу обміну між центральний процесором CPU і GPU. Графічні процесори GPU відносяться класу процесорів SIMD (Single Instruction Multiple Data), особливістю яких є використання однієї операції для одночасно опрацювання множини незалежних даних. При розробці програмного забезпечення для паралельного опрацювання зображень на базі CPU і GPU використовується кросплатформова система компіляції та виконання програм Compute Unified Device Architecture (CUDA), яка зменшує час розробки програм та підвищує їх якість.

Зменшити часу обміну між центральний процесором CPU і GPU можна за рахунок суміщення в часі процесів введення даних та виведення результатів обробки шляхом використання пристрою обміну на базі багатопортової пам'яті.

Отже, підвищення швидкодії обробки зображень у комп'ютерних системах на базі процесорів CPU і GPU є актуальною задачею

Аналіз останніх досліджень і публікацій. Аналіз публікацій [1-9] показує, що основними шляхами підвищення швидкодії обробки зображень у комп'ютерних системах на базі процесорів CPU і GPU є розпаралелення і адаптація алгоритмів обробки до архітектури апаратних засобів. У роботах [1-5] розглянуті технології розробки паралельних алгоритмів і різні моделі програмування. Проте у даних публікаціях мало уваги приділено технологіям та моделям орієнтованим на архітектуру графічних процесорів GPU.

У роботах [5-7] показано, що час обміну даними між CPU і GPU значно впливає на швидкодію обробки зображень. Таким чином недостатньо вивченими є питання, що пов'язані з організацію обміну CPU і GPU.

Метою роботи є зменшення часу обміну між CPU і GPU, розроблення методів розпаралелювання алгоритмів обробки зображень, які враховують архітектуру комп'ютерної системи і програм оброки зображень з використанням технології CUDA.

Основний матеріал. Для забезпечення опрацювання зображень у реальному часі продуктивність комп'ютерні системи повинна бути:

$$П \geq \frac{\beta R F_d}{N},$$

де R – складність алгоритмів обробки; β – коефіцієнт врахування особливостей архітектури комп'ютерній системі на основі CPU і декількох GPU, N - обсяг даних, F_d - частота надходження вхідних даних.

Забезпечити таку продуктивність можна при інтегрованому підході, який охоплює: дослідження та вдосконалення структури комп'ютерних систем базі процесорів CPU і GPU; методи і алгоритми паралельної обробки зображень; адаптацію алгоритмів обробки зображень до архітектури комп'ютерної системи; технології розробки з використанням CUDA.

Особливістю сучасних GPU п'ятого покоління, таких, як nVidia GeForce 8-GTX 200 і AMD(ATI) HD 2K-5K є набір однакових обчислювальних пристроїв (потокових процесорів), що працюють з загальною пам'яттю графічного процесора. Число потокових процесорів, а також розмір пам'яті графічного процесора може бути різним, залежно від моделі GPU. Всі потокових процесори синхронно виконують одну й ту ж команду, що дозволяє віднести GPU до класу SIMD. Система команд потокових процесорів включає арифметичні команди з 32-розрядної точністю, команди управління (розгалуження і цикли), а також команди звернення до пам'яті. Через високі затримок команди доступу до оперативної пам'яті виконуються асинхронно. Тактові частоти GPU нижче, ніж у центрального процесора комп'ютера. Однак завдяки великій кількості потокових процесорів, які працюють паралельно, продуктивність GPU може досягати декількох ТФлопс.

Пристрій обміну на базі багатопортової пам'яті. Вузьким місцем в комп'ютерній системі на основі CPU і декількох GPU є час обміну між CPU і GPU. Зменшити час обміну можна шляхом використання багатопортової пам'яті. Застосування багатопортового механізму взаємодії між CPU та GPU забезпечує суміщення у часі процесів обміну та зменшує проблеми, які пов'язані організаціє обміну комп'ютерній системі.

Поява на ринку мікросхем пам'яті великого обсягу з малим циклом запису і читання, спонукала до розробки на їх основі багатопортової пам'яті [10]. В основу такої пам'яті закладений принцип часового розподілу ресурсів пам'яті між підключеними до неї GPU і GPU. Для комп'ютерної системи обробки зображень найбільший інтерес становлять безконфліктні методи обміну між CPU і GPU. Багатопортова пам'ять, яка реалізує безконфліктні методи обміну будуються на базі швидкодіючих мікросхем пам'яті шляхом доповнення їх контролерами багатопортової пам'яті, кількість яких визначається кількістю процесорів. Для здійснення ефективного обміну між компонентами комп'ютерної системи обробки зображень багатопортова пам'яті повинна забезпечувати: високу інтенсивність обміну даними; обмін з необхідною кількістю GPU; буферизацію даних і мінімізацію затримки їх передачі; безконфліктний обмін даними між CPU та GPU; можливість обміну як послівно, так і масивами.

Для розробки багатопортової пам'яті використаємо метод часового розподілу ресурсів спільної пам'яті. Даний метод вимагає для своєї реалізації менших апаратних затрат у порівнянні з іншими та забезпечує безконфліктний доступ до спільної пам'яті. Реалізацію методу часового розподілу ресурсів спільної пам'яті доцільно використовувати у випадку виконання наступної умови:

$$T_{\min} \geq mt_{ц}\,, \tag{1}$$

де T_{min} - найменший з періодів звертання компонентів до спільної пам'яті; $t_{ц}$ - цикл доступу до спільної пам'яті; m – кількість GPU плюс CPU. Знаючи T_{min} можна визначити $t_{ц}$ циклу доступу до спільного пам'яті:

$$t_{ц} \leq \frac{m}{T_{\min}}\,. \tag{2}$$

Виконання умови (1) забезпечує обмін через спільну з інтенсивністю доступу:

$$P_d = \frac{mn}{T_{\min}}\,, \tag{3}$$

де n – розрядність даних.

Забезпечення обміну з необхідної швидкодії обміну досягається шляхом вибору необхідної швидкодії спільної пам'яті на основі узгодження інтенсивності доступу з інтенсивністю надходження даних у відповідності формули (2).

Ємність спільної пам'яті залежить як від кількості входів GPU, так від розмірів N_i і розрядності n_i масивів даних, якими здійснюється обмін у комп’ютерні системі. Крім того, в спільній пам'яті необхідно передбачити певний обсяг пам’яті V для зберігання даних, які використовуються для організації обміну. Виходячи з наведених міркувань ємність спільної пам'яті повинна бути:

$$Q = V + \sum_{i}^{m} N_i n_i\,. \tag{4}$$

Основними компонентами багатопортової пам'яті є спільна пам'ять, пристрій керування та контролери обміну. Пристрій керування забезпечує формування неперервної послідовності тактових імпульсів з періодом, який визначається за формулою (1), з допомогою яких здійснюється синхронізація доступу до магістралі спільної пам'яті асинхронно працюючих GPU. Формування тактових імпульсів у пристрої керування здійснюється на основі методу фіксованих часових інтервалів. Згідно з цим методом кожному контролеру обміну циклічно надається фіксований часовий інтервал доступу до спільної пам'яті.

Контролери обміну використовується для виконання наступних функцій: формування адреси і сигналів управління доступу до спільної пам'яті.

Основними елементами контролерів багатопортової пам'яті є конвеєрні регістри, генератор адрес та формувачі сигналів управління, які фіксують зовнішні сигнали управління та у видаленні моменти часу формують імпульси управління. В багатопортової пам'яті зв'язок між GPU і спільною пам'яттю здійснюється через конвеєрні регістри даних, які дозволяють провести часовий розподіл ресурсів пам'яті. Найбільш складним вузлом контролера обміну є генератор адреси, який забезпечує генерацію послідовностей адрес запису і читання в залежності від алгоритму обробки. Задання порядку формування адрес та їх зміна здійснюється програмним шляхом.

Обмін інформацією між багатопортовою пам'яттю і GPU може виконуватись послідовно або масивами. Перед початком кожного обміну необхідно запрограмувати контролер для генерації необхідної послідовності адреси. При кожний послідовному обміні вимагає попереднього запису адреси в генератор адреси. При обміні масивами даних початкова адреса записується один раз на початку запису або читання масиву. Запис початкової адреси в генератор адреси супроводжується читанням даних з спільної пам'яті за даною адрескою і записом її в конвеєрний регістр даних.

Читання або запис кожного елементу масиву інформації виконується в два етапи. При читанні на першому етапі виконується читання даних з конвеєрного регістру даних. На другому етапі відбувається збільшення лічильника генератора адреси на одиницю і читання за адресом, що сформувалася, інформації з спільної пам'яті та запис її в конвеєрний регістр даних. При запису інформації з GPU в спільну пам'ять на першому етапі виконується запис інформації в регістр даних, а на другому етапі - запис інформації з виходів регістра даних в спільну пам'ять та збільшення на одиницю вмісту лічильника генератора адреси.

Просторово-часове відображення алгоритмів обробки зображень Для ефективної реалізації алгоритмів обробки зображень на базі GPU є потреба у їх просторово-часовому відображенні на рівні операції потокових процесорів. Просторово-часове відображення алгоритмів обробки зображень повинно забезпечувати виявлення всіх форм паралелізму та знаходження необхідних просторово-часових рішень для його ефективної реалізації. Такі вимоги забезпечує ярусно-паралельна форма (ЯПФ) відображення алгоритму. При ЯПФ подання алгоритму здійснюється розподіл всіх його функціональних операторів Φ_i за ярусами таким чином, що в j-у ярусі розміщені функціональні оператори, які залежать хоча б від одного функціонального оператора (j-1)-го ярусу і не залежать від операторів наступних ярусів. Всі функціональні оператори одного ярусу виконуються незалежно один від одного. Для ефективної реалізації алгоритмів обробки зображень пропонується такі алгоритми подавати в вигляді конкретизованого потокового графу, який враховує особливості архітектури графічного процесора GPU. Процес розробки такого графу здійснюється за чотири етапи [1,5]:

1) декомпозиція алгоритму обробки зображень;

2) проектування комунікацій (обмінів даними) між функціональними операторами;

3) укрупнення функціональних операторів;

4) планування обчислень при реалізації алгоритму.

На етапі декомпозиції алгоритм обробки зображень Φ розбивається на функціональні оператори Φ_{ji}, між якими установлюються зв'язки, що відповідають даному алгоритму. Декомпозицію можна здійснювати за методом функціональної декомпозиції. Використання методу функціональної декомпозиції дозволяє отримати просторово-часове відображення структури алгоритму обробки зображень на основі арифметичних операцій. Результатом першого етапу розробки є граф схема алгоритму, де функціональні оператори Φ_{ji} мають приблизно однаковий час виконання, а їх складність визначається особливістю архітектури графічного процесора GPU.

На етапі проектування комунікацій необхідно визначити структуру та розрядність каналів обміну даними між функціональними операторами Φ_{ji}. Для чого виконується перехід від граф схеми алгоритму до потокового графу, в якому здійснюється просторово-часове розміщення і закріплення функціональних операторів Φ_i за ярусами. Структура зав'язків у потоковому графі між функціональними операторами Φ_{ji} сусідніх ярусів визначається кількістю каналів надходження даних і їх розрядністю.

Отриманий після двох етапів конкретизований потоковий граф забезпечує оцінювання обчислювальної складності алгоритму обробки зображень та визначення необхідної продуктивності універсального CPU і графічного GPU процесорів при його програмній реалізації.;

Третій етап розроблення зводиться до укрупнення операцій шляхом об'єднання функціональних операторів Φ_{jk} і каналів передачі даних як у межах ярусу, так і між ярусами. Цей етап використовується для його адаптації потокового графа до структури універсального CPU і графічного GPU процесорів. Даний етап тісно пов'язаний етапом планування процесу обробки зображення.

Четвертий етап планування обчислень зводиться до збереження інформації про структуру конкретизованого потокового графа алгоритму обробки зображень. На даному етапі виконується планування процесу обчислень, визначаються величини затримок і перестановки даних. Для відтворення процесу опрацювання даних у конкретизований потоковий граф вводяться оператори управління, затримки та перестановки даних. Для отримання конкретизованого потокового графу обробки зображень використовують два основні варіанти укрупнення операцій (об'єднання функціональних операторів).

Першим із варіантів отримання конкретизованого графу алгоритму є його лінійна проекція на горизонтальну вісь X. У цьому випадку укрупнення операцій здійснюється об'єднанням між ярусами функціональних операторів і каналів передачі даних..

Другим варіантом отримання конкретизованого потокового графу

алгоритму є його лінійна проекція на вертикальну вісь Y. У цьому випадку укрупнення операцій здійснюється об'єднанням функціональних операторів і каналів передачі даних як у межах ярусу, так і між ярусами.

У результаті такого укрупнення операцій отримуємо конкретизований потоковий граф, який орієнтований на архітектуру конкретного GPU. Конкретизований потоковий граф характеризується складністю функціональних операторів Φ_{ji}, шириною L (кількість функціональних операторів Φ_{ji} у ярусі графа) і висотою h (кількістю ярусів у графі). Необхідно відмітити, шо параметри графа є взаємно залежними, зміна одного з них веде до зміни інших.

Розробка програмного забезпечення. Для програмної реалізації алгоритмів обробки зображень на GPU використаємо модель програмування CUDA, яка враховує обчислювальний паралелізм та ієрархічну структуру пам'яті. Ця модель програмування ґрунтується на застосуванні однієї операції до множини даних. Послідовність таких операцій є особливістю програми, які написані у рамках даної моделі.

При розробці програмного забезпечення використовувались програмні середовища розробки Nvidia CUDA 5.5, MS Visual Studio 2010 та MS SQL Server Management Studio 2012 [6,7]. Особливістю CUDA є те, що вона дає розробникові можливість на свій розсуд організовувати доступ до набору інструкцій графічного процесора GPU і управляти його пам'яттю, організовувати на ньому складні паралельні обчислення. Графічний процесор з підтримкою CUDA стає потужною програмованою відкритою архітектурою подібно до сьогоднішніх центральних процесорів.

Мовами реалізації алгоритмів оброки зображень на графічному процесорі Nvidia GT було обрано C/C++, яка мовою програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Особливостями мови C/C++ є: швидкодія; масштабованість; можливість роботи на низькому рівні з пам'яттю, адресами, портами; створення узагальнених алгоритмів для різних типів даних, їхня спеціалізація та обчислення на етапі компіляції з використанням шаблонів; підтримка різних стилів та технології програмування.

Висновки.

1. Розробку програмних засобів паралельної обробки даних з використанням графічного процесора GPU та програмної моделі CUDA можна забезпечити при комплексному підході, який охоплює: методи і алгоритми паралельної обробки зображень; архітектуру графічного процесора GPU; методи відображення алгоритмів; програмне середовища розробки Nvidia CUDA.

2. Просторово-часове відображення алгоритмів обробки зображень повинно враховувати архітектури комп'ютерної системи, забезпечувати виявлення всіх форм паралелізму і знаходити необхідні просторово-часові рішення.

3. Застосування багатопортової пам'яті для обміну між CPU та GPU забезпечує суміщення у часі процесів обміну та зменшує час обміну та проблеми його організації.

1. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. — СПб: БХВ-Петербург, 2002. — 608 с.
2. *С.Кун.* Матричные процессоры на СБИС:-М.:Мир,1991.-672 с.
3. *Форсайт Девид.* Компьютерное зрение. Современный подход / Форсайт Девид, Жан Понс. – Москва; Санкт-Петербург; Киев, 2004. – 925 с.
4. *Р.Гонсалес, Р.Вудс.* Цифровая обработка изображений / Р.Гонсалес, Р.Вудс. – Москва: Техносфера, 2006. – 1072 с.
5. *Цмоць І.Г.* Інформаційні технології та спеціалізовані засоби обробки сигналів і зображень у реальному часі. – Львів: УАД, 2005.- 227с.
6. *В.П. Гергель.* Высокопроизводительные вычисления для мнгопроцесорных многоядерных систем // Издательство Московского университета , 2010. – 544с.
7. *А.В. Боресков* и др.Паралельные вычисления на GPU, Архитектрура и программная модель CUDA// Издательство Московского университета , 2012. – 336с.
8. *Ю.М. Рашкевич, Р.О. Ткаченко, І.Г Цмоць, Д.Д. Пелешко.* Нейроподібні методи, алгоритми та структури обробки сигналів і зображень у реальному часі: монографія / – Львів: Видавництво Львівської політехніки, 2014. -256 с.
9. *G. Bradski, A. Kaegler*, Computer Vision with the OpenCV library. 2008.
10. *Б.А. Демида, Ю.М. Рашкевич, І.Г. Цмоць.* Пат.№23358А Україна, МПК G11 C11/00. Багатопортова пам’ять / Бюл.№4, 1998.

Поступила 29.09.2016р.