Міністерство освіти і науки України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Данілов Павло Олександрович

Апаратна реалізації обчислення максимального і мінімального чисел в масиві даних / Hardware implementation of maximum and minimum numbers calculating operation in a dataset

Спеціальність 8.091501 – Комп'ютерні системи та мережі

Дипломна робота за освітньо-кваліфікаційним рівнем «магістр»

Науковий керівник
д.т.н., професор Цмоць І. Г.
_____

Дипломну роботу допущено до захисту
«___»_____ 20 __ р.
Зав. кафедри КІ
Березький О.М. _____

Тернопіль – 2017

Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Освітньо-кваліфікаційний рівень магістр
Спеціальність 8.05010201 комп'ютерні системи та мережі

«Затверджую»
завідувач кафедри
д.т.н., проф. БерезькийО.М.
_____
"____"_____20___ р.

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА
Данілова Павла Олександровича

1. Тема дипломної роботи "Апаратна реалізації обчислення максимального і мінімального чисел в масиві даних"
затверджена наказом №484 від " " 2016 р.

2. Термін здачі закінченої дипломної роботи "___"_____20__ р.

3. Об'єкт дослідження: ообчислення максимального і мінімального чисел в масиві даних

4. Апаратна реалізації обчислення максимального і мінімального чисел в масиві даних на графічному процесорі.

5. Перелік задач, які мають бути вирішені**:**
- дослідити сучасні методи та алгоритми обчислення максимального і мінімального чисел в масиві даних;
- дослідити апаратні засоби, за допомогою яких обчислюються максимальні і мінімальні числа в масиві даних;
- розробити програму обчислення максимального і мінімального чисел в масиві даних на центральному процесорі CPU;
- розробити програму обчислення максимального і мінімального чисел в масиві даних на графічному процесорі GPU;
- виконати тестування розробленого програмного забезпечення.

6. Перелік ілюстративного матеріалу:
- тема, мета, завдання, методи досліджень, наукова новизна, практичне значення;
- актуальність;

– об’єкт дослідження;

– архітектура центрального процесора для сортування даних;

– архітектура графічного процесораGPU;

– будова CPUта GPUпроцесорів;

– приклад алгоритму обчислення максимального і мінімального чисел в масиві даних;

– приклад обчислення максимального і мінімального чисел в масиві даних на центральному процесорі;

– приклад обчислення максимального і мінімального чисел в масиві даних на графічному процесорі;

7. Консультанти по роботі

Розділ	Консультант	Підпис
Нормо-контроль	Мельник Г.М.	

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз галузей застосування, методів та засобів обчислення максимальних і мінімальних значень із масиву чисел		
2	Алгоритми і процесорні елемнти для обчислення максимальних і мінімальних значень		
3	Розробка та моделювання пристрою для обчислення максимальних і мінімальних значень із одновимірного масиву чисел		
4	Нормоконтроль, попердній захист		
5	Захист		

Завдання прийняв до виконання _____ _____
(підпис) (прізвище та ініціали)

Керівник дипломної роботи _____ Цмоць І.Г.
(підпис) (прізвище та ініціали)

РЕЗЮМЕ

Дипломна робота на тему “ Апаратна реалізації обчислення максимального і мінімального чисел в масиві даних ” на здобуття освітньо-кваліфікаційного рівня “Магістр” зі спеціальності “Комп’ютерні системи та мережі” написана обсягом 96 сторінок і містить 32 ілюстрації, 2 таблиці, 4 додатки та 73 джерела за переліком посилань.

Метою роботи є розроблення алгоритмів, структур обчислення максимальних і мінімальних чисел у масиві та їх моделювання.

Методи досліджень. Для розв’язання поставлених задач у дипломній роботі використано: дослідження методів і алгоритмів обчислення максимальних і мінімальних чисел у одновимірному масиві; архітектуру графічного процесора GPU та програмну модель CUDA.

Результати дослідження: розроблено ярусно-паралельну форму пошуку мінімального та максимального значення у одновимірному масиві даних, ярусно-паралельну форму обробки масиву на GPU, який забезпечує виявлення паралелізму.

Результати роботи можуть бути використані в науковій практиці, для розв’язання широкого кола задач із пошуку мінімального та максимального значення у одновимірному масиві даних.

Орієнтовні напрямки розвитку досліджень: пошук мінімальних та максимальних значеннь у одновимірному масиві даних використовується при попередній обробці даних, розробці та моделюванні великого діапазону завдань. Пошук мінімальних та максимальних значеннь у одновимірному масиві даних застосовується при обробці великих масивів інформації.

КЛЮЧОВІ СЛОВА: ПОШУК МІНІМАЛЬНИХ ТА МАКСИМАЛЬНИХ ЗНАЧЕННЬ, ГРАФІЧНИЙ ПРОЦЕСОР, АПАРАТНІ ЗАСОБИ, ЯРУСНА ПАРАЛЕЛЬНА ФОРМА, РЕАЛЬНИЙ ЧАС.

RESUME

Diploma work: "Hardware implementation of calculating operation of maximum and minimum numbers in a dataset" for obtaining educational qualification" to education and qualification of "Master" specialty "Computer systems and networks" written 96 page volume and contains 32 illustrations, 2 tables, 4 applications and 73 sources for references.

The aim is to develop algorithms, structures calculating the maximum and minimum numbers in the array and their modeling.

Research Methods. To solve the tasks in applied research paper, research methods and algorithms for calculating the maximum and minimum numbers in a one-dimensional array; GPU architecture and GPU programming model CUDA.

The results: bunk-developed parallel search form minimum and maximum values in the one-dimensional array of data-tier form of parallel processing array on the GPU, which provides detection of parallelism.

The results can be used in scientific practice, to solve a wide range of problems with finding minimum and maximum values in the one-dimensional array of data.

The estimated directions of research Search for minimum and maximum values in the one-dimensional array of data used in the preliminary data processing, design and simulation of a large range of tasks. Search for minimum and maximum values in the one-dimensional array of data used in the processing of large volumes of information.

KEY WORDS: CALCULATING OPERATION OF MAXIMUM AND MINIMUM NUMBERS, GPU, HARDWARE, STACK PARALLEL FORM, REAL TIME.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

НВІС – надвелика інтегральна схема

ОП – операційними пристроями

ПЕ – процесорний елемент

ПЗ – програмне забезпечення

ФО – функціональні оператори

ПГА – потоковий граф алгоритму

БПП - багатопортова пам’ять.

CPU англ. Central Processing Unit – центральний процесор

CUDA англ. Compute Unified Device Architecture – програмно-апаратна архітектура паралельних обчислень

GPU англ. Graphics Processing Unit – графічний процесор

ВСТУП

Актуальність теми. На сучасному етапі розвитку інформаційних технологій намітилась тенденція збільшення ролі логічних обчислень, а саме обчислення максимальних і мінмальних чисел. Такі обчислення використовуються в нейромережах для нормалізації вхідних даних. Обчислення максимальних і мінмальних чисел мають ряд специфічних особливостей, які не дозволяють при їх реалізації використовувати відомі обчислювальні методи і алгоритми. Існуючі апаратні засоби в основному орієнтовані на реалізацію алгоритмів з перевагою обчислювальних операцій над логічними і вони не враховують специфіки обробки логічних даних. Особливістю реалізації логічних обчислень на сучасних апаратних засобах є неефективність використання багаторозрядних операційних пристроїв, що істотно знижує продуктивність та ефективність використання обладнання. У зв'язку з цим особливої актуальності набуває проблема розробки нових ефективних методів і алгоритмів логічних обчислень, орієнтованих на НВІС-реалізцію. Для переходу від алгоритмів логічних обчислень до НВІС-архітектур доцільно використовувати методи просторово-часового відображення таких алгоритмів у паралельні однорідні структури з високою ефективністю використання обладнання. Процес відображення алгоритмів у паралельні спеціалізовані НВІС-архітектури є складним і вимагає взаємної адаптації як алгоритмів, так і структур обчислювальних засобів.

Нормалізацію вхідних даних в нейромережах – це процедура попередньої обробки вхідних даних (навчальних, тестових і робочих вибірок), при якій значення ознак, які формують вхідний вектор, приводиться до деякого заданого діапазону. Після нормалізації всі значення вхідних признаків будуть приведені до деякого вузького діапазону (зазвичай, [0, 1] або [-1, 1]).

Існує велика кількість способів нормалізації вхідних значень. Найпростішою, але у більшості випадків ефективною, є лінійна нормалізація. Якщо початкові дані потрібно привести до діапазону [0, 1], то вона виконується так:

$$X^{\times} = \frac{X - \min(X)}{\max(X) - \min(X)}.$$

Для приведення початкових даних до діапазону [-1, 1] лінійна нормалізація здійснюється таким чином:

$$X^{\times} = \frac{X}{\max(|X|)}$$

Якщо вхідні дані X щільно заповнюють певний інтервал, то використання лінійної нормалізації оптимальне, оскільки вона не потребує здійснення складних обчислень.

Але лінійна нормалізація ефективна не завжди, наприклад, у випадках, коли існують рідкісні відхилення, які значно більші за типові значення. У таких випадках лінійна нормалізація призведе до того, що більшість значень початкових даних будуть близькими до нуля.

Мета і завдання дослідження. Метою роботи є розроблення алгоритмів, структур обчислення максимальних і мінімальних чисел у масиві та їх моделювання.

Для досягнення поставленої мети в роботі необхідно виконати такі завдання:

- проаналізувати галузі застосування, методи та засоби обчислення максимальних і мінімальних значень із масиву чисел;
- розробити алгоритми і процесорні елемнти для обчислення максимальних і мінімальних значень;
- розробку та моделювання пристрою для обчислення максимальних і мінімальних значень із одновимірного масиву чисел.

Об’єкт дослідження – – процеси обчислення максимальних і мінімальних значень із одновимірного масиву чисел.

Предмет дослідження- методи, алгоритми та засоби обчислення максимальних і мінімальних значень із одновимірного масиву чисел.

Методи досліджень. Для розв’язання поставлених задач у дипломній роботі використано: дослідження методів і алгоритмів обчислення

максимальних і мінімальних чисел у одновимірному масиві; архітектуру графічного процесора GPU та програмну модель CUDA.

Наукова новизна одержаних результатів.

Науковими результатами роботи є:

- розроблений алгоритм обчислення максимальних і мінімальних значень із одновимірного масиву чисел, який за рохунок використання спільної шини забезпечує високу швидкодію.
- розролена структура о днорідною з регулярними звязками, що забезпечує ефективну реалізацію у вигляді НВІС;
- розроблена модель пристрою обчислення максимальних і мінімальних значень із одновимірного масиву чисел показує, що швидкодія визначається в основному розрядністю чисел.

Публікації та апробація ДР. Публікацію тез дипломної роботи та тему «Апаратна реалізації обчислення максимального і мінімального чисел в масиві даних» надруковані у виданні «VI Всеукраїнської школи-семінару молодих вчених і студентів».

Впровадження результатів ДР. Дипломна робота та тему «Апаратна реалізації обчислення максимального і мінімального чисел в масиві даних» відповідає замовленню товариства, має певну практичну значимість і планується до впровадження у науково-дослідному інституті інтелектуальних комп’ютерних систем».

1 АНАЛІЗ ГАЛУЗЕЙ ЗАСТОСУВАННЯ, МЕТОДІВ ТА ЗАСОБІВ ОБЧИСЛЕННЯ МАКСИМАЛЬНИХ І МІНІМАЛЬНИХ ЗНАЧЕНЬ ІЗ МАСИВУ ЧИСЕЛ

1.1 Галузі застосування засобів обчислення максимальних і мінімальних значень

У сучасних комп’ютерах висока продуктивність обробки даних досягається завдяки використанню просторового і часового паралелізму. Особливо цікавою є концепція за допомогою якої підвищення продуктивності комп’ютерного пристрою досягається шляхом наближення його структури до структури виконуваного алгоритму [1]. Перетворення алгоритму в його апаратну модель відбувається шляхом повного апаратного відображення потокового графа виконуваного алгоритму (ПГА) операційними пристроями (ОП), які виконують функціональні оператори (ФО) алгоритму і з’єднані між собою відповідно до графа алгоритму [2]. Пристрої, структура яких адаптована до ПГА, мають наступні переваги переваги:

виконувати операцію із всіма операндами можна незалежно від стану інших операцій (не потрібна синхронізація);

обмін даними між операціями чітко визначений; за допомогою передавання даних між ними здійснюється управління операціями; алгоритм виконується за один такт, що надає багаторазове прискорення виконання конкретної задачі.

Але виникає дуже багато питань, що стосуються, у першу чергу того, як будувати й змінювати графи алгоритмів для можливості їх апаратної реалізації [1, 2, 3, 4].

Розглянуті різні варіанти синтезу паралельних обчислювальних пристроїв сортування та їхнєпроектування від програмного опису алгоритму до апаратної реалізації з можливістю зміниширини паралельної форми.

Доволі високою є частка сортування серед операцій, виконуваних комп’ютером, ця задача є однією з типових проблем обробки даних, і, звичайно, розуміється як задача розміщення елементів невпорядкованого

набору значень $\overline{X} = \{x_1, x_2, ..., x_N\}$ в порядку монотонного зростання або спадання $\overline{X} \approx \overline{Y} = \{(y_1, y_2, ..., y_N) : y_1 \le y_2 \le ... \le y_N\}$. Сортування на основі ПГА будуються комутуючі мережі, які забезпечують здатність до обміну даними між компонентами багатопроцесорної комп'ютерної системи [2]. Також, алгоритми сортування – зручний засіб для визначення переваг тієї або іншої моделі та для їх порівняння між собою.

Обчислювальна трудомісткість процедури впорядкування є доволі високою. Так, для деяких відомих простих методів (бульбашкове сортування, сортування включенням тощо) кількість необхідних операцій визначається квадратичною залежністю від числа даних, що впорядковують $t_S \approx N \log_2 N$ [5]. Паралельне виконання операцій алгоритму сортування декількома операційними пристроями одночасно значно прискорює час виконання алгоритму. Серед алгоритмів сортування паралельне виконання операцій можна виконувати для алгоритмів, в яких послідовність виконуваних операцій залежить тільки від числа вхідних даних N і не залежить від значень їхніх ключів (неадаптивні алгоритми) [6]. Серед алгоритмів сортування неадаптивними є: алгоритм сортування за методом "парно-непарної" перестановки [7], алгоритм сортування модифікованим методом "бульбашки" [6], алгоритм сортування за методом Бетчера [5]. Під час проектування виплили деякі недоліки, які із ускладненням проектів стають доволі значними, одним із недоліків є те що часто приходиться виконувати вручну роботу з розпаралелювання алгоритмів або зміни ширини паралельної форми. З цього виходить що така робота вимагає додаткового часу, високої кваліфікації, і не зважаючи на це, не гарантує збереження простого розв'язку, тим самим, призводить до того, що на деякому етапі ускладнення, проект стає нерентабельним. Це змушує шукати нові шляхи прискорення проектування.

Застосування нейронних мереж є одним з основних методів щодо прискорення проектування засобів пошуку мінімальних та максимальних елементів в масиві даних.

Вище перелічені методи паралельного сортування засновані на застосуванні однієї й тієї самої базової операції "порівняти й переставити", яка

полягає в порівнянні тієї або іншої пари ключів із набору даних для сортування та перестановки цих значень, якщо їхній порядок не відповідає умовам сортування. Ці методи відрізняються між собою лише порядком порівняння пар, а базова операція “порівняти й переставити” однакова для таких алгоритмів. Отримані ПГА даних алгоритмів для сортування 16 вхідних значень зображені на рисунку 1.1.1

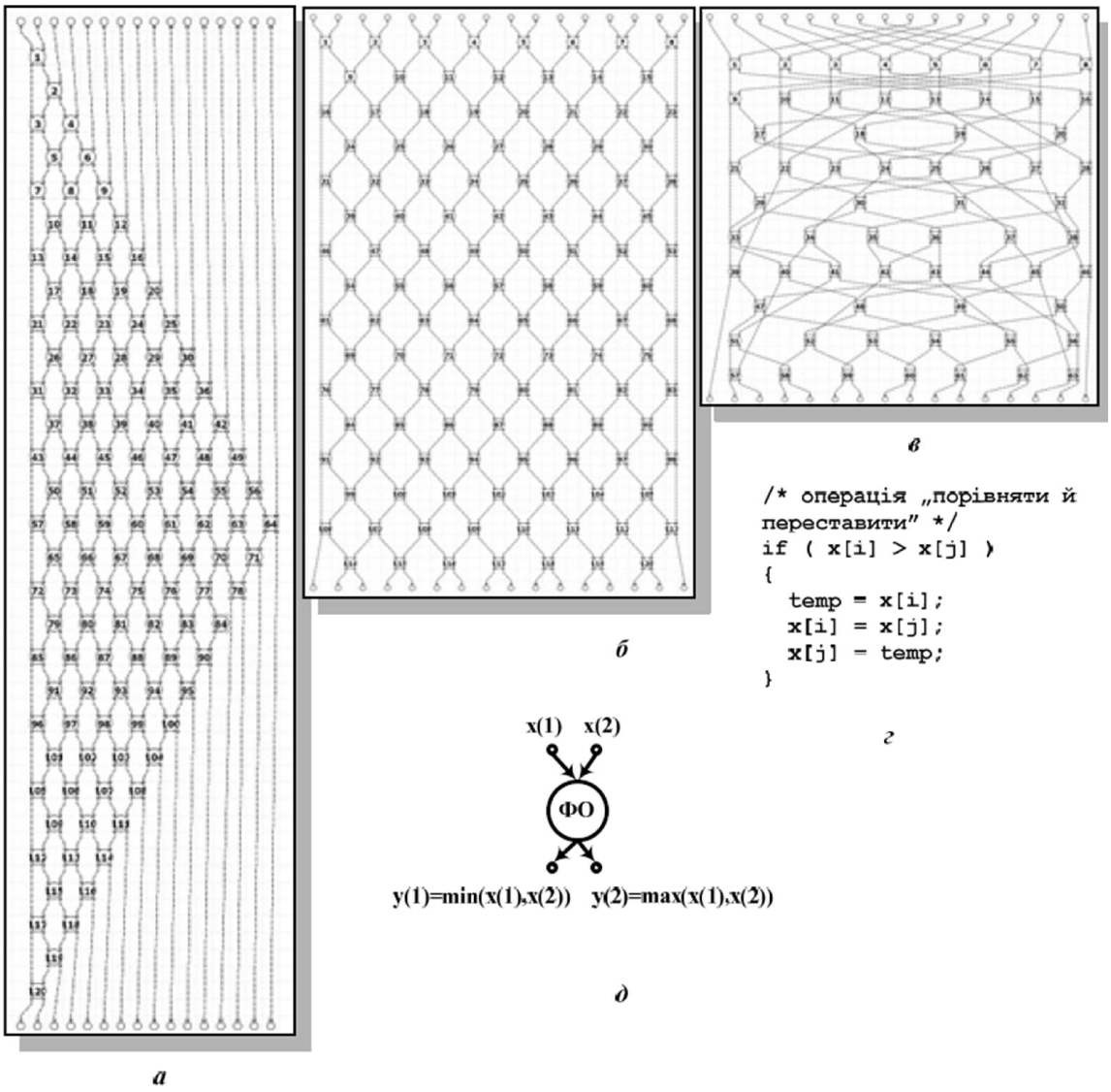


Рисунок 1.1.1 Алгоритми сортування 16 значень: а – ПГА сортування модифікованим методом “бульбашки”; б – ПГА сортування “парно-непарної” перестановки; в – ПГА сортування Бетчера; г, д – базова операція “порівняти й переставити”

Для однакової кількості вхідних значень N дані алгоритми мають однакову ширину ПГА – $\left[\frac{N}{2}\right]$ але виконують різну кількість операцій „порівняти й переставити” та мають різну кількість ярусів.

Загальна кількість ФО ПГА дорівнює загальній кількості операцій „порівняти й переставити”. Для алгоритмів сортування за методом “парно-непарної” перестановки та модифікованим методом “бульбашки” для N вхідних значень кількість ФО дорівнює $N(N-1)/2$ [2], а для алгоритму сортування за методом Бетчера – $0{,}48N\ln^2 N$, що підтверджено результатами моделювання. Висота ПГА для цих алгоритмів різна і є найбільшою для модифікованого алгоритму сортування за методом “бульбашки” – $2N-3$ [2]. Для алгоритму сортування за методом “парно-непарної” перестановки висота ПГА дорівнює кількості вхідних значень N , і для алгоритму сортування за методом Бетчера є найменшою і дорівнює $\frac{1}{2}\lceil\log_2 N\rceil(\lceil\log_2 N\rceil+1)$. Оскільки алгоритми виконують одну операцію „порівняти й переставити”, можна вважати, що часова затримка всіх ФО однакова і дорівнює одиниці. Апаратна складність цих обчислювальних пристроїв дорівнюватиме кількості ФО ПГА, а часова затримка – кількості ярусів ПГА.

Проаналізувавши таблиця 1.1.1 , було зроблено висновок, що з усіх алгоритмів алгоритм сортування за методом Бетчера має найкращі часові властивості за найменших апаратних затрат. Оскільки ПГА сортування за методом Бетчера для N вхідних даних містить $\frac{1}{2}\lceil\log_2 N\rceil(\lceil\log_2 N\rceil+1)$ ярусів. [5], то час виконання алгоритму становитиме $0(\frac{1}{2}\lceil\log_2 N\rceil(\lceil\log_2 N\rceil+1))$.

Таблиця 1.1.1 – Обчислювальні характеристики алгоритмів сортування залежно від кількості вхідних значень

Алгоритм сортування	Кількість операцій, w	Кількість ярусів, t
модифікований “бульбашки”	$N(N-1)/2$	$2N-3$

Продовження таблиці 1.1.1

“парно-непарної” перестановки	$N(N-1)/2$	N
за методом Бетчера	$0{,}48N\ln^2 N$	$\frac{1}{2}[\log_2 N]([\log_2 N]+1)$

За останні 20 років штучні нейронні мережі отримали широке поширення і дозволяють вирішувати складні завдання обробки даних, часто значно перевершуючи точність інших методів статистики і штучного інтелекту, або будучи єдино можливим методом вирішення окремих завдань. Нейромережа подібна до структуру і властивостей нервової системи живих організмів: нейронна мережа складається з великого числа простих обчислювальних елементів (нейронів) і володіє більш складною поведінкою в порівнянні з можливостями кожного окремого нейрона. На вході нейромережа отримує набір вхідних сигналів і видає відповідний їм відповідь тобто вихідні сигнали, що є рішенням задачі.

Сфери застосування нейромереж:

- Економіка і бізнес: передбачення ринків, автоматичний дилінг, оцінка ризику неповернення кредитів, передбачення банкрутств, оцінка вартості нерухомості, виявлення пере-і недооцінених компаній, автоматичне рейтингування, оптимізація портфелів, оптимізація товарних і грошових потоків, автоматичне зчитування чеків і форм, безпеку транзакцій по пластиковим карткам.
- Медицина: обробка медичних зображень, моніторинг стану пацієнтів, діагностика, факторний аналіз ефективності лікування, очищення показань приладів від шумів.
- Авіоніка: учні автопілот, розпізнавання сигналів радарів, адаптивне пілотування сильно пошкодженого літака.
- Зв'язок: стиснення відео-інформації, швидке кодування-декодування, оптимізація стільникових мереж і схем маршрутизації пакетів.

- Інтернет: асоціативний пошук інформації, електронні секретарі і агенти користувача в мережі, фільтрація інформації в push-системах, Коллаборативні фільтрація, рубрикація новинних стрічок, адресна реклама, адресний маркетинг для електронної торгівлі.
- Автоматизація виробництва: оптимізація режимів виробничого процесу, комплексна діагностика якості продукції (ультразвук, оптика, гамма-випромінювання, ...), моніторинг і візуалізація багатовимірної диспетчерської інформації, попередження аварійних ситуацій, робототехніка.
- Політичні технології: аналіз і узагальнення соціологічних опитувань, передбачення динаміки рейтингів, виявлення значущих факторів, об'єктивна кластеризація електорату, візуалізація соціальної динаміки населення.
- Безпека та охоронні системи: системи ідентифікації особистості, розпізнавання голосу, осіб у натовпі, розпізнавання автомобільних номерів, аналіз аеро-космічних знімків, моніторинг інформаційних потоків, виявлення підробок.
- Введення і обробка інформації: Обробка рукописних чеків, розпізнавання підписів, відбитків пальців і голоси. Введення в комп'ютер фінансових та податкових документів.
- Геологорозвідка: аналіз сейсмічних даних, асоціативні методики пошуку корисних копалин, оцінка ресурсів родовищ.

Існує декілька широко поширених комерційних універсальних нейромережевих програмних пакетів (Statistica Neural Networks, NeuroShell, Matlab Neural Network Toolbox, NeuroSolutions, BrainMaker). Спеціалізованих, некомерційних чи розроблених вченими-дослідниками для власних потреб нейропрограмм набагато більше.

Нейромережі широко використовуються в даний час: нейромережі - це не що інше, як новий інструмент аналізу даних. І краще за інших ним може скористатися саме фахівець у своїй предметній області. Основні труднощі на шляху ще більш широкого розповсюдження нейротехнологій - у невмінні

широкого кола професіоналів формулювати свої проблеми в термінах, що допускають просте нейромережне рішення.

Основними цікавими на практиці можливостями нейронних мереж є такі:

- Існування швидких алгоритмів навчання: нейронна мережу навіть при сотнях вхідних сигналів і десятках-сотнях тисяч еталонних ситуацій може бути швидко навчена на звичайному комп'ютері. Тому нейронні мережі мають широке коло застосування і дозволяють вирішувати складні завдання прогнозу, класифікації або діагностики.
- Можливість роботи за наявності великої кількості неінформативних, шумових вхідних сигналів - попереднього їх відсіву робити не потрібно, нейронна мережа сама визначить їх малопридатними для вирішення завдання і може їх явно відкинути.
- Можливість роботи зі корельованими незалежними змінними, з різнотипною інформацією - дискретнозначній, кількісної та якісної, що часто приносить утруднення методам статистики
- Нейронна мережа одночасно може вирішувати кілька завдань на єдиному наборі вхідних сигналів - маючи кілька виходів, прогнозувати значення декількох показників.

Алгоритми навчання накладають досить мало вимог на структуру нейронної мережі та властивості її нейронів. Тому при наявності експертних знань або у випадку спеціальних вимог можна цілеспрямовано вибирати вигляд і властивості нейронів і нейронної мережі, збирати структуру нейронної мережі вручну, з окремих елементів, і задавати для кожного з них потрібні властивості.

1.2 Методи обчислення максимальних і мінімальних значень

Оскільки всі штучні нейронні мережі базуються на концепції нейронів, з'єднань та передатних функцій, існує подібність між різними структурами або архітектурами нейронних мереж. Більшість змін походить з різних правил навчання. Розглянемо деякі з найвідоміших штучних нейромереж.

Перцептрон Розенбалата

Першою моделлю нейромереж вважають перцептрон Розенбалата. Основою для багатьох типів штучних нейромереж прямого поширення є теорія перцептронів яка є класикою для вивчення.

Одношаровий перцептрон здатний розпізнавати найпростіші образи. Обчислювання зваженої суми сигналів вхідних елементів виконує окремий нейрон, який віднімає значення зсуву і пропускає результат через жорстку порогову функцію, вихід якої дорівнює $+1$ чи -1. В залежності від значення вихідного сигналу приймається рішення:

- $+1$ - вхідний сигнал належить до класу A,
- -1 - вхідний сигнал належить до класу B.

Вирішальні області визначають, які вхідні образи будуть віднесені до класу A, які - до класу B. Перцептрон, що складається з одного нейрона, формує дві вирішальні області, які розділено гіперплощиною.

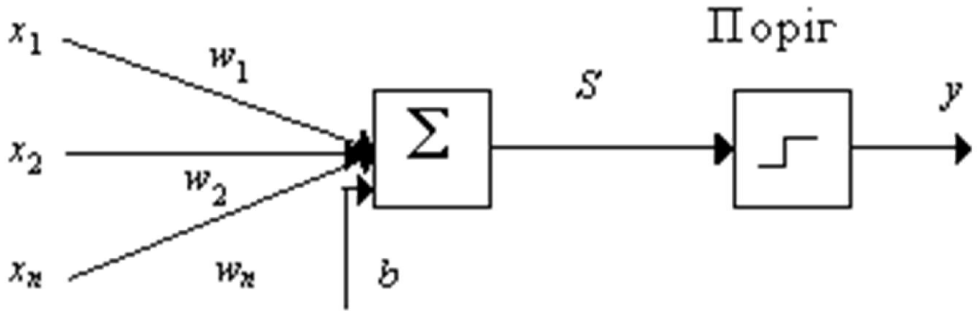


Рисунок 1.2.1. Схема нейрона

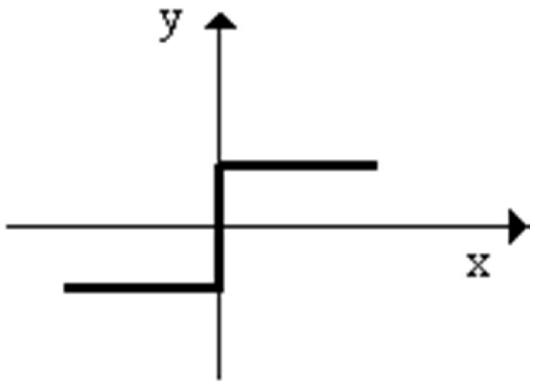


Рисунок 1.2.2. Графік передатної функції

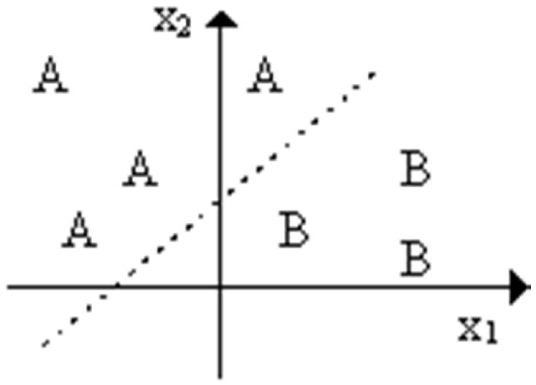


Рисунок 1.2.3. Поділяюча поверхня

На рисунку показано випадок з розмірністю вихідного сигналу - 2. Поділяюча поверхня є прямою лінією на площині. Від значень синаптичних ваг і зсуву залежить рівняння, що задає поділяючу пряму.

Нейромережа зворотного поширення похибки (Back Propagation)

Архітектура FeedForward BackPropagation була розроблена на початку 1970-х років декількома незалежними авторами: Вербор (Werbor); Паркер (Parker); Румельгарт (Rumelhart), Хінтон (Hinton) та Вільямс (Williams). На даний час, популярною, ефективною та легкою моделю навчання для складних, багатошарових мереж є парадигма BackPropagation. ЇЇ використання в різних типах застосувань породила великий клас нейромереж з різними структурами та методами навчання.

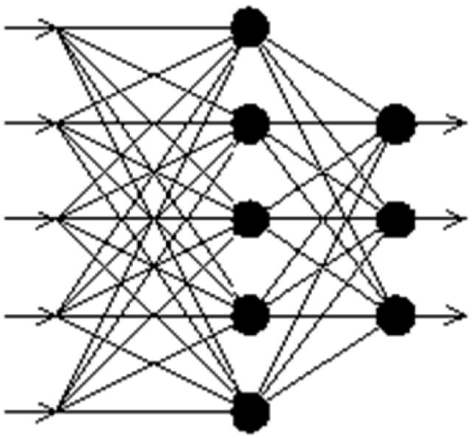


Рисунок 1.2.4. Мережа зворотного поширення похибки

Типова мережа BackPropagation має вхідний прошарок, вихідний прошарок та принаймні один прихований прошарок. Теоретично, обмежень

відносно числа прихованих прошарків не існує, але практично застосовують один або два.

В пошарову структуру з прямою передачею (вперед) сигналу організовано нейрони. Кожний нейрон мережі продукує зважену суму своїх входів, пропускає цю величину через передатну функцію і видає вихідне значення. Практично будь якої складності мережа може моделювати функцію, причому число прошарків і число нейронів у кожному прошарку визначають складність функції.

Визначення числа проміжних прошарків і числа нейронів в них є важливим аспектом при моделюванні мережі. Більшість дослідників та інженерів використовують загальні правила, зокрема:

- Кількість входів та виходів мережі визначаються кількістю вхідних та вихідних параметрів досліджуваного об'єкту, явища, процесу, тощо.
- Якщо складність у відношенні між отриманими та бажаними даними на виході збільшується, кількість нейронів прихованого прошарку повинна також збільшитись.
- Якщо процес, що моделюється, може розділятись на багато етапів, потрібен додатковий прихований прошарок (прошарки). Якщо процес не розділяється на етапи, тоді додаткові прошарки можуть допустити перезапам'ятовування і, відповідно, невірне загальне рішення.

Після врахування всіх правил, визначається число прошарків і число нейронів в кожному з них, потрібно знайти значення для синаптичних ваг і порогів мережі, які спроможні мінімізувати похибку спродукованого результату. Саме для цього існують алгоритми навчання, де відбувається підгонка моделі мережі до наявних навчальних даних. Шляхом проходження через мережу всіх навчальних прикладів і порівняння спродукованих вихідних значень з бажаними значеннями визначається похибка для конкретної моделі мережі. Множина похибок створює функцію похибок, значення якої можна розглядати, як похибку мережі. Найчастіше суму квадратів похибок використовують в якості функції похибок.

Розглянувши поняття поверхні станів можна краще зрозуміти алгоритм навчання мережі Back Propagation . Кожному значенню синаптичних ваг і порогів мережі (вільних параметрів моделі кількістю N) відповідає один вимір в багатовимірному просторі. $N+1$-ий вимір відповідає похибці мережі. Для різноманітних сполучень ваг відповідну похибку мережі можна зобразити точкою в $N+1$-вимірному просторі, всі ці точки утворюють деяку поверхню - поверхню станів. Знаходження на багатовимірній поверхні найнижчої точки є метою навчання нейромережі.

Поверхня станів має складну будову і досить неприємні властивості, зокрема, наявність локальних мінімумів (точки, найнижчі в своєму певному околі, але вищі від глобального мінімуму), пласкі ділянки, сідлові точки і довгі вузькі яри. Визначити розташування глобального мінімуму на поверхні станів неможливо аналітичними засобами, тому навчання нейромережі по суті полягає в дослідженні цієї поверхні.

Враховуючи початкові конфігурації ваг і порогів (від випадково обраної точки на поверхні), алгоритм навчання поступово відшукує глобальний мінімум. Обчислюється вектор градієнту поверхні похибок, який вказує напрямок найкоротшого спуску по поверхні з заданої точки. Для зменшення похибки потрібно трошки просунутись по ньому. Зрештою алгоритм зупиняється в нижній точці, що може виявитись лише локальним мінімумом (в ідеальному випадку - глобальним мінімумом).

Складність полягає у виборі довжини кроків. При великій довжині кроку збіжність буде швидшою, але є небезпека перестрибнути рішення, або піти в неправильному напрямку. При маленькому кроці, правильний напрямок буде виявлено, але зростає кількість ітерацій. На практиці розмір кроку береться пропорційним крутизні схилу з деякою константою, якою є швидкістю навчання. Правильний вибір швидкості навчання залежить від конкретної задачі і здійснюється дослідним шляхом. Ця константа може також залежати від часу, зменшуючись по мірі просування алгоритму.

Алгоритм діє ітеративне, його кроки називаються епохами. На вхід мережі по черзі подаються всі навчальні приклади на кожній епосі, вихідні

значення мережі порівнюються з бажаними значеннями і обчислюється похибка. Значення похибки, а також градієнту поверхні станів використовують для корекції ваг, і дії повторюються. Коли пройдена визначена кількість епох, або коли похибка досягає визначеного рівня малості, або коли похибка перестає зменшуватись (користувач переважно сам вибирає потрібний критерій останову) процес навчання припиняється.

Мережа Кохонена

На початку 1980-х рр. розроблена мережа Тойво Кохоненом яка принципово відрізняється від розглянутих вище мереж, відмінісьть полягає в тому що використовується неконтрольоване навчання і навчальна множина складається лише із значень вхідних змінних.

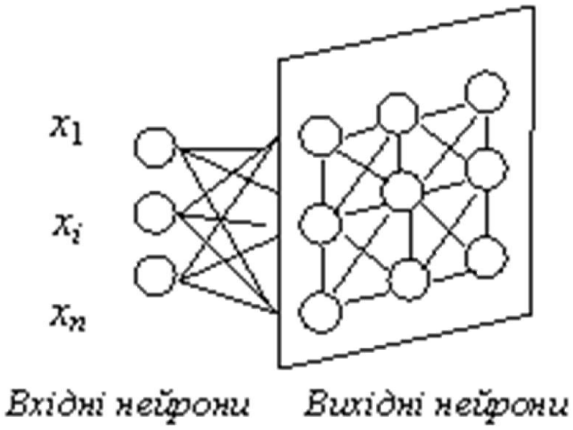


Рисунок 1.2.5. Мережа Кохонена

Мережа в навчальних даних розпізнає кластери далі відбувається розподіл даних до відповідних кластерів. Якщо в наступному мережа зустрічається з набором даних, несхожим ні з одним із відомих зразків, вона відносить його до нового кластеру. Задачі класифікації спроможна вирішувати мережа якщо в даних містяться мітки класів. Мережі Кохонена можна використовувати і в задачах, де класи відомими - перевага буде у спроможності мережі виявляти подібність між різноманітними класами.

Мережа Кохонена має лише два прошарки: вхідний і вихідний. Елементи карти розташовуються в деякому просторі, як правило, двовимірному. Навчання мережі Кохонена відбувається методом послідовних наближень. При

навчанні на входи подаються дані, але при цьому мережа підлаштовується не під еталонне значення виходу, а під закономірності у вхідних даних. З вибраного випадковим чином вихідного розташування центрів, починається навчання.

В процесі послідовної подачі на вхід мережі навчальних прикладів визначається найбільш схожий нейрон (той, у якого скалярний добуток ваг і поданого на вхід вектора є мінімальним). Цей нейрон оголошується переможцем і є центром при підлаштуванні ваг в сусідніх нейронів. Таке правило навчання передбачає "змагальне" навчання з врахуванням відстані нейронів від "нейрона-переможця".

Для найбільшого збігу з вхідними даними навчання полягає не в мінімізації помилки, а в підлаштуванні ваг (внутрішніх параметрів нейронної мережі).

Основний ітераційний алгоритм Кохонена послідовно проходить ряд епох, на кожній з яких обробляється один приклад з навчальної вибірки. Вхідні сигнали послідовно пред'являються мережі, при цьому бажані вихідні сигнали не визначаються. Після пред'явлення достатнього числа вхідних векторів синаптичні ваги мережі стають здатні визначити кластери. Ваги організуються так, що топологічно близькі вузли реагують до схожих вхідних сигналів.

Центр кластера встановлюється в певній позиції в результаті роботи алгоритму, яка задовольняє кластеризовані приклади, для яких даний нейрон є "переможцем". В результаті навчання мережі потрібно визначити міру сусідства нейронів, тобто окіл нейрона-переможця, який представляє кілька нейронів, що оточують нейрон-переможець.

Спочатку до околу належить велике число нейронів, далі її розмір поступово зменшується. Мережа формує топологічну структуру, в якій схожі приклади утворюють групи прикладів, які близько знаходяться на топологічної карті.

Мережа Хопфілда

У 1982 р. в Національній Академії Наук вперше Джон Хопфілд представив свою асоціативну мережу. Тому мережна парадигма згадується як

мережа Хопфілда, на честь Хопфілда та нового підходу до моделювання. На аналогії фізики динамічних систем базується мережа Хопфілда. Початкові застосування мережі включали асоціативну, або адресовану за змістом пам'ять та вирішували задачі оптимізації.

В мережі Хопфілда використовується три прошарки: вхідний, прошарок Хопфілда та вихідний прошарок. Кількість нейронів в кожному прошароку однакова. Виходи нейронів вхідного прошарку надходять до входів відповідних нейронів прошарку Хопфілда. Тут, зв’язки мають фіксовані вагові коефіцієнти. Виходи прошарку Хопфілда під'єднуються до входів всіх нейронів прошарку Хопфілда, за винятком самого себе, а також до відповідних елементів у вихідному прошарку. Мережа скеровує дані з вхідного прошарку до прошарку Хопфілда, скеровування відбувається під час навчання. Поки не буде завершена певна кількість циклів прошарок Хопфілда коливається, і біжучий стан сигналів нейронів прошарку передається на вихідний прошарок. Цей стан відповідає образу, який буде запам’ятовано в мережі.

Навчання мережі вимагає, щоб на вхідному та вихідному прошарках навчальний образ був представлений одночасно. Рекурсивний характер прошарку Хопфілда забезпечує засоби корекції всіх ваг з'єднань. Відповідні пари "вхід-вихід" мають відрізнятися між собою для правильного навчання мережі.

Якщо мережа Хопфілда використовується як пам'ять, що адресується за змістом вона має два головних обмеження.

Строге обмеження для числа образів, що можна зберегти та точно відтворити. Якщо зберігається занадто багато образів, мережа може збігтись до нового неіснуючого образу, відмінному від всіх запрограмованих образів, або не збігтись взагалі. Приблизно 15% від числа нейронів межа ємності пам'яті для мережі у прошарку Хопфілда.

Прошарок Хопфілда може стати нестабільним, якщо навчальні приклади є занадто подібними. Нестабільним зразок образу вважається тоді, коли він застосовується за нульовий час і мережа збігається до деякого іншого образу з

навчальної множини. За допомогою вибору більш ортогональних між собою навчальних прикладів, ця проблема може бути вирішена.

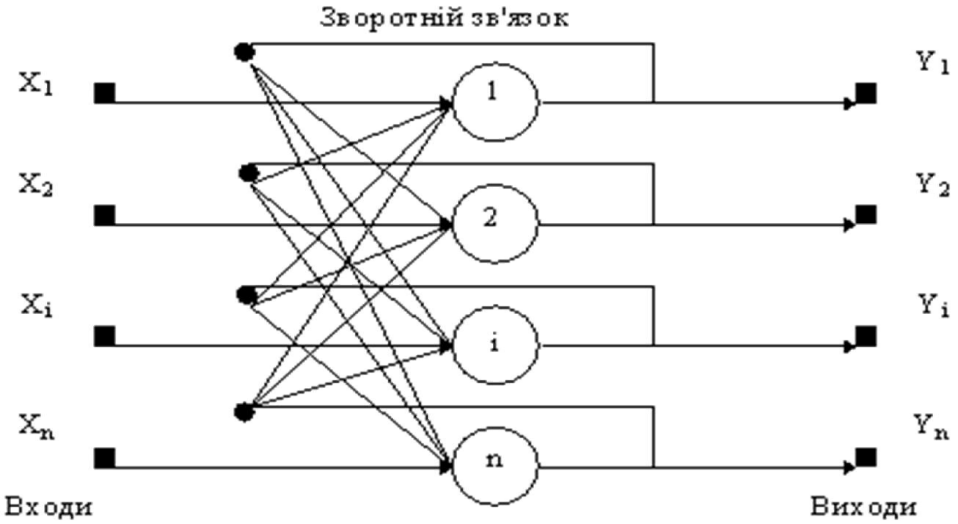


Рисунок 1.2.6. Мережа Хопфілда

Є деякий набір двійкових сигналів (зображень, звукових оцифровок, інших даних, що описують об'єкти або характеристики процесів) для вирішення завдання асоціативної пам'яті, який вважається зразковим. Мережа повинна вміти з зашумленого сигналу, поданого на її вхід, виділити ("пригадати" за частковою інформацією) відповідний зразок або "дати висновок" про те, що вхідні дані не відповідають жодному із зразків.

В загальному випадку, будь-який сигнал може бути описаний вектором $x1, xi, xn, ..., n$ - число нейронів у мережі і величина вхідних і вихідних векторів. Кожний елемент xi дорівнює або $+1$, або -1. Позначимо вектор, що описує k - ий зразок, через Xk, а його компоненти, відповідно, - $xik, k = 0,...,m-1,m$ - число зразків. Якщо мережа розпізнає (або "пригадує") якийсь зразок на основі пред'явлених їй даних, її виходи будуть містити саме його, тобто $Y = Xk$, де Y - вектор вихідних значень мережі: $y1, yi, yn$. У противному випадку, вихідний вектор не збігається з жодний зразковим.

Якщо, наприклад, сигнали є певним образом зображення, то, відобразивши у графічному виді дані з виходу мережі, можна буде побачити

картинку, що цілком збігається з однієї зі зразкових (у випадку успіху) або ж "вільну імпровізацію" мережі (у випадку невдачі).

Мережа Хемінга

Розширенням мережі Хопфілда є мережа Хемінга (Hamming). У середині 80-х рр. ця мережа була розроблена Річардом Ліппманом (Richard Lippman). Мережа Хемінга реалізує класифікатор, що базується на найменшій похибці для векторів двійкових входів, де похибка визначається відстанню Хемінга. Число бітів, які відрізняються між двома відповідними вхідними векторами фіксованої довжинит визначається як відстань Хемінга. Один вхідний вектор є незашумленим прикладом образу, інший є спотвореним образом. Вектор виходів навчальної множини є вектором класів, до яких належать образи. У режимі навчання вхідні вектори розподіляються до категорій для яких відстань між зразковими вхідними векторами та біжучим вхідним вектором є мінімальною.

Мережа Хемінга має три прошарки: вхідний прошарок з кількістю вузлів, скільки є окремих двійкових ознак; прошарок категорій (прошарок Хопфілда), з кількістю вузлів, скільки є категорій або класів; вихідний прошарок, який відповідає числу вузлів у прошарку категорій.

Мережа є простою архітектурою прямого поширення з вхідним рівнем, повністю під'єднаним до прошарку категорій. Кожен нейрон у прошарку категорій є зворотно під'єднаним до кожного нейрона у тому ж самому прошарку і прямо під'єднаним до вихідного нейрону. Через конкуренцію формується вихід з прошарку категорій до вихідного прошарку.

Навчання мережі Хемінга є подібним до методології Хопфілда. На вхідний прошарок надходить бажаний навчальний образ, а на виході вихідного прошарку надходить значення бажаного класу, до якого належить вектор. Вихід містить лише значення класу до якої належить вхідний вектор. Рекурсивний характер прошарку Хопфілда забезпечує засоби корекції всіх ваг з'єднань.

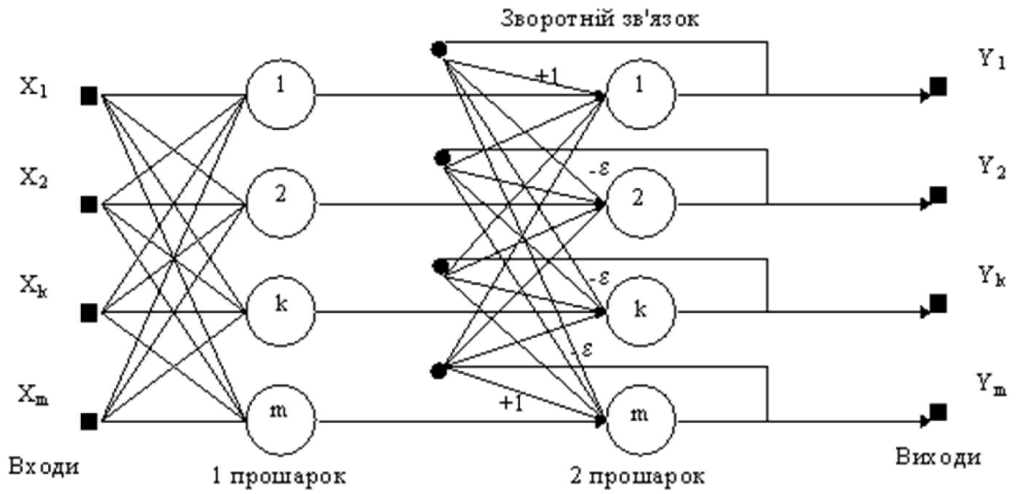


Рисунок 1.2.7. Мережа Хемінга

1.3 Структури апаратних засобів обчислення максимальних і мінімальних значень

При реалізації пристроїв обміну необхідно враховувати принципи обміну інформацією. Синхронним і асинхронним може бути обмін інформацією. Вибір принципу обміну є важливим аспектом який впливає не тільки на пропускну здатність, але й на фізичну довжину каналу обміну і на кількість пристроїв, що можуть до нього приєднуватись.

Синхронний принцип обміну

Обмін дані передаються блоками з постійною прив’язкою по часу при використанні цього принципу. При цьому передача даних базується на узгодженні таймерів пристрою-передавача і пристрою-приймача. Синхросигнали задають певний інтервал часу за який зчитується інформація з каналу обміну. Для початку синхронізації використовуються спеціальні сигнали синхронізації. Розміщення сигналів адреси і даних відносно сигналів синхронізації визначає Фіксований протокол. Для того щоб вирішити що робити в наступний момент часу додаткової логіки практично не потрібно, тому синхронна передача може бути швидкою і дешевою. Канал обміну більш ефективно використовується при синхронній передачі, підтримується висока ефективність, швидкодія передачі даних та вбудований надійний механізм

визначення помилок. Основний недолік полягає в тому що через проблему перекосу синхросигналів, шини при синхронній передачі не можуть бути довгими та інтерфейсне обладнання є більш дорогим і складним.

Асинхронний принцип обміну

Передача виконується без прив'язки до часу. Можуть виникати додаткові затримки обміну даними за рахунок відсутності зв'язку моментів звернення до пристрою-передавача та моментів видачі ним готових даних. Для виділення блоків даних на початку і вкінці кожного з них записується службова інформація. Цей принцип дозволяє простіше підключати до шини різноманітні за своїми характеристиками пристрої та такий принцип має кращу завадостійкість. Не переживаючи за перекіс сигналів синхронізації можна збільшити довжину шини. Одним з недоліків є те, що порівняно з синхронною передачею невисока швидкість передачі даних через передачу великої кількість службової інформації. Асинхронний принцип передачі даних доцільно використовувати в системах, де обмін даними не є постійним і не вимагається висока швидкість передачі даних.

Аналіз способів вирішення конфліктів в пристрої обміну

Коли декілька пристроїв в системі здійснюють одночасну передачу даних в пристрої обміну можуть виникати конфлікти. Щоб вирішення ці конфлікти існують різні прийоми, зокрема: призначення кожному пристрою унікального пріоритету, виділення фіксованих часових інтервалів кожному пристрою та використання багатопортової пам'яті.

Вирішення конфліктів в багатопортовій пам'яті

В n-портовій пам'яті є n незалежних наборів шин адреси, даних і управління, що забезпечують одночасний і незалежний доступ до пам'яті n пристроям. Така властивість дозволяє суттєво спростити створення складних систем, але при цьому можуть виникати конфлікти і необхідно використовувати способи їх вирішення. Конфлікти виникають при звертанні декількох активних пристроїв до однієї комірки пам'яті при записі даних з декількох портів, або при записі через один порт і читанні через інші. Способи вирішення конфліктів в синхронній і асинхронній багатопортовій пам'яті (БП)

відрізняються. Синхронна БП передбачає взаємну синхронізацію активних пристроїв від одного системного таймера, тому для вирішення конфліктних ситуацій не передбачається використання додаткової логіки. В асинхронній БП для вирішення конфліктів використовується: арбітражна логіка, семафори, запити на переривання та система Master/Slave.

1.4 Постановка завдання на дипломну роботу

Існують різні способи побудови пристроїв обміну залежно від схем розподілу пам’яті: з спільною пам’яттю і розподіленою . При цьому виділяють дві основні моделі обміну: на базі передачі повідомлень (для розподіленої пам’яті) та з використанням спільної пам’яті.

Спільна пам’ять

Розподіл спільного простору пам’яті між усіма вхідними портами дозволяє ефективно використовувати її у відповідності до характеру вхідного потоку даних. При цьому всі основні обчислювальні вузли об’єднуються з пам’яттю через загальну шину. Перевагами такого підходу є те, що обмін здійснюється шляхом запису/зчитування інформації з комірок спільної пам’яті, які є доступні для всіх вузлів, а, отже, не витрачається час на пересилку даних. До недоліків відноситься можливість виникнення конфліктів при одночасному доступі в одну комірку пам’яті, проблема повільного звертання до оперативної пам’яті та її обмеженого об’єму і проблема масштабованості. Чим більша кількість вузлів використовується, тим більша вартість розробки та менша ефективність роботи.

Виділяють системи з однорідним і неоднорідним доступом до пам’яті. В системах з однорідним доступом всі вузли можуть здійснювати доступ до пам’яті за однаковий час. Такий спосіб організації обміну найбільш часто використовується для побудови пристроїв обміну. В системах з неоднорідним доступом до пам’яті кожному вузлу виділяється частина спільної пам’яті. Дана пам’ять має єдиний адресний простір, тому можливе звертання до будь-якої

комірки спільної пам’яті безпосередньо, використовуючи її адресу. Час доступу до модулів спільної пам’яті від різних вузлів є різною.

Розподілена пам’ять

В даному випадку кожен вузол має доступ до певної фіксованої кількості виділених комірок пам’яті. Для забезпечення обміну інформацією вузли пов’язані каналами зв’язку. Пакет, призначений для певного вихідного вузла втрачається при умові переповнення виділеного для даного вихідного вузла блоку пам’яті, хоча інші блоки в цей час можуть бути порожніми. Обмін в системі здійснюється шяхом посилання та отримання повідомлень. Така схема розподілу пам’яті використовується для задач, які потребують малого обміну даними та великих об’ємів пам’яті.

До переваг відноситься можливість масштабування, тобто можна об’єднати велику кількість вузлів без істотного зменшення ефективності їх взаємодії. При цьому вартість системи буде пропорційна кількості вузлів. До недоліків відноситься проблема обміну даними та велике енергоспоживання. Обмін даними в таких системах здійснюється дуже повільно порівняно з швидкодією обчислень (і з великими затримками). Тому неможливо ефективно розв’язати на таких системах задачі, що потребують інтенсивного обміну.

Постановка завдання.

1. Провести аналіз існуючих рішень в галузі обчислення максимальних і мінімальних значень чисел в масиві даних
2. Розробити структурну схему апаратного пристрою обчислення максимальних і мінімальних значень чисел в масиві даних

2 АЛГОРИТМИ І ПРОЦЕСОРНІ ЕЛЕМЕНТИ ДЛЯ ОБЧИСЛЕННЯ МАКСИМАЛЬНИХ І МІНІМАЛЬНИХ ЗНАЧЕНЬ

2.1 Формування вимог до пристроїв обчислення максимальних і мінімальних значень

Алгоритми визначення максимального та мінімального чисел для НВІС-реалізацій повинні забезпечувати детерміноване переміщення даних, бути добре структурованими та орієнтованими на реалізацію на множині взаємозв'язаних процесорних елементів (ПЕ). Від вимог, що висуваються до часу реалізації алгоритму залежить структура та операції, які виконують ПЕ. Переважно базові операції алгоритмів обчислення максимального та мінімального чисел реалізує ПЕ. Потрібно одночасно враховувати багато взаємопов'язаних факторів при розробці або виборі алгоритмів обчислення максимального та мінімального чисел.

Аалгоритми повині були рекурсивними та локально залежними. Всі ПЕ повинні виконувати приблизно однакові операції в рекурсивному алгоритмі. Кожний із ПЕ буде повторювати виконання фіксованого набору операцій над послідовністю даних, що надходять. Ефективність відображення алгоритму на ПЕ безпосередньо пов'язана зі способом декомпозиції розв'язання задачі та перетворення на незалежні базові операції, що виконуються паралельно, або на залежні, що виконуються у конвеєрному режимі.

В декартовій системі координат можна утворити точкові системи (решітки) з множини ПЕ, які є моделлю паралельних апаратних структур [23]. Така модель дозволяє оцінити часову та апаратну складність реалізації алгоритму. В решітковій моделі кожному ПЕ ставиться у відповідність часовий *i* та просторовий *j* індекси, які вказують коли і де виконується кожна із операцій алгоритму. В алгоритмах з локальними пересилками даних різниця між просторовими індексами *j* на кроці рекурсії обмежена деякою константою, оскільки в таких алгоритмах обміни здійснюються тільки між сусідніми ПЕ. До класу алгоритмів з глобальними зв'язками відносяться алгоритми, які при рекурсії мають рознесені просторові індекси.

Забезпечення високої швидкодії є однією з основних вимог, що ставиться до пристроїв визначення максимального та мінімального чисел із масиву чисел. Проблема виникає, при використанні пристроїв для розв'язання задач в реальному часі, який накладає певні обмеження на процес визначення максимального та мінімального чисел із масиву чисел. В першу чергу, ці обмеження пов'язані з часом розв'язання задачі T_p, який не повинен перевищувати часу обміну повідомленнями $T_{обм}$, тобто:

$$T_p \le T_{обм} \tag{2.1.1}$$

Час обміну залежить як від обсягу масиву N, розрядності n і частоти надходження вхідних даних F_d, так і від кількості k каналів та їх розрядності n_k. Такий час визначається за формулою:

$$T_{обм} = \frac{Nn}{F_d k n_k} \tag{2.1.2}$$

Одним із основних інтегральних параметрів оцінки НВІС-пристроїв обчислення максимального та мінімального чисел із масиву чисел є ефективність використання обладнання, який враховує кількість виводів інтерфейсу, однорідність структури, кількість і локальність зв'язків, зв'язує продуктивність з витратами обладнання та дає оцінку елементам пристрою за продуктивністю [23]. Кількісна величина ефективності використання обладнання визначається так:

$$E = \frac{R}{t_o(k_1 \sum_{i=1}^{s} W_{ПЕ_i} d_i + k_2 Q + k_3 Y)} \tag{2.1.3}$$

де R - складність алгоритму обчислення максимального та мінімального чисел; t_o - час обчислення максимального та мінімального чисел; $W_{ПЕ_i}$ - витрати

обладнання на реалізацію i-го процесорного елемента; d_i - кількість функціональних вузлів i-го типу; k_1 - коефіцієнт врахування однорідності $k_1 = f(s)$; s - кількість видів функціональних вузлів; Q - загальна кількість зв'язків; k_2 - коефіцієнт врахування регулярності зв'язків $k_2 = f(\Delta j)$; Δj - просторова зв'язкова віддаіль; Y - кількість виводів інтерфейсу; k_3 - коефіцієнт врахування кількості виводів інтерфейсу зв'язку $k_3 = f(Y)$.

При апаратній реалізації алгоритмів визначення максимального та мінімального чисел із масиву чисел висока ефективність використання обладнання досягається узгодженням інтенсивності надходження даних $P_d = knF_d$, де k - кількість каналів надходження даних; n - розрядність каналів надходження даних; F_d - частота надходження даних, із інтенсивністю обчислень (обчислювальною здатністю) апаратних засобів, яку визнають так[37]:

$$D_k = \frac{mn_m}{T_k} \tag{2.1.4}$$

де m - кількість каналів надходження даних у сходинках конвеєра; n_m - розрядність каналів надходження даних у сходинках конвеєра; T_k - такт конвеєра.

Задача розроблення пристроїв для визначення максимального та мінімального чисел із масиву чисел, орієнтованих на НВІС-реалізацію, з високою ефективністю використання обладнання зводиться до мінімізації апаратних затрат, кількості виводів інтерфейсу, збільшення однорідності структури та регулярності зв'язків при забезпечені режиму реального часу. В основу розроблення таких пристроїв доцільно покласти наступні принципи для найповнішого використання переваг сучасної НВІС-технології та забезпечення даних вимог [23-38]:

- однорідності та регулярності структури;

- локалізації та спрощення зв’язків між елементами;
- модульності побудови;
- конвеєризації та просторового паралелізму опрацювання даних;
- узгодженості інтенсивності надходження даних із інтенсивністю обчислень в пристрої;
- програмування архітектури пристрою шляхом використання програмованих логічних інтегральних схем.

2.2 Вибір принципів та елементної бази для апаратної реалізації обчислення максимальних і мінімальних значень

Пропонується розробку нейроорієнтованих комп’ютерних систем здійснювати на основі інтегрованого підходу, який охоплює :

- сучасну елементу базу, апаратні та програмні комп’ютерні засоби;
- нейромережеві методи та алгоритми;
- обчислювальні методи, алгоритми та НВІС-структури для реалізації базових операцій нейроалгоритмів.

Необхідно в основу побудови нейроорієнтованих комп’ютерних систем покласти принципи, які дозволять зменшити вартість, терміни і розширити галузі їх застосування. Аналіз показує, що забезпечити дані вимоги можна при використанні таких принципів побудови :

- змінного складу обладнання, що передбачає наявність обчислювального ядра та змінних спеціалізованих апаратно-програмних модулів, за допомогою яких ядро адаптується до вимог конкретного застосування;
- модульності, який передбачає розробку компонентів нейроорієнтованих комп’ютерних систем у вигляді модулів, що мають вихід на стандартний інтерфейс;
- конвеєризації та просторового паралелізму обробки даних у спеціалізованих апаратно- програмних модулях;

- відкритості програмного забезпечення, що передбачає можливості нарощування та його вдосконалення, максимального використанням стандартних драйверів та програмних засобів;
- узгодженості та адаптації апаратно-програмних спеціалізованих модулів до інтенсивності надходження даних і структури нейроалгоритмів;
- програмованості архітектури шляхом використання репрограмованих логічних інтегральних мікросхем.

2.3 Алгоритм та структура процесорного елементу для одночасного обчислення максимальних і мінімальних чисел із однорідного масиву чисел

Комп'ютерної системи можна представити у вигляді постійної частини Б - універсального обчислювального ядра та змінної частини V - спеціалізованих модулів, які реалізують базові операції нейроалгоритмів. Структура нейроорієнтованої комп'ютерної системи наведена на рис.3, де БПП - багатопортова пам'ять.

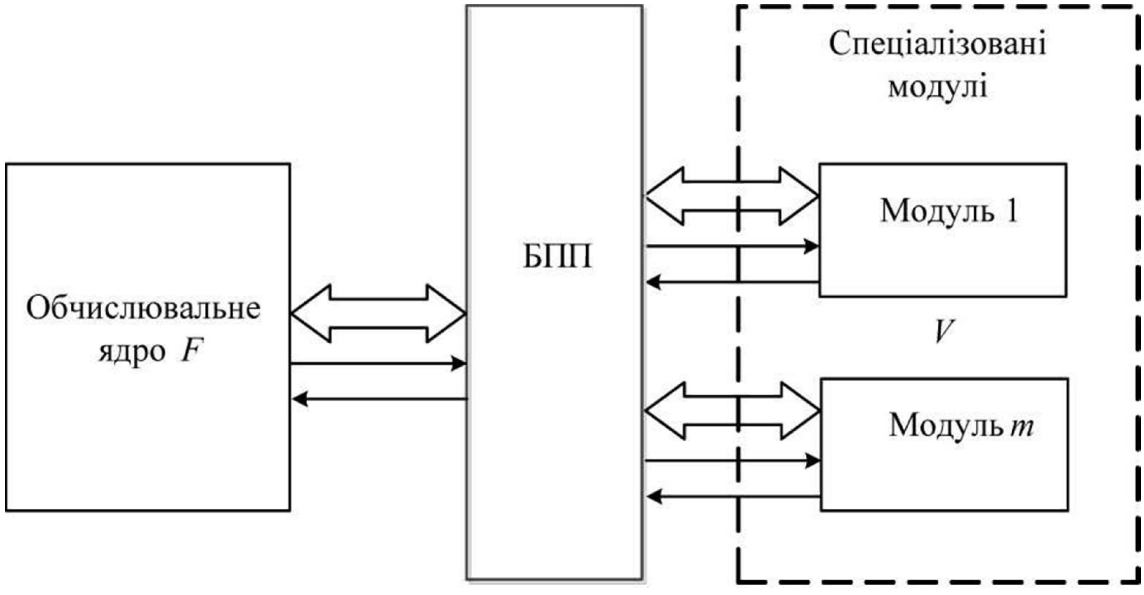


Рисунок 2.3.1. Структура нейроорієнтованої комп'ютерної системи

Базовими компонентами нейроорієнтованої комп'ютерної системи є обчислювальне ядро, набір спеціалізованих модулів і БПП. Паралелізм обробки даних у нейроорієнтованій комп'ютерній системі висуває свої вимоги до організації обміну між обчислювальним ядром і набором спеціалізованих модулів. Такий обмін у нейроорієнтованій комп'ютерній системі доцільно здійснювати використовуючи БПП, яка забезпечує паралельний доступу до множини даних як із сторони обчислювального ядра, так і сторони спеціалізованих модулів. Нейропроцесор (нейрочіп) є основою обчислювального ядра нейроорієнтованої комп'ютерної системи, він має набір команд добре пристосованих для виконання базових операцій нейроалгоритмів, а також повний набір команд загального призначення. Переважна більшість алгоритмів нейропарадигм зводиться до виконання обмеженого набору базових операцій типу "додавання - множення".

Основними перевагами нейрочіпів є:

- відносно більша швидкодія (порівняно з CPU);
- полегшена реалізація зв'язків «всі зі всіма» (для розробника нейромереж);
- низьке споживання електроенергії;
- відносно доступна ціна.
- Основні недоліки:
- велика структурна складність і низька надійність системи;
- велика складність ефективної реалізації процедури навчання, самонавчання, самоорганізації;
- значне збільшення споживаної потужності та втрата швидкодії під час збільшення ступеню інтеграції нейрочіпів;
- жорстко задана наперед топологія.

Особливостями архітектури і технічними характеристиками нейропроцесора визначаються основні характеристики обчислювального ядра нейроорієнтованої комп'ютерної системи. До таких характеристик відносяться: довжина інформаційного слова; число основних команд і час їх виконання;

ємність пам’яті, що може адресуватися; ємності внутрішньокристальної пам’яті даних і програм та кількість внутрішніх регістрів.

Структура нейроорієнтованих комп’ютерних систем залежить від конкретних вимог і множини нейроалгоритмів (N), які використовуються для розв’язання задач. При розв’язанні конкретної задачі здійснюється розподіл алгоритмів розв’язання задачі між обладнанням F і V

$$N = N_F + N_V \tag{2.3.1}$$

де N_F - множина алгоритмів, які виконуються на обладнанні F; N_V - множина алгоритмів, які виконуються на обладнанні V.

В залежності від співвідношення N_F і N_V комп’ютерні нейроорієнтовані системи діляться на такі типи:

з переважним використанням процесорного ядра (постійного обладнання F), коли

$$N_F \wedge N, N_V \wedge \wedge N_F >> N_V \tag{2.3.2}$$

з переважним використанням спеціалізованих модулів (змінної частини V), коли

$$N_F \wedge 0, N_V \wedge N, N_F << N_V \tag{2.3.3}$$

з рівномірним використанням постійного обладнання F і змінної частини V, коли $N_F \sim \mathrm{N_V}$.

Перший тип нейроорієнтованих комп’ютерних систем характеризується тим, що основний обсяг обчислювальних потужностей зосереджений в процесорному ядрі.

В другому типі нейроорієнтованих комп’ютерних систем основні обчислювальні алгоритми реалізуються з допомогою спеціалізованих модулів, а

процесорне ядро використовується для виконання допоміжних сервісних функцій.

Третій тип нейроорієнтованих комп'ютерних систем характеризується тим, що процесорне ядро забезпечує реалізацію алгоритмів управління, операцій введення-виведення та сервісних функцій, а спеціалізовані модулі - реалізують обчислювальні нейроалгоритми, які вимагають великого об'єму обчислень.

У більшості випадків для нейромережевого опрацювання даних і реалізації алгоритмів Data Mining потрібно здійснити нормалізацію вхідних даних. Нормалізація - це процедура попереднього опрацювання вхідних даних (навчальних, тестових і робочих вибірок), при якій значення ознак, які формують вхідний вектор, приводиться до деякого заданого діапазону зазвичай, цей діапазон [0, 1] або [-1, 1]. Існує велика кількість способів нормалізації вхідних значень. Найпростішою, але у більшості випадків ефективною, є лінійна нормалізація. Якщо початкові дані потрібно привести до діапазону [0, 1], то вона виконується так:

$$x_i^* = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \qquad (2.3.4)$$

Для приведення початкових даних до діапазону [-1, 1] лінійна нормалізація здійснюється таким чиом:

$$x_i^* = \frac{x_i}{x_{\max}} \qquad (2.3.5)$$

Якщо вхідні дані X щільно заповнюють певний інтервал, то використання лінійної нормалізації оптимальне, оскільки вона не потребує здійснення складних обчислень.

Основними операціями нормалізації даних є визначення максимального та мінімального чисел із масиву чисел. Для забезпечення опрацювання потоків

даних у режимі реального часу при реалізації таких операцій вимагається висока швидкодія, яку можна досягнути за допомогою розпаралелювання процесу обчислення та апаратною реалізацією. Тому актуальною проблемою є розроблення орієнтованих на апаратну реалізацію паралельних алгоритмів визначення максимального та мінімального чисел із масиву чисел.

Відомі методи визначення максимального та мінімального чисел із масиву чисел грунтуються на послідовному порівняні значення кожного числа, починаючи з другого, із поточним значенням максимального числа, яке на першому кроці приймає значення першого числа. Якщо значення числа більше за поточне значення максимального числа, то воно стає поточним максимальним значенням. Отже, на кожній ітерації обчислення у поточному значенні максимального числа міститиметься найбільше значення з пройденої частини масиву, а по завершенні обчислення поточне значення буде максимальним числом у всьому масиві[37-38].

Операції визначення максимального та мінімального чисел із масиву чисел мають ряд специфічних особливостей, які не дозволяють при їх реалізації використовувати відомі обчислювальні методи і алгоритми. Відомі апаратні засоби в основному орієнтовані на реалізацію алгоритмів з перевагою обчислювальних операцій над логічними і вони не враховують специфіки обчислення максимального та мінімального чисел із масиву. Особливістю сучасних апаратних засобів є неефективність використання багаторозрядних операційних пристроїв, що істотно знижує продуктивність та ефективність використання обладнання[39-42].

З аналізу літературних джерел можна зробити висновки, що недоліком відомих методів та алгоритмів є те, що вони не орієнтовані на апаратну реалізацію, а використання існуючих апаратних засобів для обчислення максимального та мінімального чисел із масиву чисел є малоефективним.

Тому метою роботи є розроблення алгоритмів і спеціалізованих НВІС-структур для визначення максимальних і мінімальних значень з високою ефективністю використаня обладнання.

Паралельні алгоритми та НВІС-структури пристроїв обчислення максимальних і мінімальних значень із масиву чисел. Аналіз методів і алгоритмів обчислення максимальних і мінімальних значень із масиву чисел показав, що для НВІС-реалізацій найефективнішими є алгоритмами, які грунтуються на методі порозрядного порівняння [42]. Обчислення максимального $A_{\max}$ і мінімального $A_{\min}$ чисел із групи чисел $A_1, A_2, ..., A_j, ...A_m$, за таким методом виконується послідовним порівнянням розрядів всіх чисел починаючи зі старшого. При кожному порівнянні отримуємо i-і розряди максимального і мінімального чисел, обчислення яких здійснюється за формулами:

$$\overline{A_{i\max}} = \bigwedge_{j=1}^{m} \overline{a_{ji} \wedge y_{ij}}, y_{1j} = 1, \tag{2.3.6}$$

$$A_{i\min} = \bigwedge_{j=1}^{m} \overline{\overline{a_{ji}} \wedge z_{ij}}, z_{1j} = 1, \tag{2.3.7}$$

Де y_{ij}, z_{ij} - i-і розряди j-х слів управління; a_{ji} - i-розряд j-о числа; m - кількість чисел у групі.

Формування $(i+1)-x$ розрядів j-х слів управління виконується за формулами (13) і (14) та виконання наступних логічних обчислень :

$$y_{(i+1)j} = (\overline{A}_{i\max} \vee x_{ji}) \wedge y_{ij}, \tag{2.3.8}$$

$$z_{(i+1)j} = (A_{i\min} \vee x_{ji}) \wedge z_{ij}, \tag{2.3.9}$$

Процес синтезу паралельних НВІС-структур для обчислення максимальних і мінімальних чисел із групи чисел зводиться до виконання наступних етапів:

- виділення базової операції;
- просторово-часове відображення алгоритму;

- розробка схеми процесорного елемента (ПЕ), що реалізує базову
- операцію алгоритму;
- синтез НВІС-структур на базі ПЕ;
- організація інтерфейсу НВІС.

Аналіз алгоритмів паралельного обчислення максимальних і мінімальних значень із групи чисел на основі методу порозрядного порівняння дозволив виділити для НВІС-реалізації базову операцію. Така базова операція включає формування розрядів слів управління за формулами $y_{(i+1)j} = (\overline{A}_{i\max} \vee x_{ji}) \wedge y_{ij}$ і $z_{(i+1)j} = (A_{i\min} \vee x_{ji}) \wedge z_{ij}$ та виконання наступних логічних обчислень:

$$\overline{A_{ji\max}} = \overline{a_{ji} \wedge y_{ij}}, \tag{2.3.10}$$

$$\overline{A_{ji\min}} = \overline{\overline{a}_{ji} \wedge z_{ij}}, \tag{2.3.11}$$

Виділена базова операція реалізується у вигляді ПЕ, схеми яких наведені на рис.4, де - *а* - ПЕ однотактного пристрою; б - ПЕ конвеєрного пристрою; *в* - ПЕ пристрою з вертикальним опрацюванням вхідних чисел.

Особливістю розроблених ПЕ є використання спільних шин результатів, підключення до яких здійснюється за допомогою логічних елементів *2І-НЕ* з відкритим колектором. Вертикальне підключення до спільних шин результатів забезпечує збільшення розміру масиву чисел, з якого визначаються максимальне і мінімальне значення. Використання спільних шин результатів забезпечує високу швидкодії, яка не залежить від кількості чисел у масиві, а залежить тільки від часу затримки на ПЕ.

Вартість НВІС-пристроїв обчислення максимальних і мінімальних значень із групи чисел в основному залежить від площі кристала, яка визначається як витратами обладнання (кількість транзисторів), так і кількістю зовнішніх виводів, число яких обмежене рівнем технології та розміром кристалу. Орієнтація структур сортування на НВІС-реалізацію вимагає

зменшення числа виводів інтерфейсу та кількості з'єднань між ПЕ. Забезпечити ці вимоги можна використанням паралельно-вертикального алгоритму обчислення максимальних і мінімальних значень із групи чисел, при якому надходження чисел і видача результатів здійснюється розрядними зрізами.

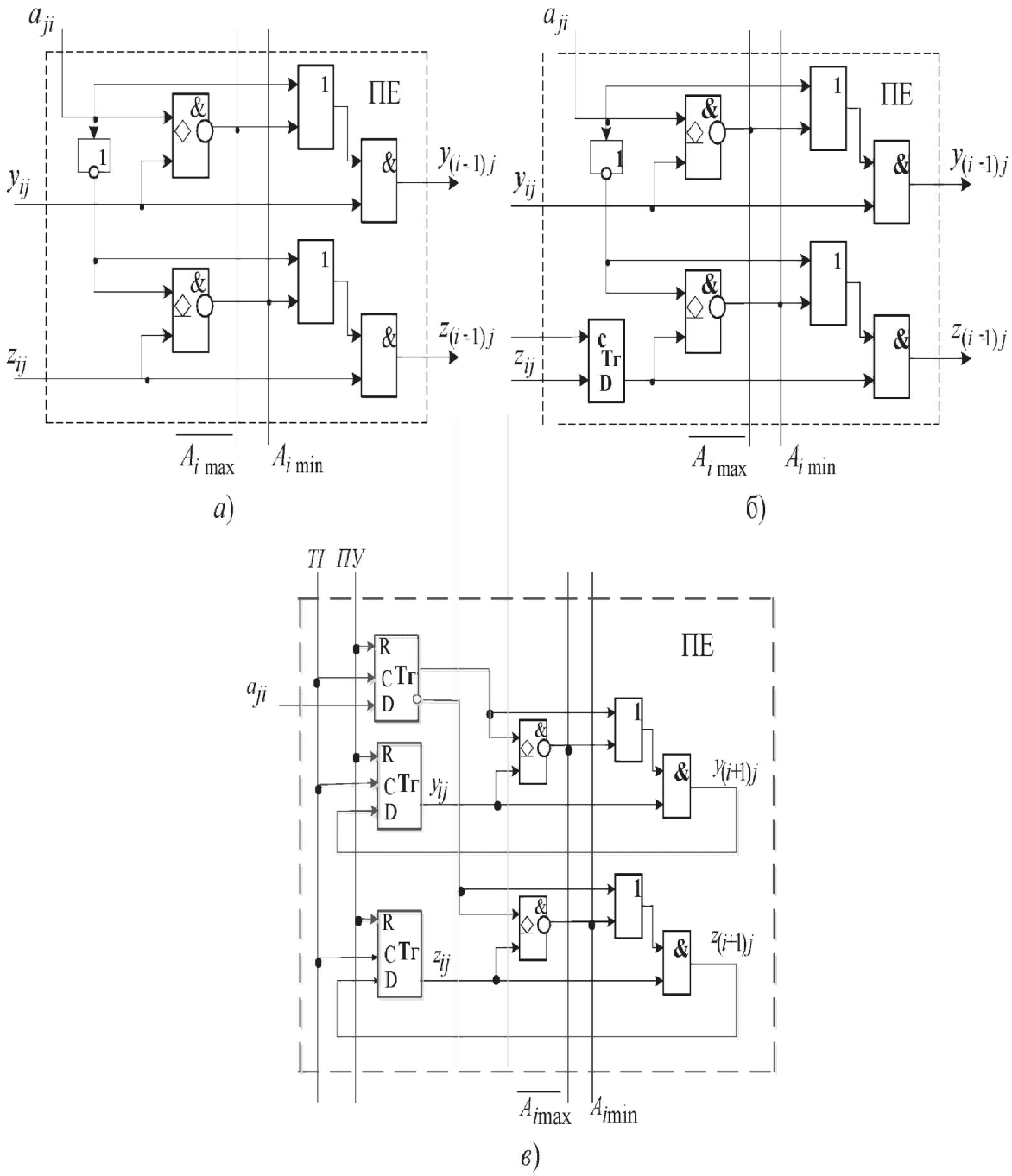


Рисунок 2.3.2, де - *а* - ПЕ однотактного пристрою; б - ПЕ конвеєрного пристрою; *в* - ПЕ пристрою з вертикальним опрацюванням вхідних чисел

Структура паралельно-вертикального НВІС-пристрою обчислення максимальних і мінімальних значень із групи чисел наведена на рисунку 2.3.3,

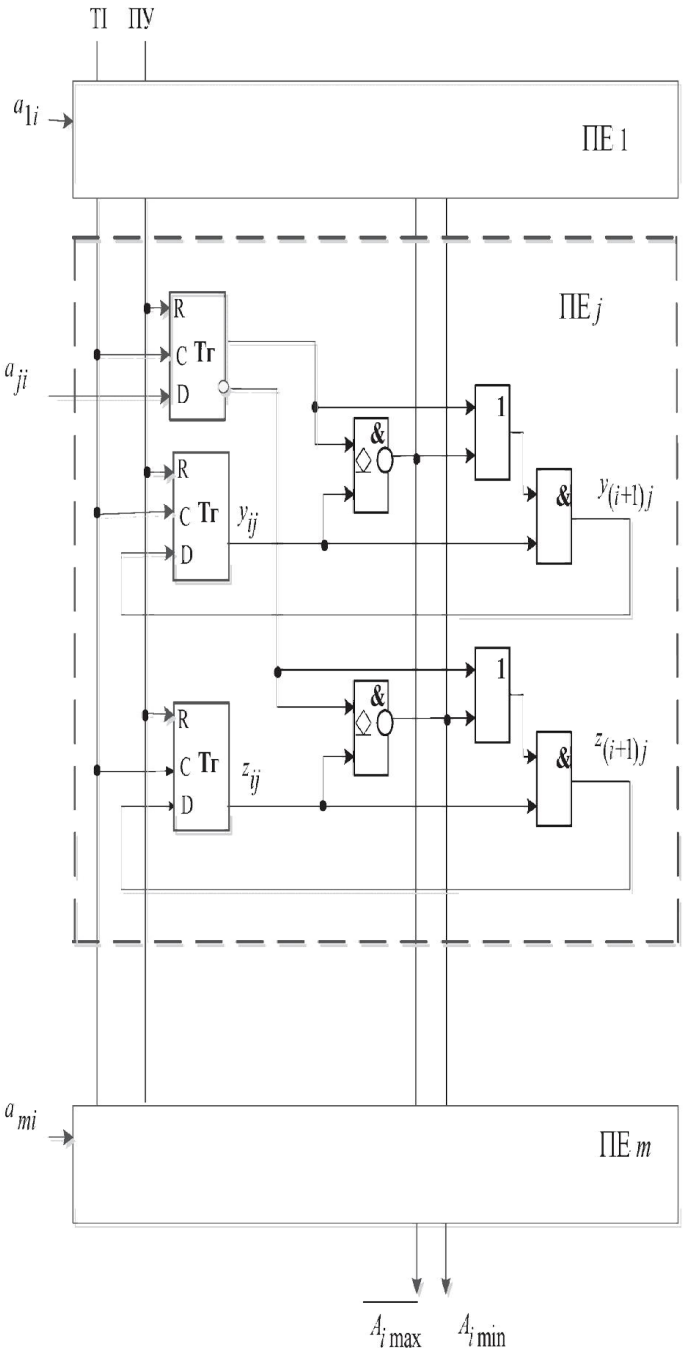


Рисунок 2.3.3. Схема паралельно-вертикального НВІС-пристрою обчислення максимальних і мінімаоьних значень

де TI - тактовий вхід, $ПУ$ - вхід початкової установки тригерів, $a_i,...,a_m$ - однорозрядні інформаційні входи, де m - кількість чисел, що порівнюються, $ПЕ_1,...,ПЕ_m$ - $ПЕ$ пристрою з вертикальним опрацюванням вхідних чисел, Тг1 і

Тг2 - D-тригери, $\overline{A_{\max}}$ і $A_{\min}$ - порозрядний вихід відповідно інверсного максимального та мінімального чисел.

Паралельно-вертикальний НВІС-пристрій обчислення максимальних і мінімальних значень працює наступним чином. Перед початком роботи імпульсом початкової установки, який надходить із входу ПУ, всі тригери пристрою, встановлюються у лог.0. При одночасному визначенні максимального та мінімального чисел в одновимірному масиві із m чисел дані на інформаційні входи надходить порозрядно старшими розрядами вперед. Так у i-у такті роботи пристрою в тригери даних з інформаційних входів запукається i-і розряди чисел, в тригери управління $-i$-і розряди y_{ij}, z_{ij} слів управління. За n тактів, де n-розрядність чисел, на виході $\overline{A_{\max}}$ отримуємо інверсне значення максимального числа, а на виході $A_{\min}$ – мінімальне значення числа із однорідного масиву з m чисел.

Час обчислення максимального та мінімального чисел із масиву з m рівний:

$$t_m = (t_{Tг} + 3t_I)n, \tag{2.3.12}$$

де $t_{Tг}$ - час запису інформації у тригер, t_I - час затримки інформації при проходженні через логічні елементи типу *АБО, І, І-НЕ.*

Затрати обладнання на реалізацію даного пристрою рівні:

$$W_m = (3W_{Tг} + 6W_I)m \tag{2.3.13}$$

де $W_{Tг}$ - затрати обладнання на реалізацію D-тригера, W_I - затрати обладнання на реалізацію логічних елементів типу *АБО, І, І-НЕ.*

Для опрацювання інтенсивніших потоків даних вимагається збільшення обчислювальної здатності, яке досягається за рахунок збільшення розрядності каналів надходження даних, яка може бути два і більше. Максимальна

швидкодія досягається коли розрядність каналів надходження даних дорівнює n, тобто коли одночасно опрацьовуються всі розряди чисел.

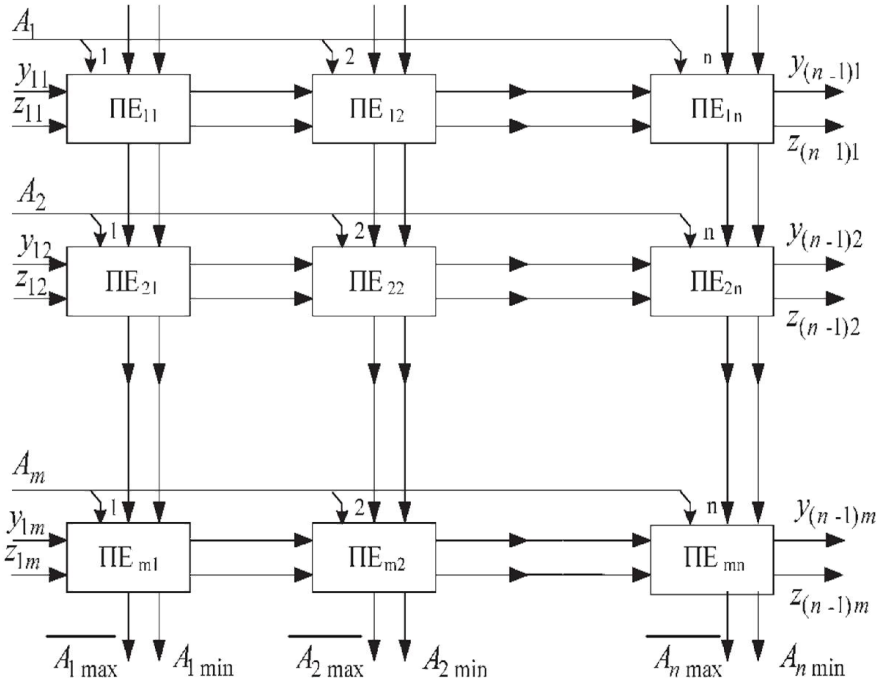


Рисунок 2.3.4. Матрична однотактна НВІС-структура пристрою обчислення максимальних і мінімальних значень із групи m чисел

На основі однотактного і конвеєрного ПЕ синтезовано відповідно матричні однотактні (рис.6) та конвеєрні (рис.7) паралельні НВІС-структури пристрою обчислення максимальних і мінімальних значень із групи чисел. Кожна із цих структур складається із матриці $(m \times n)$ ПЕ, в якій $ПЕ_{1i}, -ПЕ_{mi}$, - i-го стовпчика обчислюють відповідно до формул $\overline{A_{i\max}} = \bigwedge_{j=1}^{m} \overline{a_{ji} \wedge y_{ij}}, y_{1j} = 1,$ і $A_{i\min} = \bigwedge_{j=1}^{m} \overline{\overline{a_{ji}} \wedge z_{ij}}, z_{1j} = 1,$ значення i-х розрядів максимального $\overline{A_{\max}}$ і мінімального $A_{\min}$ чисел та формують у відповідності з формулами $y_{(i+1)j} = (\overline{A}_{i\max} \vee x_{ji}) \wedge y_{ij}$, і $y_{(i+1)j} = (\overline{A}_{i\max} \vee x_{ji}) \wedge y_{ij}, z_{(i+1)j} = (A_{i\min} \vee x_{ji}) \wedge z_{ij}$, значення $(i+1) - x$ слів управління $y(i+1)$ і $z(i+1)$.

Час обчислення максимальних і мінімальних значень у однотактному пристрої визначається за формулою:

$$T_O = 4nt_i, \tag{2.3.14}$$

де t_i - час спрацювання логічного елемента "I", n - розрядність чисел.

Апаратні витрати на реалізацію однотактного пристрою дорівнюють:

$$W_O = 7NnW_i, \tag{2.3.15}$$

де W_i - апаратні витрати на логічні елементи типу І, АБО, І-НЕ.

Особливістю конвеєрної НВІС-структури (рис.7) є введення у ПЕ тригерів та використання n блоків пам'яті типу FIFO. Кожний i-ий блок $FIFO_i$,- забезпечує затримку інформації на i тактів. Тривалість конвеєрного такту у такому пристрої визначається за наступною формулою:

$$T_K = t_{Tг} + 4nt_i, \tag{2.3.16}$$

де $t_{Tг}$ - час спрацювання тригера.

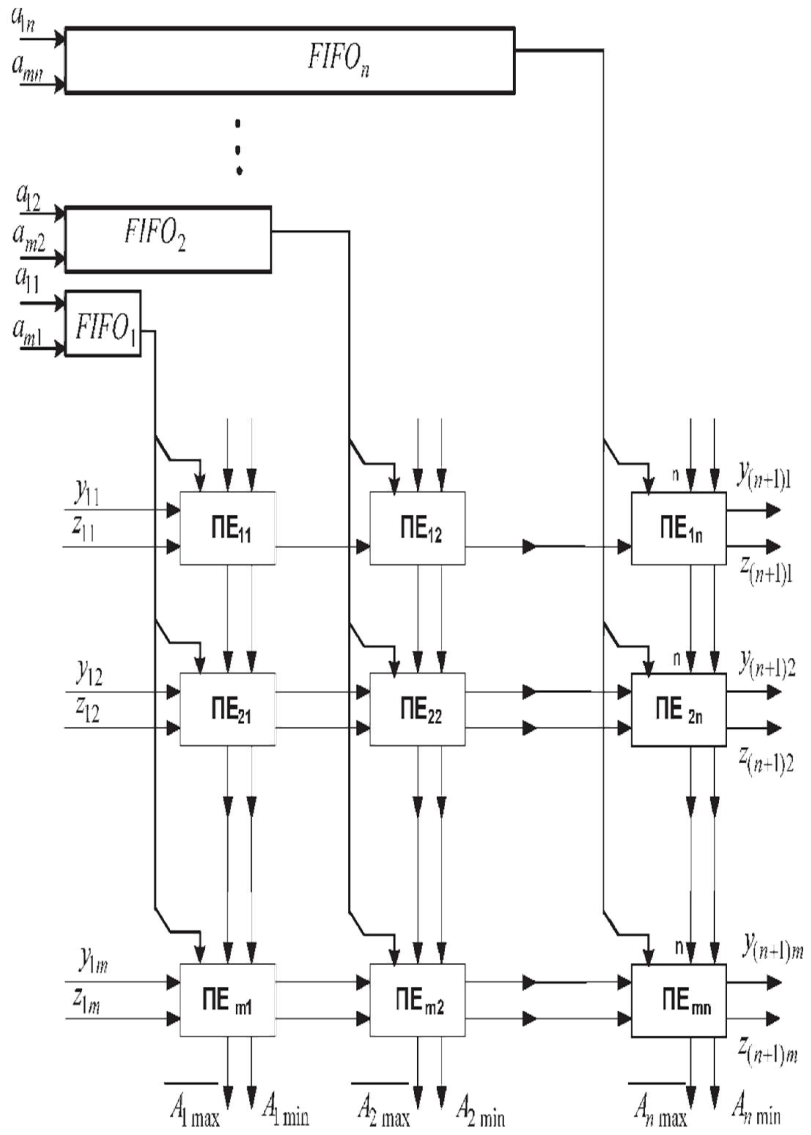


Рисунок 2.3.5. Матрична конвеєрна НВІС-структура пристрою обчислення максимальних і мінімаоьних значень із групи m чисел

Апаратні затрати на реалізацію конвеєрного пристрою для обчислення максимальних і мінімальних значень із групи *m* чисел дорівнюють:

$$W_K = W_{FIFO} + mn(W_{Tг} + 7w_i), \qquad (2.3.17)$$

де W_{FIFO} і $W_{Tг}$ - апаратні затрати відповідно на пам’ять типу FIFO і тригер.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА СИНТЕЗ АПАПРАТНОГО ПРИСТРОЮ СОРТУВАННЯ ДАНИХ МЕТОДОМ ЗЛИТТЯ

3.1 Принципи розпаралелювання обчислень

Для багатьох методів матричних обчислень характерним є повторення одних і тих же обчислювальних дій для різних елементів матриць. Даний момент свідчить про наявність паралелізму за даними при виконанні матричних розрахунків і, як результат, розпаралелювання матричних операцій зводиться в більшості випадків до поділу оброблюваних матриць між процесорами використовуваної обчислювальної системи. Вибір способу поділу матриць призводить до визначення конкретного методу паралельних обчислень; існування різних схем розподілу даних породжує цілий ряд паралельних алгоритмів матричних обчислень.

Найбільш загальні і широко використовувані способи поділу матриць складаються у разі необхідності розділення даних на смуги (по вертикалі або горизонталі) або на прямокутні фрагменти (блоки).

Стрічкові розбиття матриці. При стрічковому (block-striped) розбитті кожному процесору виділяється та чи інша підмножина рядків (rowwise або горизонтальне розбиття) або стовпців (columnwise або вертикальне розбиття) матриці (рисунок. 3.1). Поділ рядків і стовпців на смуги в більшості випадків відбувається на безперервній (послідовної) основі. При такому підході для горизонтального розбиття по рядках, наприклад, матріця А представляється у вигляді (рисунок. 3.1)

$$A = (A_0, A_1, ..., A_{p=1})^T, A_i = (a_{i_0}, a_{i_1}, ..., a_{i_{k-1}}), i_j = ik + j, 0 \le j < k, k = m / p \qquad (3.1.1)$$

де, $A_i = (a_{i_0}, a_{i_1}, ..., a_{i_{k-1}}), i_j = ik + j, 0 \le j < k, k = m / p$ є і-тий рядок матриці А (передбачається, що кількість рядків т кратно числу процесорів р, тобто т.(т = к*р). У всіх алгоритмах матричного множення і множення матриці на вектор, які будуть розглянуті нами в цьому і наступному розділах, використовується поділ даних на безперервній основі.

Інший можливий підхід до формування смуг полягає в застосуванні тієї чи іншої схеми чергування (циклічності) рядків або стовпців. Як правило, для чергування використовується число процесорів p - в цьому випадку при горизонтальному розбитті матриця A набирає вигляду

Інший можливий підхід до формування смуг полягає в застосуванні тієї чи іншої схеми чергування (циклічності) рядків або стовпців. Як правило, для чергування використовується число процесорів p - в цьому випадку при горизонтальному розбитті матриця A набирає вигляду

$$A=(A_0,A_1,...,A_{p=1})^T, A_i=(a_{i_0},a_{i_1},...,a_{i_{k-1}}), i_j=ik+jp, 0\le j<k, k=m/p \qquad (3.1.1)$$

Циклічна схема формування смуг може виявитися корисною для кращої балансування обчислювального навантаження процесорів (наприклад, при вирішенні системи лінійних рівнянь з використанням методу Гауса).

При блоковому (chessboard block) поділі матриця ділиться на прямокутні набори елементів - при цьому, як правило, використовується поділ на безперервній основі. Нехай кількість процесорів p = S • q, кількість рядків матриці є кратним s, а кількість стовпців - кратним q, тобто m = k • s і n = l • q. Уявімо вихідну матрицю A у вигляді набору прямокутних блоків таким чином:

$$A=\begin{pmatrix} A_{00} & A_{02} & ...A_{0q-1} \\ & ... & \\ A_{s-11} & A_{s-12} & ...A_{s-1q-1} \end{pmatrix}, \qquad (3.1.2)$$

где Aij - блок матрицы, состоящий из элементов:

$$A_{ij}=\begin{pmatrix} a_{i_0j_0} & a_{i_0j_s} & ...a_{i_0j_{i-1}} \\ & ... & \\ a_{i_{k-1}j_0} & a_{i_{k-1}j_1} & a_{i_{k-1}j_{i-1}} \end{pmatrix}, i_v=ik+v, 0\le v<k, k=m/s, j_m=jl+u, 0\le u\le l, l=n/q.$$

(3.1.3
)

При такому підході доцільно, щоб обчислювальна система мала фізичну або, принаймні, логічну топологію процессорной решітки з 5 рядків і q стовпців. В цьому випадку при поділі даних на безперервній основі процесори, сусідні в структурі решітки, обробляють суміжні блоки вихідної матриці. Слід зазначити, однак, що і для блокової схеми може бути застосовано циклічне чергування рядків і стовпців.

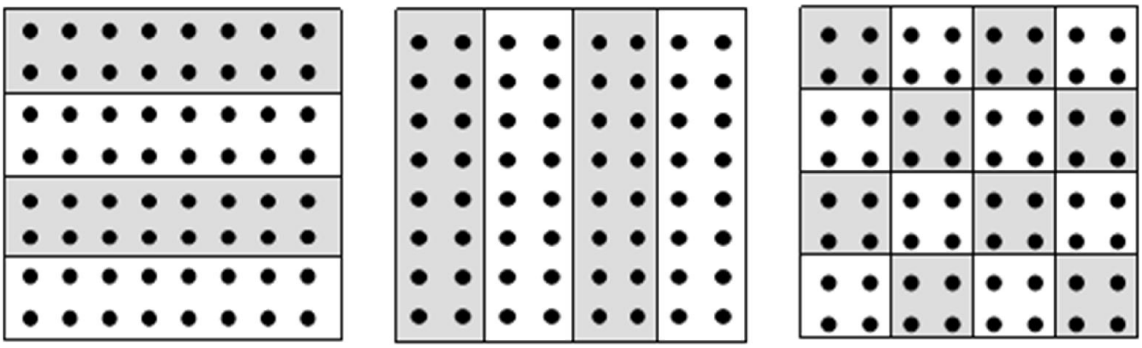


Рисунок 3.1.1 Способи поділу елементів матриці між процесорами обчислювальної системи

Розглядаються три паралельних алгоритму для множення квадратної матриці на вектор. Кожен підхід заснований на різному типі розподілу вихідних даних (елементів матриці і вектора) між процесорами. Поділ даних змінює схему взаємодії процесорів, тому кожен з представлених методів істотно відрізняється від двох інших.

Послідовний алгоритм множення матриці на вектор може бути представлений таким чином.

Матрично-векторне множення - це послідовність обчислення скалярних добутків. Оскільки кожне обчислення скалярного добутку векторів довжини п вимагає виконання п операцій множення і п-1 операцій додавання, його трудомісткість порядку О (п). Для виконання матрично-векторного множення

необхідно виконати т операцій обчислення скалярного твори, таким чином, алгоритм має трудомісткість порядку O (тп).

Розподіл даних

При виконанні паралельних алгоритмів множення матриці на вектор, крім матриці A необхідно розділити ще вектор b і вектор результату c. Елементи векторів можна продублювати, тобто скопіювати всі елементи вектора на всі процесори, складові многопроцессорную обчислювальну систему, або розділити між процесорами. При блоковому розбитті вектора з n елементів кожен процесор обробляє безперервну послідовність з до елементів вектора (ми припускаємо, що розмірність вектора n остачі ділиться на число процесорів, тобто n = кр).

Дублювання векторів b і c між процесорами є допустимим рішенням (далі для простоти викладу будемо вважати, що m = n). Вектора b і з складаються з n елементів, т. Е. Містять стільки ж даних, скільки і один рядок або один стовпець матриці. Якщо процесор зберігає рядок або стовпець матриці і поодинокі елементи векторів b і c, то загальне число зберігаються елементів має порядок O (n). Якщо процесор зберігає рядок (стовпець) матриці і все елементи векторів b і c, то загальне число зберігаються елементів також порядку O (n). Таким чином, при дублюванні і при поділі векторів вимоги до обсягу пам'яті з одного класу складності.

Множення матриці на вектор при поділі даних по рядках

Розглянемо в якості першого прикладу організації паралельних матричних обчислень алгоритм множення матриці на вектор, заснований на представленні матриці безперервними наборами (горизонтальними смугами) рядків. При такому способі поділу даних в якості базової підзадачі може бути обрана операція скалярного множення одного рядка матриці на вектор.

Для виконання базової підзадачі скалярного твори процесор повинен містити відповідний рядок матриці A і копію вектора b. Після завершення обчислень кожна базова подзадача визначає один з елементів вектора результату c.

Для об'єднання результатів розрахунку і отримання повного вектора c на

кожному з процесорів обчислювальної системи необхідно виконати операцію узагальненого збору даних (див. Розділ 6), в якій кожен процесор передає свій обчислений елемент вектора c всім іншим процесорам. Цей крок можна виконати, наприклад, з використанням функції MPI_Allgather з бібліотеки MPI. У загальному вигляді схема інформаційної взаємодії підзадач в ході виконуваних обчислень показана на рис. 7.2.

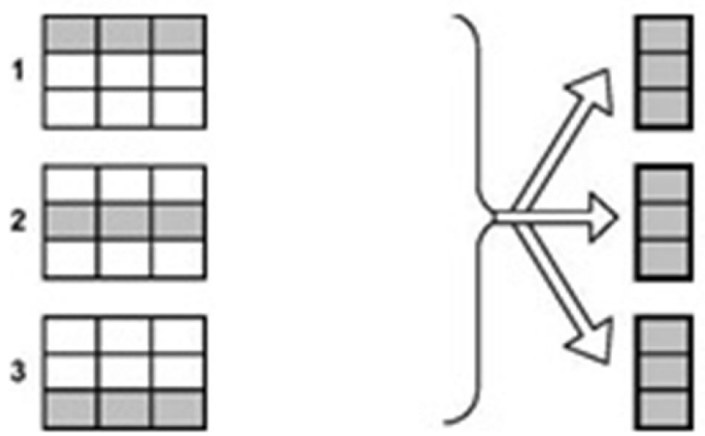


Рисунок 3.1.2 - Організація обчислень при виконанні паралельного алгоритму множення матриці на вектор, заснованого на поділі матриці по рядках

Масштабування і розподіл підзадач по процесорам

В процесі множення щільної матриці на вектор кількість обчислювальних операцій для отримання скалярного твори однаково для всіх базових підзадач. Тому в разі, коли число процесорів p менше числа базових підзадач т, ми можемо об'єднати базові підзадачі таким чином, щоб кожен процесор виконував кілька таких завдань, відповідних безперервній послідовності рядків матриці A. У цьому випадку після закінчення обчислень кожна базова подзадача визначає набір елементів результуючого вектора c. Розподіл підзадач між процесорами обчислювальної системи може бути виконане довільним чином.

Для аналізу ефективності паралельних обчислень тут і далі будуть будуватися два типи оцінок. У першій з них трудомісткість алгоритмів

оцінюється в кількості обчислювальних операцій, необхідних для вирішення поставленого завдання, без урахування витрат часу на передачу даних між процесорами, а тривалість всіх обчислювальних операцій вважається однаковою. Крім того, константи в одержуваних співвідношеннях, як правило, не вказуються - для першого типу оцінок важливий насамперед порядок складності алгоритму, а не влучний вислів часу виконання обчислень. Як результат, в більшості випадків подібні оцінки виходять досить простими і можуть бути використані для початкового аналізу ефективності розроблюваних алгоритмів і методів.

Другий тип оцінок спрямований на формування якомога більше точних співвідношень для передбачення часу виконання алгоритмів. Отримання таких оцінок проводиться, як правило, за допомогою уточнення виразів, отриманих на першому етапі. Для цього в наявні співвідношення вводяться параметри, що задають тривалість виконання операцій, будуються оцінки трудомісткості комунікаційних операцій, вказуються всі необхідні константи. Точність отриманих виразів перевіряється за допомогою проведення обчислювальних експериментів, за результатами яких часи виконаних розрахунків порівнюються з теоретично передбаченими оцінками тривалості обчислень. Як результат, оцінки подібного типу мають, як правило, більш складний вид, але дозволяють більш точно оцінювати ефективність розроблюваних методів паралельних обчислень.

Трудомісткість алгоритму множення матриці на вектор. У разі, якщо матриця А квадратна (т = п), послідовний алгоритм множення матриці на вектор має складність Т1 = п2. У разі паралельних обчислень кожен процесор виробляє множення тільки частини (смуги) матриці А на вектор Ь, розмір цих смуг дорівнює п / р рядків. При обчисленні скалярного твори одного рядка матриці і вектора необхідно провести п операцій множення і (п-1) операцій додавання. Отже, обчислювальна трудомісткість паралельного алгоритму визначається виразом:

$$Тр=п^2/р.$$

$$S_p = \frac{n^2}{n^2/p} = p,\ E_p = \frac{n^2}{p \cdot (n^2/p)} = 1.$$

(3.2.1)

З урахуванням цієї оцінки показники прискорення і ефективності паралельного алгоритму мають вигляд:

Побудовані вище оцінки часу обчислень виражені в кількості операцій і, крім того, визначено без урахування витрат на виконання операцій передачі даних. Використовуємо раніше висловлені припущення про те, що виконуються операції множення і складання мають однакову тривалість т. Крім того, будемо припускати також, що обчислювальна система є однорідною, тобто всі процесори, що складають цю систему, мають однакову продуктивністю. З урахуванням введених припущень час виконання паралельного алгоритму, пов'язане безпосередньо з обчисленнями, становить

$$T_p(calc) = \lceil n/p \rceil \cdot (2n-1) \cdot \tau \qquad (3.2.2)$$

Оцінка трудомісткості операції узагальненого збору даних вже виконувалася в розділі 6 (див. П.6.3.4). Як уже зазначалося раніше, дана операція може бути виконана за \ log2 p \ ітерацій? На першій ітерації взаємодіючі пари процесорів обмінюються повідомленнями об'ємом w \ njp \ (w є розмір одного елемента вектора с в байтах), на другій ітерації цей обсяг збільшується вдвічі і виявляється рівним 2w \ n / p \ і т.д. Як результат, тривалість виконання операції збору даних при використанні моделі Хокні може бути визначена за допомогою наступного виразу \logiP~\

$$T_p = (comm) = \sum_{i=1}^{\lceil \log_2 p \rceil} (\alpha + 2^{i-1} w \lceil n/p \rceil / \beta) = \alpha \lceil \log_2 p \rceil + w \lceil n/p \rceil (2^{\lceil \log_2 p \rceil} - 1)/\beta, \qquad (3.2.3)$$

де а - латентність мережі передачі даних, в - пропускна здатність мережі.

Таким чином, загальний час виконання паралельного алгоритму становить.

$$T_p = (n/p)\cdot(2n-1)\cdot\tau + \alpha\cdot\log_2 p + w(n/p)(p-1)/\beta \tag{3.2.4}$$

3.2 Розробка пристрою знаходження мінімального та максимального значення за допомогою GPU

Реалізація обчислення максимальних і мінімальних значень на GPU Nvidia CUDA, на графічному адаптері GEFORCE GT540m. CUDA- програмно-апаратна архітектура паралельних обчислень, яка дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів фірми Nvidia. CUDA SDK дозволяє програмістам реалізовувати на спеціальному спрощеному діалекті мови програмування Сі алгоритми, здійснимі на графічних процесорах Nvidia, і включати спеціальні функції в текст програми на Сі. Архітектура CUDA дає розробнику можливість на свій розсуд організовувати доступ до набору інструкцій графічного прискорювача і управляти його пам'яттю.

Первісна версія CUDA SDK була представлена 15 лютого 2007 року. В основі інтерфейсу програмування додатків CUDA лежить мова Сі з деякими розширеннями. Для успішної трансляції коду на цій мові до складу CUDA SDK входить власний Сі-компілятор командного рядка nvcc компанії Nvidia. Компілятор nvcc створений на основі відкритого компілятора Open64 і призначений для трансляції host-коду (головного, керуючого коду) і device-коду (апаратного коду) (файлів з розширенням .cu) в об'єктні файли, придатні в процесі збірки кінцевої програми або бібліотеки в будь-який середовищі програмування, наприклад, в NetBeans.

В архітектурі CUDA використовується модель пам'яті грід, кластерне моделювання потоків і SIMD-інструкції. Чи можна застосувати не тільки для високопродуктивних графічних обчислень, але і для різних наукових обчислень

з використанням відеокарт nVidia. Вчені і дослідники широко використовують CUDA в різних областях, включаючи астрофізику, обчислювальну біологію та хімію, моделювання динаміки рідин, електромагнітних взаємодій, комп'ютерну томографію, сейсмічний аналіз і багато іншого. У CUDA є можливість підключення до додатків, що використовують OpenGL і Direct3D. CUDA - багатоплатформність для таких операційних систем як Linux, Mac OS X і Windows.22 березня 2010 року nVidia випустила CUDA Toolkit 3.0, який містив підтримку OpenCL.

Платформа CUDA вперше з'явилися на ринку з виходом чіпа NVIDIA восьмого покоління G80 і стала бути присутнім у всіх наступних серіях графічних чіпів, які використовуються в родинах прискорювачів GeForce, Quadro і NVidia Tesla.

Перша серія обладнання, що підтримує CUDA SDK, G8x, мала 32-бітний векторний процесор одинарної точності, який використовує CUDA SDK як API (CUDA підтримує тип double мови Сі, однак зараз його точність знижена до 32-бітного з плаваючою комою). Пізніші процесори GT200 мають підтримку 64-бітної точності (тільки для SFU), але продуктивність значно гірше, ніж для 32-бітної точності (через те, що SFU всього два на кожен потоковий мультипроцесор, а скалярних процесорів - вісім). Графічний процесор організовує апаратну багатопоточність, що дозволяє задіяти всі ресурси графічного процесора. Таким чином, відкривається перспектива перекласти функції фізичного прискорювача на графічний прискорювач (приклад реалізації - PhysX). Також відкриваються широкі можливості використання графічного обладнання комп'ютера для виконання складних неграфічних обчислень: наприклад, в обчислювальної біології та в інших галузях науки.

У порівнянні з традиційним підходом до організації обчислень загального призначення за допомогою можливостей графічних API, у архітектури CUDA відзначають наступні переваги в цій області:

- Інтерфейс програмування додатків CUDA (CUDA API) заснований на стандартній мові програмування Сі з деякими обмеженнями. На думку

розробників, це повинно спростити і згладити процес вивчення архітектури CUDA

- Колективна між потоками пам'ять (shared memory) розміром в 16 Кб може бути використана під організований користувачем кеш з більш широкою смугою пропускання, ніж при вибірці з звичайних текстур
- Більш ефективні транзакції між пам'яттю центрального процесора і відеопам'яттю
- Повна апаратна підтримка цілочисельних і побітових операцій
- Підтримка компіляції GPU коду засобами відкритого LLVM

GEFORCE GT540m- Досить швидка відеокарта середнього класу, представлена виробником на початку січня 2011 року. На період виходу цей графічний процесор є досить міцним «середнячком» в своєму класі і цінової категорії. Новинка підтримує технологію DirectX 11, а також CUDA, PhysX, Optimus.

Таблиця 3.2.1 Технічна характеристика GEFORCE GT540m

Серія	GEFORCE GT540m
Архітектура	Fermi
Ядер CUDA	96
Тактова частота	672 МГц
Частота пам’яті	900 МГц
Тип пам’яті	DDR3
Розрядність шини пам’яті	128 біт
Максимум відеопам’яті	1536 Мб

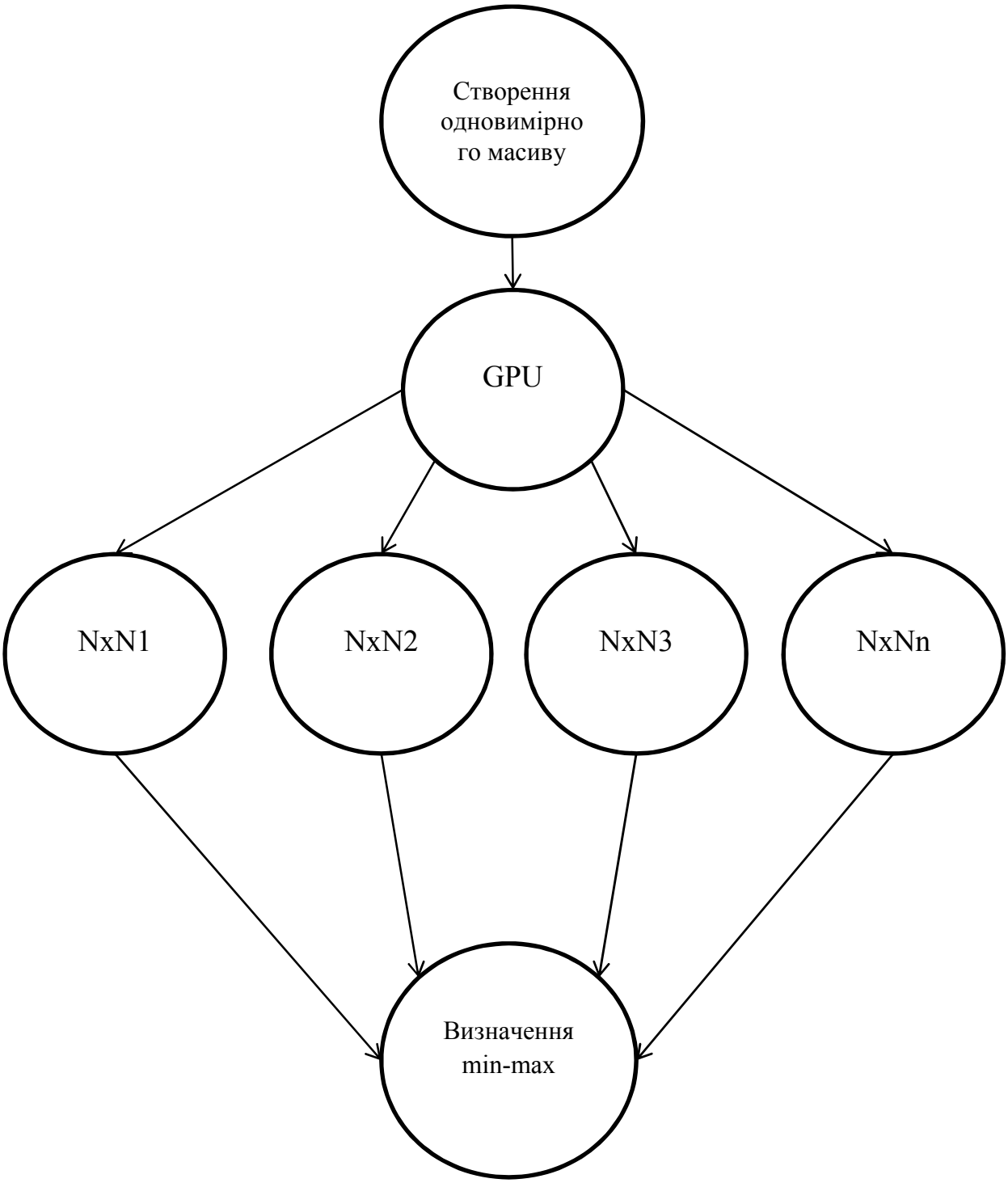


Рисунок 3.2.1. Ярусно-параллельна форма пошуку мінімального і максимального значення на GPU

Пошук мінімального і максимального значення на GPUNvidiaCUDA

Графічний адаптер GEFORCE GT 540m

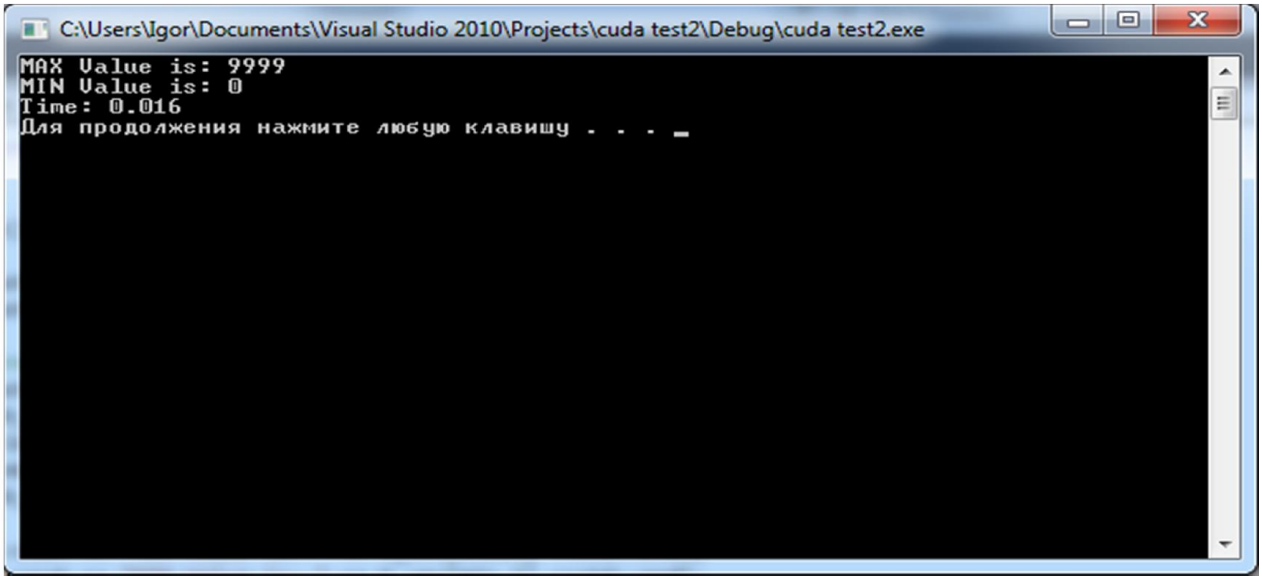
```
#include<thrust/device_vector.h>
#include<thrust/tuple.h>
```

```
#include<thrust/reduce.h>
#include<thrust/fill.h>
#include<thrust/generate.h>
#include<thrust/sort.h>
#include<thrust/sequence.h>
#include<thrust/copy.h>
#include<cstdlib>
#include<ctime>
#include<time.h>

usingnamespace thrust;

// return the biggest of two tuples
template<class T>
struct bigger_tuple {
   __device__ __host__
tuple<T,int>operator()(const tuple<T,int>&a, const tuple<T,int>&b)
   {
if (a > b) return a;
elsereturn b;
   }
};
template<class T>
int max_index(device_vector<T>& vec) {

// create implicit index sequence [0, 1, 2, ... )
   counting_iterator<int>begin(0); counting_iterator<int> end(vec.size());
tuple<T,int> init(vec[0],0);
tuple<T,int> smallest;
```

```
smallest = reduce(make_zip_iterator(make_tuple(vec.begin(), begin)),
make_zip_iterator(make_tuple(vec.end(), end)),
init, bigger_tuple<T>());
return get<1>(smallest);
}
// return the biggest of two tuples
template<class T>
struct smaller_tuple {
   __device__ __host__
tuple<T,int>operator()(const tuple<T,int>&a, const tuple<T,int>&b)
   {
if (a < b) return a;
elsereturn b;
   }

};

template<class T>
int min_index(device_vector<T>& vec) {

// create implicit index sequence [0, 1, 2, ... )
   counting_iterator<int>begin(0); counting_iterator<int> end(vec.size());
tuple<T,int> init(vec[0],0);
tuple<T,int> smallest;

smallest = reduce(make_zip_iterator(make_tuple(vec.begin(), begin)),
make_zip_iterator(make_tuple(vec.end(), end)),
init, smaller_tuple<T>());
return get<1>(smallest);
}
```

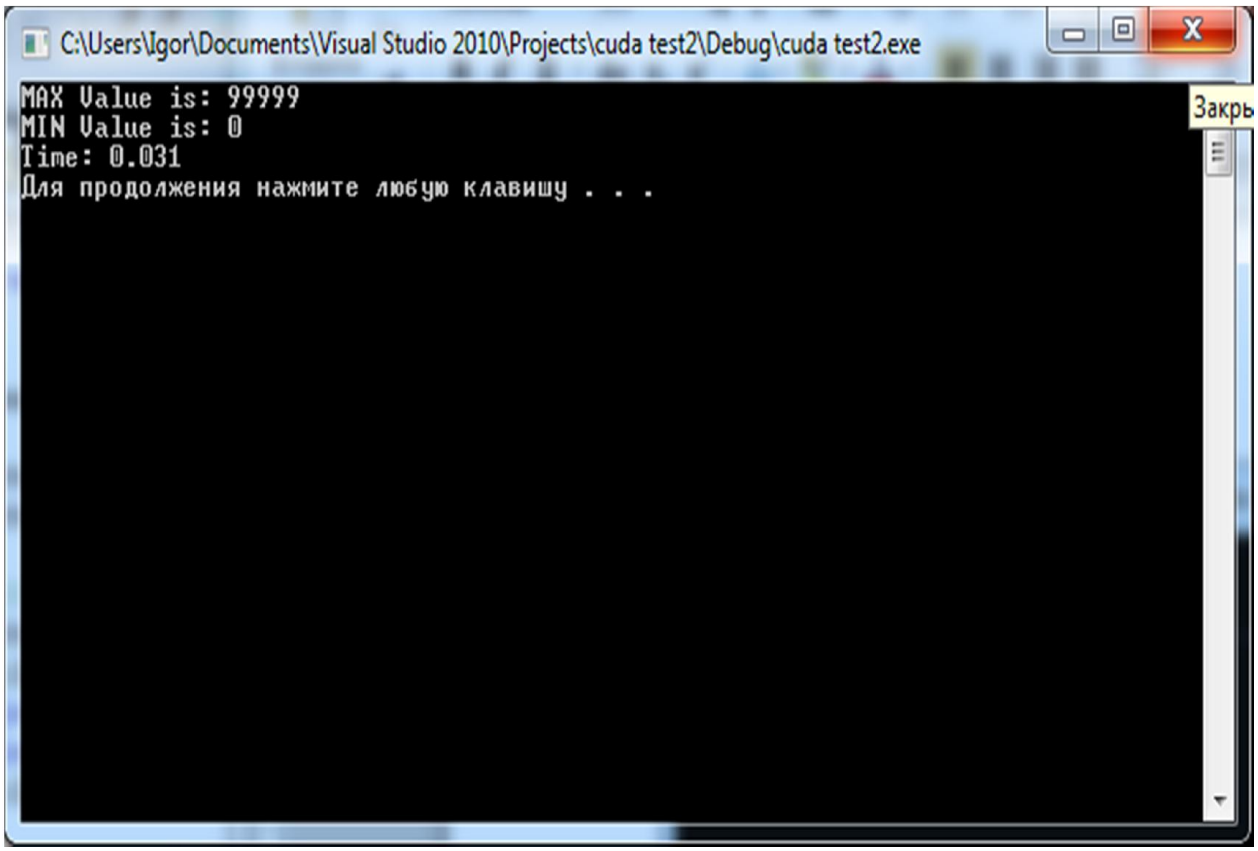
```
int main(){

    thrust::host_vector<int> h_vec(1000000);
    thrust::sequence(h_vec.begin(), h_vec.end()); // values = indices

// transfer data to the device
    thrust::device_vector<int> d_vec = h_vec;
        clock_t begin = clock();
int index = max_index(d_vec);
        int min_i = min_index(d_vec);
        clock_t end = clock();

    std::cout <<"MAX Value is: "<< h_vec[index] <<std::endl;
    std::cout <<"MIN Value is: "<< h_vec[min_i] <<std::endl;
        std::cout <<"Time: "<<double(end - begin) / CLOCKS_PER_SEC <<
std::endl;
        std::system("Pause");
return 0;
}
```

Експеримент 1, матриця з 1000 елементів



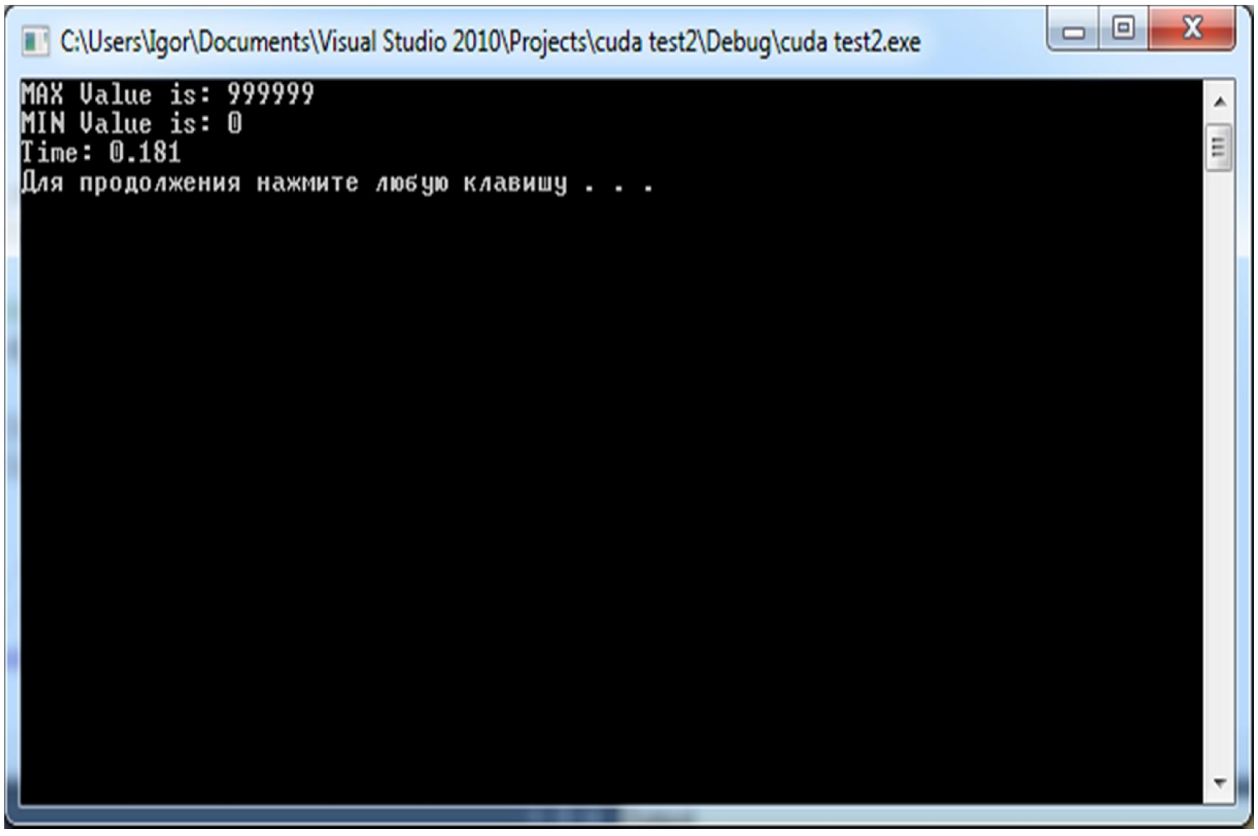
Зображення 3.2.1. Результат експеримента 1

Експеримент 2, матриця з 10000 елементів



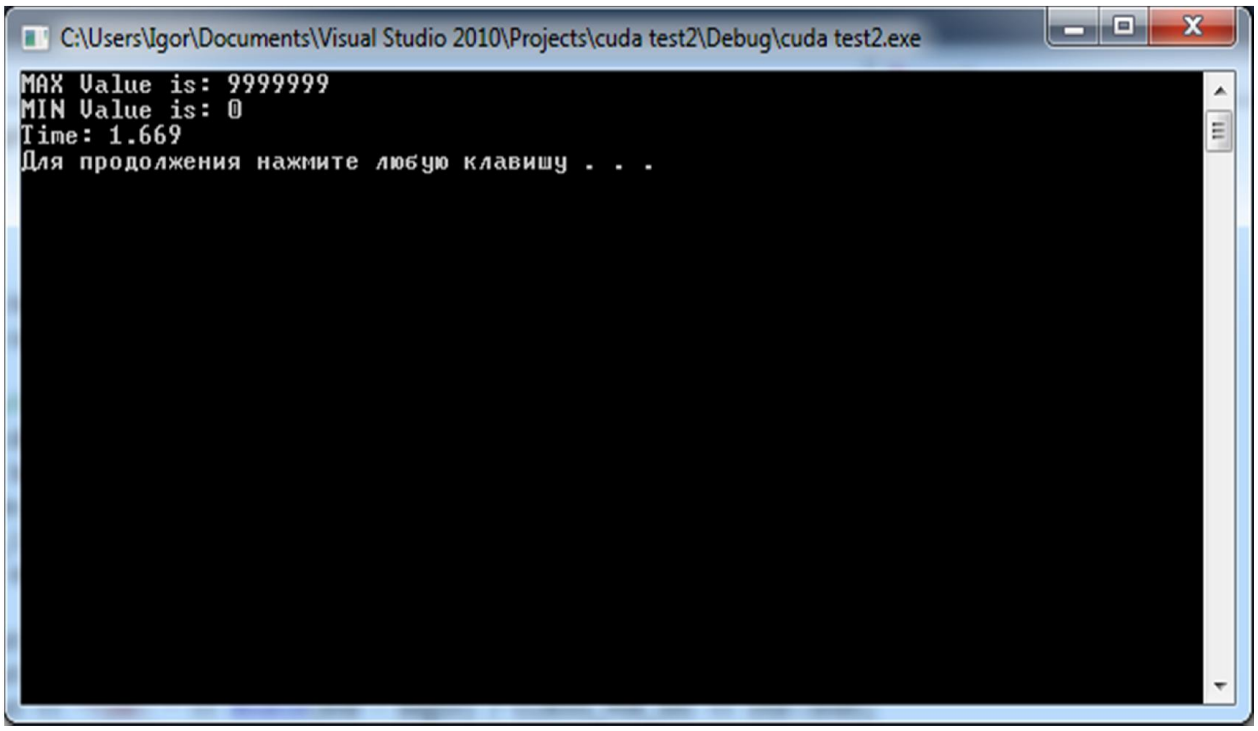
Зображення 3.2.2. Результат експеримента 2

Експеримент 3, матриця з 100000 елементів



Зображення 3.2.3. Результат експеримента 3

Експеримент 4, матриця з 10000000 елементів



Зображення 3.2.4. Результат експеримента 4

Таблиця 3.2.2 Результати експериментів обчислення максимальних і мінімальних значень чисел на GPU

Експеримент	Кількість елементів в матриці	Час виконання(мс)
1	1000	0,016
2	10000	0,031
3	100000	0,181
4	10000000	1,669

3.3 Розробка пристрою знаходження мінімального та максимального значення за допомогою CPU

Реалізація обчислення максимальних і мінімальних значень на CPU Intel core I3- 2330m. Intel Core i3-2330M - двоядерний процесор, призначений для ноутбуків. Він заснований на архітектурі Sandy Bridge і підтримує технологію Hyperthreading, для одночасної обробки 4 потоків даних (для кращого

використання конвеєра). У порівнянні з більш швидким Core i5, i3 не підтримує функцію підвищення тактової частоти Turbo Boost, тому працює на постійній частоті 2.2 ГГц. У порівнянні з Core i3-2310M, 2330M працює на частоті лише на 100 МГц перевищує основний показник.

Архітектура Sandy Bridge прийшла на зміну Arrandale, і підтримує нові інструкції AVX 256-біт, поліпшену технологію Turbo 2.0 і оснащується вбудованою графічною картою, виконаної по 32 нм технологічному процесу.

Core i3-2330M оснащується вбудованою графічною картою Intel HD Graphics 3000, яка є помітно більш швидкої в порівнянні з Intel HD Graphics в процесорах сімейства Arrandale. Новий графічний чіп інтегрований в CPU і проводиться по 32 нм техпроцесу, може використовувати кеш-пам'ять третього рівня спільно з самим процесором (завдяки використанню нової кільцевої шини). У 2330M відеокарта працює на частоті від 650 до 1100 МГц (при використанні Turbo Boost). У більш швидких процесорах лінійки Sandy Bridge ця карта може підвищити частоту до 1300 МГц (наприклад, i5-2520M).

Крім того, покращений двоканальний контролер пам'яті для стандарту DDR3 розташовується на кристалі CPU, що наближає нас до реалізації повнофункціональної «системи в чіпі».

Завдяки покращеній архітектурі, середня продуктивність Core i3-2330M вище, ніж у Core i3 сімейства Arrandale з аналогічною частотою без Turbo Boost. У синтетичних бенчмарках, продуктивність даного процесора буде приблизно такою ж, як у Core i3-390M (з більш високою тактовою частотою - 2.5 ГГц). Отже, 2330M повинен без проблем впоратися з більшістю додатків. Показник 35 Вт TDP включає енергоспоживання самого процесора, вбудованого GPU і контролера пам'яті.

Таблиця 3.3.1. Технічна характеристика Intel Core i3-2330M

Серія	Intel Core i3-2330M
Архітектура	Sandy Bridge
Тактова частота	2200 МГц
Кеш 1-го рівня	128 Кб

Продовження таблиця 3.3.1

Кеш 2-го рівня	512 Кб
Кеш 3-го рівня	3072 Кб
Кількість ядер	2
Кількість потоків	4
Максимальне енергоспоживання	35 Вт
Число транзисторів	624 Млн
Техпроцес	32 нм
Максимальна температура	85(PGA); 100(BGA) °C
Сокет	rPGA988B / BGA 1023
Додатково	HD Graphics 3000 (650- 1100MHz), DDR3- 1066/1333 Memory Controller (max 8GB), HyperTheading, AVX, Quick Synk, Virtualization
64 Bit	підтримка 64 Bit
Апаратна віртуалізація	VT-x
Дата виходу	15.05.2011

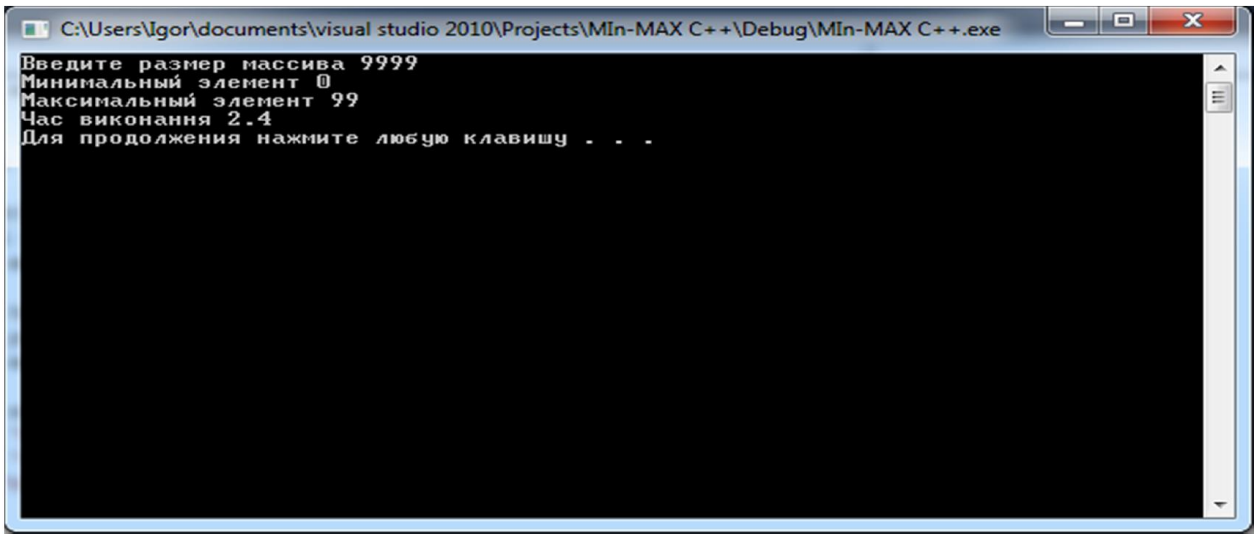
Опис бібліотеки iostream - заголовки з класами, функціями і змінними для організації введення-виведення в мові програмування C ++. Він включений в стандартну бібліотеку C ++. Назва утворена від Input / Output Stream («потік введення-виведення»). У мові C ++ і його попередника, мовою програмування Ci, немає вбудованої підтримки введення-виведення, замість цього використовується бібліотека функцій. iostream управляє введенням-висновком, як і stdio.h в Ci. iostream використовує об'єкти cin, cout, cerr і clog для передачі інформації в і з стандартних потоків введення, виведення, помилок (без буферизації) і помилок (з буферизацією) відповідно. Будучи частиною стандартної бібліотеки C ++, ці об'єкти також є частиною стандартного простору імен - std.

Пошук мінімального і максимального значення на CPUintelcoreI3 -2330m

```
#include<iostream>
#include<ctime>

usingnamespace std;
int main()
{
setlocale(0,"Rus");
srand((unsigned) time(NULL));
int* ptr;
int n;
unsignedlongint i;
int min, max;
int temp;
        time_t  start, stop;

cout<<"Введитеразмермассива ";
clock_t begin = clock();
        cin >> n;
   ptr = newint[n];  // динамически выделяем память
for ( i = 0; i < n; i++)
ptr[i] = rand() % 100; // заполняемслучайнымичислами

/*for ( i = 0; i < n; i++)
cout<< ptr[i] << " ";
cout<< endl; */

min = ptr[0];

for ( i = 1; i < n; i++)
if(min > ptr[i])
      {
```

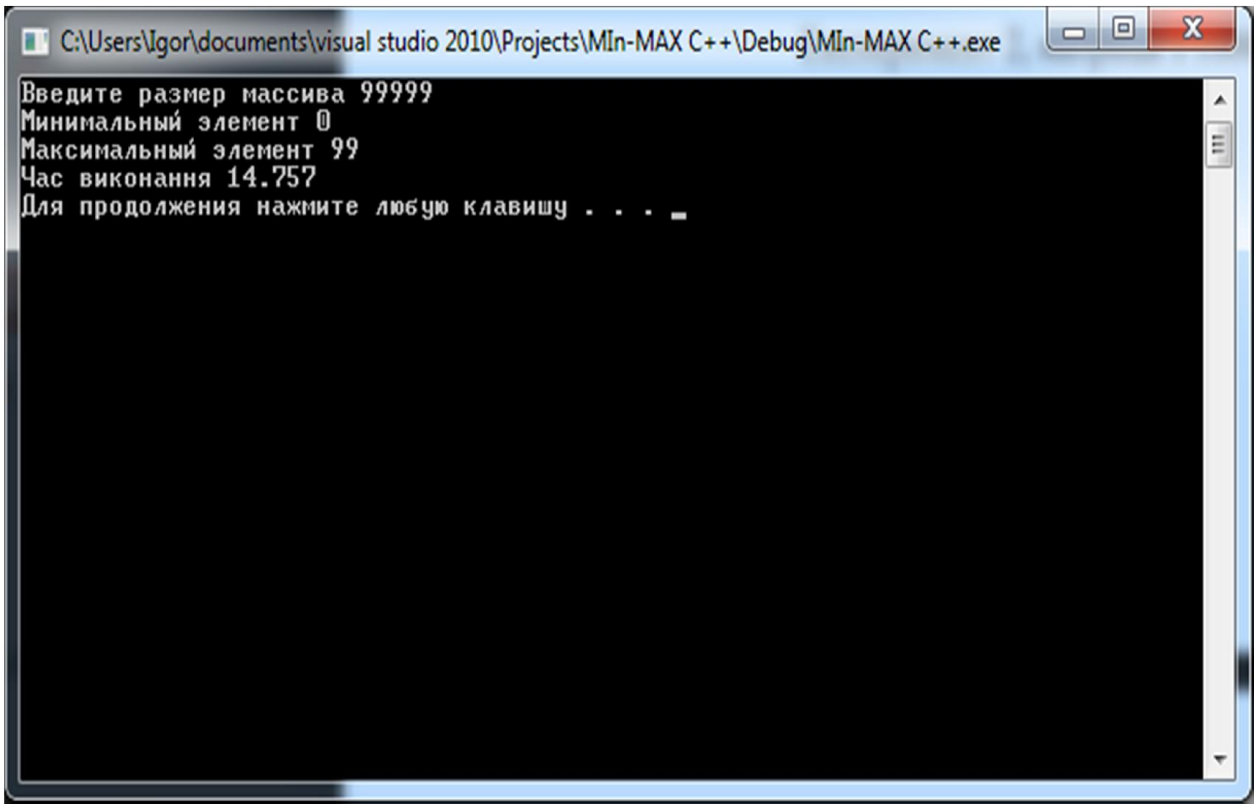
```
temp = ptr[i];
ptr[i] = min;
min = temp;
}

      cout <<"Минимальный элемент "<< min << endl;

        max = ptr[0];
        for (  i = 1; i < n; i++)
                if( max < ptr[i])
                {
                        temp = ptr[i];
                        ptr[i] = max;
                        max = temp;
                }
        clock_t end = clock();
        cout<<"Максимальныйэлемент "<< max << endl;
cout<<"Часвиконання "<<double(end - begin) / CLOCKS_PER_SEC << endl;
        system("Pause");
}
```

Експеримент 1, матриця з 1000 елементів



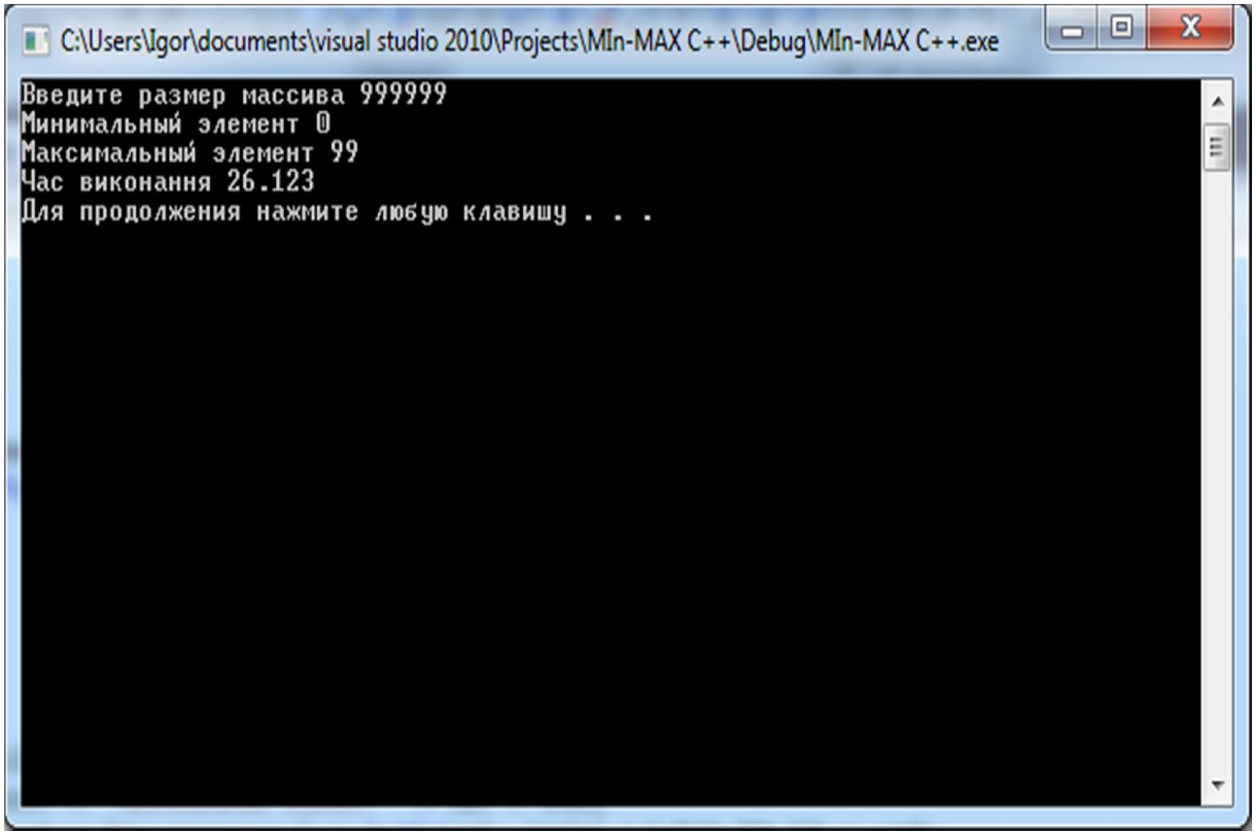
Зображення 3.3.1. Результат експеримента 1

Експеримент 2, матриця з 10000 елементів



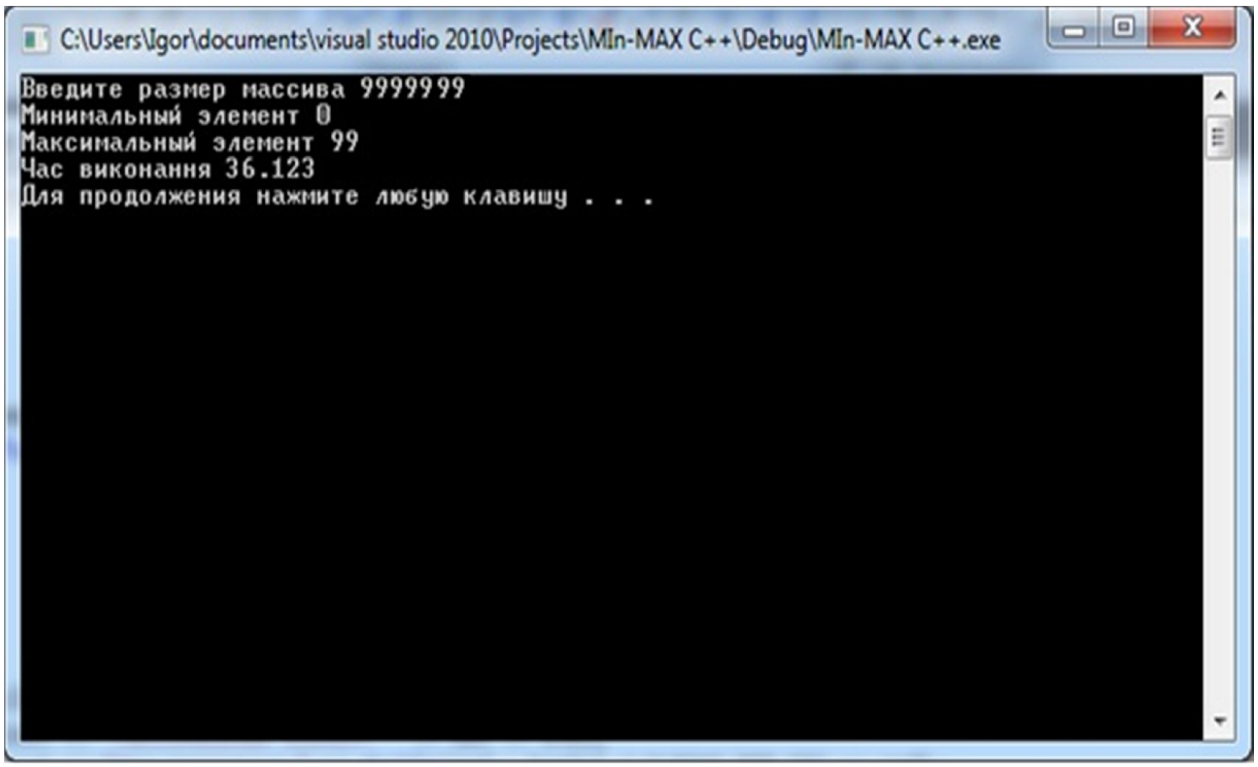
Зображення 3.3.2. Результат експеримента 2

Експеримент 3, матриця з 100000 елементів



Зображення 3.3.3. Результат експеримента 3

Експеримент 4, матриця з 10000000 елементів



Зображення 3.3.4. Результат експеримента 4

Таблиця 3.3.2 Результати експериментів обчислення максимальних і мінімальних значень чисел на CPU

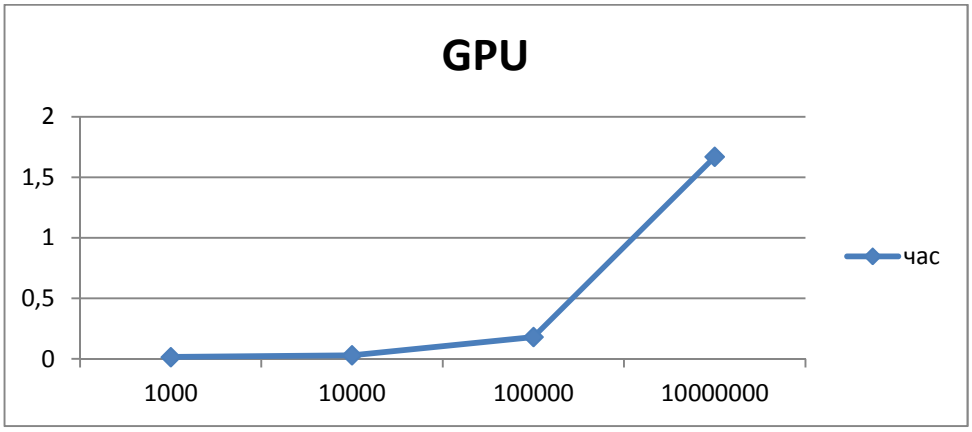
Експеримент	Кількість елементів в матриці	Час виконання(мс)
1	1000	2,4
2	10000	14,757
3	100000	26,123
4	10000000	36,123

3.4 Порівняння результатів виконання обчислення максимальних і мінімальних значень чисел на GPU і CPU

Для візуального представлення і порівняння результатів експериментів було прийнято рішення про побудову графіків. В графіках взято такі значення як кількість елементів в матриці і час виконання обчислення.

Таблиця 3.4.1 Результати експериментів обчислення максимальних і мінімальних значень чисел на GPU

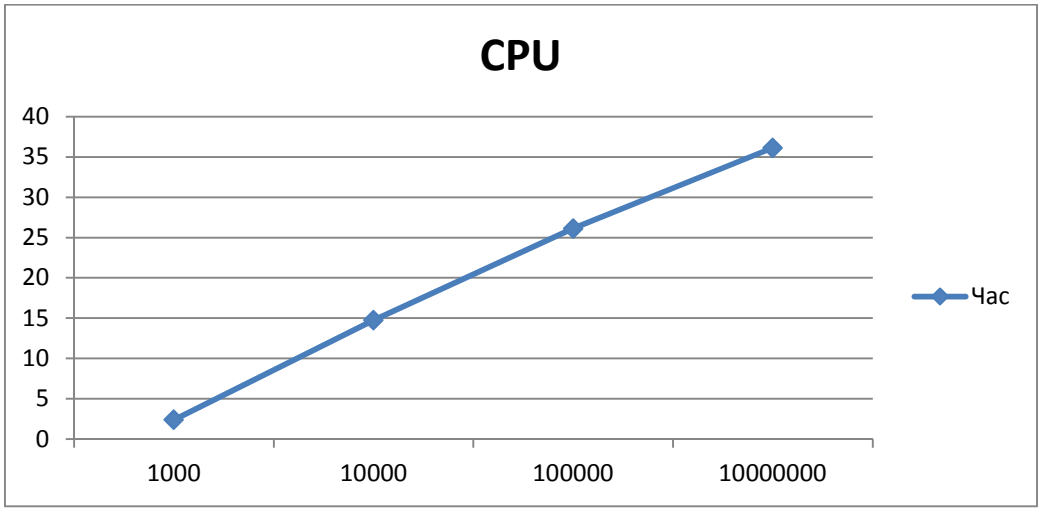
Експеримент	Кількість елементів в матриці	Час виконання(мс)
1	1000	0,016
2	10000	0,031
3	100000	0,181
4	10000000	1,669



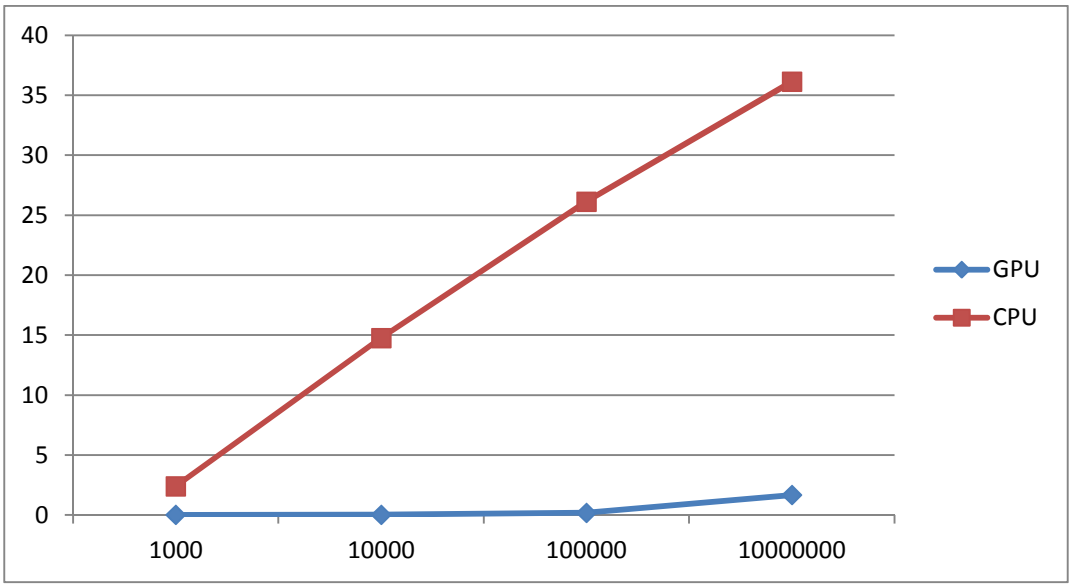
Графік 3.4.1 Обчислення максимальних і мінімальних значень чисел на GPU

Таблиця 3.4.2 Результати експериментів обчислення максимальних і мінімальних значень чисел на CPU

Експеримент	Кількість елементів в матриці	Час виконання(мс)
1	1000	2,4
2	10000	14,757
3	100000	26,123
4	10000000	36,123



Графік 3.4.2 Обчислення максимальних і мінімальних значень чисел на CPU



Графік 3.4.3 Порівняння обчислення максимальних і мінімальних значень чисел на GPU і CPU

ВИСНОВКИ

1. Обчислення максимальних і мінімальних значень чисел в масиві даних з використанням графічного процесора GPU та програмної моделі CUDA можна забезпечити при комплексному підході, який охоплює: дослідження методів і алгоритмів . обчислення максимальних і мінімальних значень чисел в масиві даних; архітектуру графічного процесора GPU та програмну модель CUDA.

2. Обчислення максимальних і мінімальних значень чисел в масиві даних ґрунтується на методі порозрядного порівняння, за таким методом виконується послідовне порівняння розрядів всіх чисел починаючи зі старшого..

3.Для розробки програмних засобів . обчислення максимальних і мінімальних значень чисел в масиві даних з використанням графічного процесора GPU та програмної моделі CUDA алгоритм обчислення максимальних і мінімальних значень чисел необхідно подати у вигляду ярусно-паралельній формі.

4.Розроблення високоефективних паралельних структур для обчислення максимальних і мінімальних значень чисел в масиві даних методом порозрядного порівняння найдоцільніше здійснювати при інтегрованому підході, який охоплює методи, алгоритми, структури і НВІС-технологію та враховує особливості конкретного застосування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. Хайкин. Нейронные сети: полный курс, 2 –е изд.:Пер. с англ.. – М.:”Вильямс”, 2006. – 1104 с.

2. Галушкин А.И. Нейрокомпьютеры. Кн.3.-.М; ИПРЖР,2000.-528с.

3. Нейроподібні методи, алгоритми та структури обробки сигналів і зображень у реальному часі: монографія / Ю.М. Рашкевич, Р.О. Ткаченко, І.Г Цмоць, Д.Д. Пелешко. – Львів: Видавництво Львівської політехніки, 2014. -256 с.

4. Проблемно–ориентированные высокопроизводительные вычислительные системы: В.Ф. Гузик, В.Е. Золотовский: Учебное пособие. Таганрог:Изд-во ТРТУ, 1998. 236 с.

5. Уоссермен Ф.Нейрокомпьютерная техника. – М.: Мир,1992. – 259с.

6. А.В. Палагин, В.Н. Опанасенко. Реконфигуриремые вычислительные системы. – К.: Просвіта, 2006.- 280с.

7. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. – М.: Горячая Линия-Телеком, 2002. – 382 с.

8. Николаев А.Б., Фоминых И.Б. Нейросетевые методы анализа и обработки данных. Учебное пособие. - М.: МАДИ (ГТУ), 2003, - 95с.

9. Цмоць І.Г. Інформаційні технології та спеціалізовані засоби обробки сигналів і зображень у реальному часі. – Львів: УАД, 2005.- 227с.

10. Грибачев В. П. Элементная база аппаратных реализаций нейронных сетей // Компоненты и технологии. 2006. № 8

11. Круг П.Г. Нейронные сети и нейрокомпьютеры: Учебное пособие по курсу «Микропроцессоры». – М.: Издательство МЭИ, 2002. – 176 с.

12. Проблемы построения и обучения нейронных сетей / под ред. А.И.Галушкина и В.А.Шахнова. - М. Изд-во Машиностроение. Библиотечка журнала Информационные технологии №1. 1999. 105 с.

13. А.И.Галушкин Некоторые исторические аспекты развития элементной базы вычислительных систем с массовым параллелизмом (80- и 90-годы) // Нейрокомпьютер, №1. 2000. - С.68-82

14. С.И.Аряшев, С.Г.Бобков, Е.А.Сидоров Параллельный перепрограммируемый вычислитель для систем обработки информационных сигналов // "Нейроинформатика -99". - Москав, МИФИ. Часть 2. С.25-33.

15. Э.Ю. Кирсанов Цифровые нейрокомпьютеры: Архитектура и схемотехника / Под ред. А.И.Галушкина. - Казань: Казанский Гос. У-т. 1995. 131 с.

16. А.И. Власов. Аппаратная реализация нейровычислительных управляющих систем //Приборы и системы управления - 1999, №2, С.61-65.

17. Борисов В.Л., Капитанов В.Д. Методика быстрого создания нейроускорителей // Нейрокопьютеры: разработка и приенение, №1, 2000 год.- С.12-24.

18. Роберт Хехт-Нильсен Нейрокомпьютинг: история, состояние, перспективы // Открытые системы. N4. 1998.

19. А.И. Власов Нейросетевая реализация микропроцессорных систем активной акусто- и виброзащиты// Нейрокомпьютеры:разработка и применение, №1, 2000. С.40-44.

20. Нейроприскорювачі на базі нейрочіпів - http://citforum.ru/hardware/neurocomp/neyrocomp_07.shtml

21. Елементна база нейрообчислювачів - http://opticstoday.com/katalog-statej/stati-na-ukrainskom/nejrokomputeri/elementna-baza-nejroobchislyuvachiv.html

22. Сучасні напрямки розвитку нейрокомп’ютерних технологій - http://www.victoria.lviv.ua/html/oio/html/theme9.htm

23. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы – М.: Изд. Дом «Вильямс», 2001. – 384с.

24. Браунси К. Основные концепции структур данных и реализация в C++. – М.: Изд. Дом «Вильямс», 2002. – 320с.

25. Проценко В.С. Техніка програмування мовою Сі: Навчальний посібник – К.: Либідь, 1993. – 224 с.

26. Шилдт Г. Теория и практика C++ – СПб.: BHV – Санкт-Питербург, 1996. – 416 с.

27. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: «Мир», 1979. - 536с.

28. Вирт Н. Алгоритмы + структуры данных = программы. - М.: "Мир", 1985. - 544 с.

29. Гудман С. Хидетниеми С. Введение в разработку и анализ алгоритмов. - М.: "Мир", 1981. - 366 с.

30. Кнут Д. Искусство программирования для ЭВМ. т.3. Сортировка и поиск. М.:Мир, 1976. - 678 с.

31. Мейер Б., Бодуэн К. Методы программирования: В 2-х томах – М.: Мир, 1982. – 356+368с.

32. Керниган, Б.,Мова програмування Сі. Завдання по мові Сі/Б.Керниган, Д.Ритчи. —М.:ФиС, 1985. — 280 с.

33. Страустрап, Б. Мова програмування Сі ++ /Б.Страуструп. — М.:Радіо та зв’язок, 1991.—352 с.

34. Белецкий, Я. Енциклопедія мови Сі /Я.Белецкий. — М.:Мир, 1992. — 687 с.

35. Белецкий, Я.ТурбоСі++: Нова розробка: навч. посібник для студентів вищих навчальних закладів / Я.Белецкий. — М.'.Машинобудівництво, 1994. — 400с.

36. Пильщиков, В. Н. Збірник вправ по мові Паскаль: навч. Посібник для втузов /В. Н.Пильщиков. — М.:Висш. шк.,1990. —223 с.

37. *Кармен, Томас Х., Лейзерон, Чарльз И., Ривест, Рональд Л., Штайн, Клиффорд.* Алгоритми: построение и анализ, 2-е издание. :Пер. с англ. - М.: Издательский дом “Вильямс”, 2005. - 1296 с.

38. *Кнут Д.* Искусство программирования для ЗВМ: Сортировка и поиск. М., - 1978. - 844с.

39. *Кухарев Г.А.* и др. Техника параллельной обработки бинарних данньїх на СБИС. - М.: Виш. Шк., 1991. - 226 с

40. Пат. № 66138, Україна, МПК 006Б 7/38. Пристрій для обчислення сум парних добутків: Патент на корисну модель / *І.Г. Цмоць, О.В. Скорохода*;

заявник і патентовласник Національний університет «Львівська політехніка». - № и201106811; заявл. 30.05.2011; опубл. 26.12.2011, Бюл. № 24. - 8 с..

41. Патент України на винахід №29700. Пристрій для визначення максимального числа з групи чисел. Бюл. №6-11. - 2000. *Рашкевич Ю.М., Зербіно Д.Д, Цмоць І.Г.*

42. *Кун С.* Матричние процессорина СБИС. - М.: Мир, 1991. – 672

43. Галушкин А.И. Нейрокомпьютеры. Кн.3.-.М; ИПРЖР,2000.-528с.

44. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. – 2-е изд., стереотип. – М.: Горячая линия-Телеком, 2002. – 382 с.

45. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика – М.: Горячая Линия-Телеком, 2002 – 382 с.

46. Рутковская Д., Пилиньский Л., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы / Пер. с польского – М.: Горячая линия-Телеком, 2007. – 452 с.

47. с. Рассел С., Норвиг П. Искусственный интеллект: современный подход / Пер. с английского – М.: Вильямс, 2007. – 1408 с.

48. Рассел С., Норвиг П. Искусственный интеллект: современный подход / Пер. с английского – М.: Вильямс, 2007. – 1408 с.

49. Цмоць І.Г. Принципи розробки і оцінка основних характеристик високопродуктивних процесорів на надвеликих інтегральних схемах/ Вісник ДУ “Львівська політехніка”, №349, Львів, 1998 - с.5-11.

50. Батюк А.Є., Цмоць І.Г. Методи синтезу спеціалізованих обчислювальних систем для розв’язання задач у реальному часі / Інформаційні технології і системи. Т2, №1, Львів 1999 – с.155-161.

51. Данілов П.О., Цмоць І.Г., Ігнатєв І.В. Апаратна реалізація обчислення максимального і мінімального чисед в масиві даних/ Сучасні комп’ютерні інформаційні технології.