УДК 004.2

**А.О. Гриценко** (асист.)**, С.Ю. Сіроштан** (інженер)**,**
**Ю.Є. Зінченко** (канд.техн.наук, доц.)
Донецький національний технічний університет
anthony.grytsenko@gmail.com

# МЕТОД ОПТИМІЗАЦІЇ АЛГОРИТМУ ШВИДКОГО СОРТУВАННЯ ДЛЯ СИСТЕМ НА ПЛІС

Задача сортування та алгоритми, що можуть бути застосовані для її вирішення, є загальновідомою і розглядається у великій кількості фундаментальних джерел. Деякі із зазначених алгоритмів застосовуються частіше за інші і їх оптимізація стала темою для відомих робіт в області комп'ютерних алгоритмів. Використовувані натепер реалізації цих алгоритмів рідко потребують подальшої програмної оптимізації. Але залишається можливість їх апаратної оптимізації, особливо, у разі використання таких алгоритмів у системах на програмованих кристалах. Такі системи вирізняються наявністю апаратних ресурсів, що можуть видозмінюватися за потребою розробника системи.

У даній статті запропоновано та досліджено метод оптимізації загальновідомого алгоритму швидкого сортування із використанням апаратних ресурсів ПЛІС FPGA для застосування в рамках розгорнутої на ній системи на ПЛІС.

**сортування, система на ПЛІС, програмно-апаратна система**

## *Вступ*

Задача сортування, яку визначено в [1, 2] як упорядкування вихідної послідовності $\langle a_1, a_2, \ldots, a_n \rangle$ так, щоб отримати послідовність $\langle a_1', a_2', \ldots, a_n' \rangle$, для якої вірно $a_1' \le a_2' \le \ldots \le a_n'$, є загальновідомою і детально розглядається, зокрема, в [2].

При створенні програмного забезпечення найчастіше застосовується метод швидкого сортування [1, 2], реалізації якого, наприклад, входять до складу стандартних бібліотек мов програмування C [3] та C++ [4]. Даний алгоритм розвивається на протязі довготривалого періоду, що обумовило розробку великої кількості варіантів його оптимізації.

Одним із основних недоліків алгоритму швидкого сортування є його низька ефективність при роботі із невеликими масивами. В [5] розглянуто аспекти реалізації даного алгоритму сортування задля його застосування в реальних програмних системах. У цій роботі також було розглянуто різні варіанти оптимізації даного алгоритму. Пізніше, дослідження практичних аспектів реалізації та оптимізації цього алгоритму було продовжено в [6]. Зауважимо, що в [6] аналізується реалізації функції **`qsort`**, яка є частиною стандартної бібліотеки мови програмування C [3].

У обох дослідженнях [5, 6] низька ефективність алгоритму швидкого сортування при роботі із невеликими масивами, які в будь-якому випадку виникають в процесі розподілення сортованого масиву, компенсується за рахунок використання алгоритму сортування методом вставок [1, 2]. У цій

роботі пропонується варіант вирішення зазначеної проблеми за допомогою спеціалізованого модулю апаратного сортування.

Задача апаратного сортування, як і відповідна їй задача програмного сортування, має декілька варіантів вирішення. Наприклад, в [2] досліджено так звані мережі сортування, яким також присвячено роботи К.Є. Батчера, зокрема, [7]. Даний метод апаратного сортування не буде застосовуватися у цій роботі, бо він передбачає, по-перше, значні затрати ресурсів, оцінки яких надано в [2, 7], по-друге, є за природою паралельним, а тому потребує великої кількості інформаційних входів та виходів. Альтернативним цьому є метод, який розглядається у [8] із посиланням на [9]. Він передбачає, по-перше, лінійну залежність між ресурсами, які є необхідними для розміщення апаратного модулю, та розміром масиву, по-друге, лінійну залежність між часом сортування та розміром масиву, по-третє, можливість послідовного запису вихідної послідовності та читання впорядкованої, по-четверте, цей модуль дозволяє сортувати масиви будь-якого розміру, обмежуючи тільки максимальну кількість елементів. Останній аспект важливий, зважаючи на те, що для алгоритму швидкого сортування задається лише максимальний розмір масиву, який повинен оброблятися більш підходящим чином, але не гарантується, що оброблятися будуть масиви меншого розміру.

Усі вихідні коди, розроблені для експериментального підтвердження ефективності пропонованого у роботі методу оптимізації, можна отримати за адресою: **http://code.google.com/p/fpga-hw-sw-sorter/**.

### *Постановка задачі*

Отже, алгоритм швидкого сортування має недоліки [1, 2], які можна усунути за допомогою декількох модифікацій [5, 6]. Один із його недоліків полягає у низькій ефективності при роботі із масивами невеликих розмірів. Цей недолік може бути усунений шляхом використання альтернативного механізму сортування, яким може слугувати, наприклад, спеціалізований апаратний модуль.

Реалізація програмної частини алгоритму сортування не є частиною цієї роботи, крім того, як вже зазначалося, на даний момент існує достатня кількість різноманітних її варіантів. Зважаючи на це, в якості початкового варіанту програмної реалізації алгоритму сортування обрано варіант, який входить до складу сучасної та широко застосовуваної GNU бібліотеки для мови програмування C [10]. Аналіз даного варіанту реалізації показав, що в ньому реалізовано підтримку алгоритму сортування вставками невеликих масивів. Необхідна модифікація передбачає заміну коду, який відповідає за сортування вставками, на код взаємодії із апаратним модулем.

Наступним кроком є створення апаратного модулю сортування. Цей модуль, по-перше, має інтегруватися із системою на ПЛІС, а, по-друге, він має надавати простий інтерфейс доступу з боку програмного забезпечення. В якості платформи у даній роботі було обрано ПЛІС FPGA, які належать сімейству Cyclone II фірми Altera [11]. Відповідно до обраної платформи,

система на ПЛІС буде базуватися на мікропроцесорі Altera NIOS II [12] та системній шині Altera Avalon [13]. Слід зауважити, що пропонований у цій роботі варіант реалізації апаратного модулю, за необхідністю, може бути перенесений на іншу платформу, що буде розглянуто далі.

Останнім кроком є розробка програмного забезпечення, необхідного задля тестування пропонованого варіанта оптимізації та збору інформації про його ефективність. В процесі аналізу ефективності у якості еталонного варіанту розглядається програмна реалізація алгоритму сортування, яка є частиною стандартної бібліотеки C для мікропроцесора Altera NIOS II.

### *Розробка апаратного модулю сортування*

Як зазначалося, у [8], базуючись на [9], розглянуто варіант структури модуля сортування. На рис. 1 показано концепцію роботи даного модуля. Відповідно неї сортувальник складається із кількох послідовно поєднаних функціональних блоків, кожен із яких зберігає два елементи, забезпечуючи при цьому їх впорядкування.

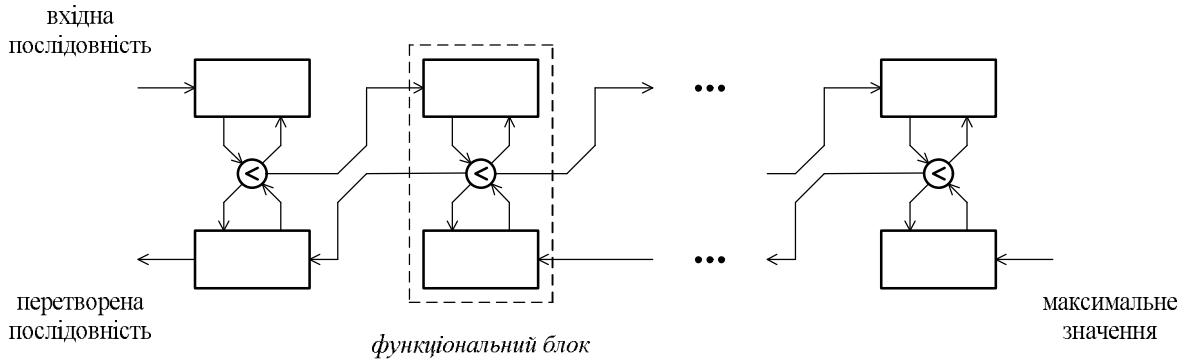


Рисунок 1 - Концепція модулю апаратного сортування

Зауважимо, що показана на рис. 1 концепція функціонального блоку відповідає запропонованій у [8]. Дану концепцію можна використати, щоб розрахувати очікувані затрати, які необхідні для розміщення модуля. Для цього визначимо, що задля сортування масивів, максимальний розмір яких складає $N$ елементів, необхідно $\frac{N+1}{2}$ (1) блоків (враховуючи, що $N$ може бути непарним).

$$K_{FB} = \left\lfloor \frac{N+1}{2} \right\rfloor. \qquad (1)$$

Далі потрібно визначити затрати, необхідні для розміщення одного блоку. На рис. 2 пропонується варіант схеми рівня регістрових передач для функціонального блоку.

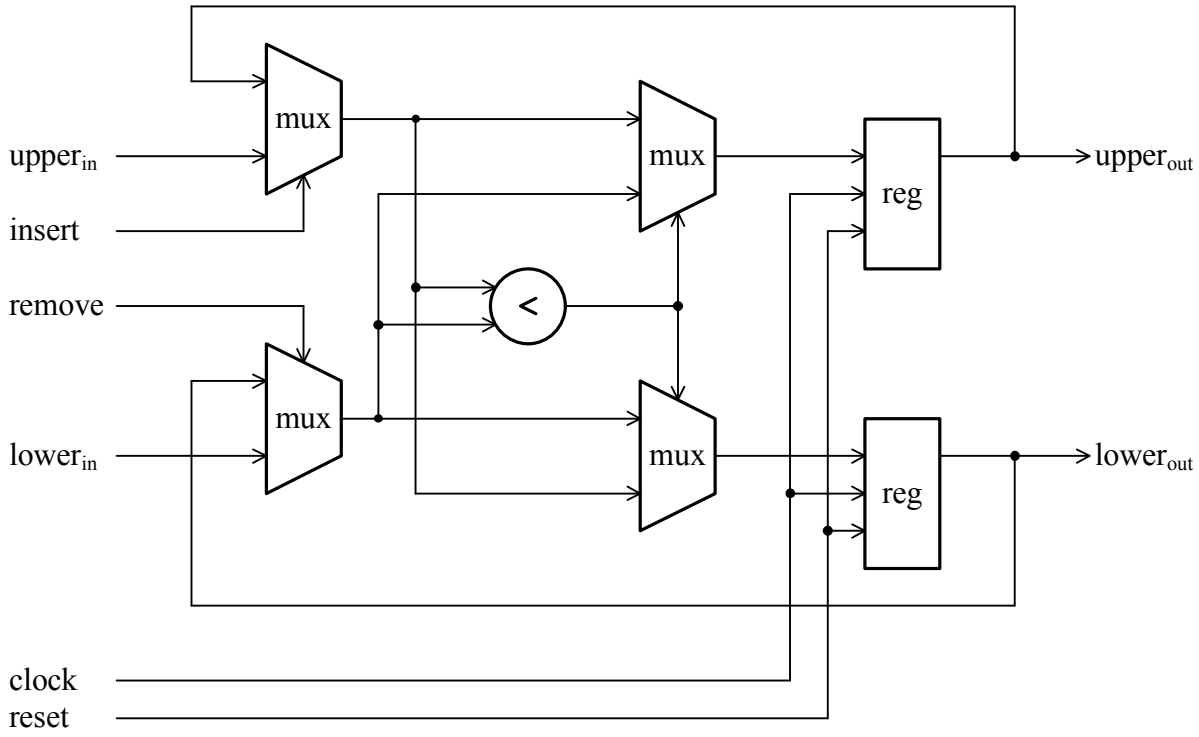


Рисунок 2 - Варіант схеми рівня ргістрових передач для функціонального блоку модуля сортування

Відповідно до пропонованого варіанту затрати на розміщення блоку можна розрахувати за наступною формулою:

$$L_{FB} = 4 \times L_{mux} + 2 \times L_{reg} + L_{cmp} . \quad (2)$$

Затрати на розміщення окремих елементів визначаються відповідно до використовуваної платформи та засобу синтезу схеми рівня ргістрових передач [14]. Наприклад, для ПЛІС FPGA сімейства Cyclone II (або більш пізнього) і засобу синтезу Altera Quartus II версії 10.0 [15], вони складають (з передумовою, що сортуються цілі числа без знаку, які задано $M$ бітами):

$$K_{LUT} = (5 \times K_{FB} - 1) \times M , \quad (3)$$

$$K_{FF} = 2 \times K_{FB} \times M . \quad (4)$$

де $K_{LUT}$ – необхідна кількість комбінаційних схем, а $K_{FF}$ – необхідна кількість тригерних схем. Слід звернути увагу, що кількість комбінаційних схем на одну менше, аніж очікувалось, у відповідності до схеми на рис. 2. Це пов’язано з тим, що спосіб синтезу спростив схему для останнього блоку, прибравши один із комутаторів.

На поточний момент, для опису апаратних модулів для ПЛІС широко застосовується мова VHDL [16]. Ця мова дозволяє специфікувати апаратні модулі на кількох рівнях абстракції. Найбільш високим рівнем абстракції є опис поведінки модуля, який не включає ніяких подробиць, що стосуються структури цього модуля. Поведінковий опис достатньо легко переносити з

однієї платформи на іншу. Також поведінку деяких модулів можна описати дуже стисло, у порівнянні із описом їх структурної моделі.

Основний недолік поведінкового опису полягає в тому, що достатньо складно спрогнозувати результати його синтезу. Тому треба перевіряти їх для кожної окремої апаратної платформи та засобу синтезу.

### *Інтеграція модулю сортування до системи на ПЛІС*

Наступним кроком є інтеграція розробленого модуля сортування до системи на ПЛІС. Як було зазначено раніше, у даній роботі застосовується система на ПЛІС на базі мікропроцесора Altera NIOS II та системної шини Altera Avalon. На рис. 3 показано структурну схему підключення модуля до зазначеної системної шини процесора.

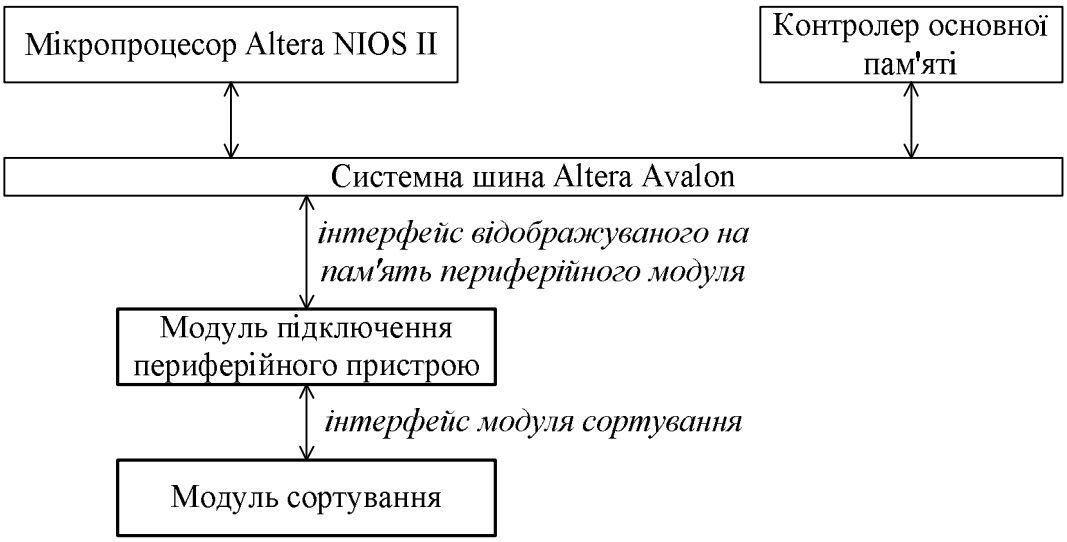


Рисунок 3 - Структурна схема підключення периферійного модуля до системної шини Altera Avalon

Системна шина Altera Avalon позволяє застосовувати декілька різних інтерфейсів для модулів, які підключаються до неї [13]. Для підключення розробленого пристрою можна застосувати інтерфейс відображуваного на пам’ять периферійного модуля. В такому разі, програмні модулі зможуть взаємодіяти із пристроєм за посередництва набору портів, кожний із яких відповідає унікальній комірці пам’яті.

У даному випадку програмний модуль буде взаємодіяти із апаратним через два порти:

- перший із них призначений для послідовного запису елементів вихідної послідовності;
- а другий, відповідно – призначений для послідовного читання елементів перетвореної послідовності.

Щоб відмежувати предметну логіку від логіки взаємодії з системною шиною, підключення виконується через додатковий модуль, який відіграє роль шлюзу. Такий підхід дозволяє, зокрема, перейти до іншої системи на ПЛІС, замінивши тільки модуль підключення до шини.

На рис. 4 наведено розширений варіант структури системи на ПЛІС, який застосовувався для тестування та оцінки ефективності пропонованої оптимізації. Цей варіант відповідає показаному на рис. 3, але додатково він включає таймер високого розділення для заміру часу роботи еталонного й модифікованого алгоритмів та контролер JTAG для передачі отриманої в процесі експериментів інформації.

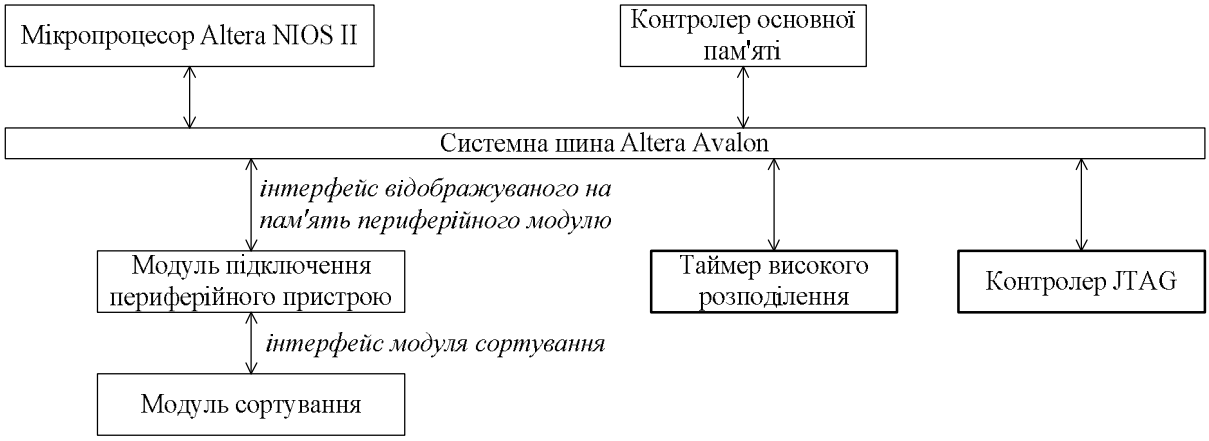


Рисунок 4 – Структурна схема системи на ПЛІС, використовуваної для тестування та оцінки ефективності використання модуля

### *Дослідження ефективності сортування з застосуванням програмно-апаратного модуля*

Для дослідження ефективності пропонованої оптимізації визначено наступні критерії:

- модифікація мікропроцесора, яка визначає його ефективність, що, відповідно, впливає на ефективність виконання програмної частини алгоритму сортування;
- співвідношення розміру сортованого масиву та максимального розміру масиву, який може впорядковуватися апаратно;
- початкові дані, в залежності від яких алгоритм може змінювати свою ефективність [1].

Дослідження проводилися із використанням налагоджувальної плати Altera DE2 [17] із ПЛІС FPGA Cyclone II моделі EPC35F672C6 та 512 КБ SRAM моделі ISSI IS61WV25616.

У якості початкових даних використовувалися:

- попередньо впорядкований масив;
- генерований випадковим чином масив (використано середнє 30 експериментів).

У табл. 1 та табл. 2 наведені результати для мікропроцесора Altera NIOS II/e. Відповідні графіки показано на рис. 5 та рис. 6.В свою чергу, у табл. 3 та табл. 4 наведені результати для мікропроцесоруaAltera NIOS II/f. Відповідні графіки показано на рис. 7 та рис. 8.

Таблиця 1. Результати досліджень (процесор Altera NIOS II/e, вхідні дані – попередньо впорядкований масив)

| Максимальний розмір масиву, який може впорядковуватися апаратно | Розмір початкового масиву | Еталонний результат | Результат програмно-апаратного модулю |
|---|---|---|---|
| 8 | 128 | 1416 | 2123 |
| | 256 | 2763 | 4868 |
| | 512 | 5453 | 11007 |
| | 1024 | 10825 | 24521 |
| 16 | 128 | 1416 | 1568 |
| | 256 | 2763 | 3760 |
| | 512 | 5453 | 8779 |
| | 1024 | 10825 | 20081 |
| 32 | 128 | 1416 | 1109 |
| | 256 | 2763 | 2843 |
| | 512 | 5453 | 6944 |
| | 1024 | 10825 | 16393 |
| 64 | 128 | 1416 | 710 |
| | 256 | 2763 | 2046 |
| | 512 | 5453 | 5343 |
| | 1024 | 10825 | 13207 |

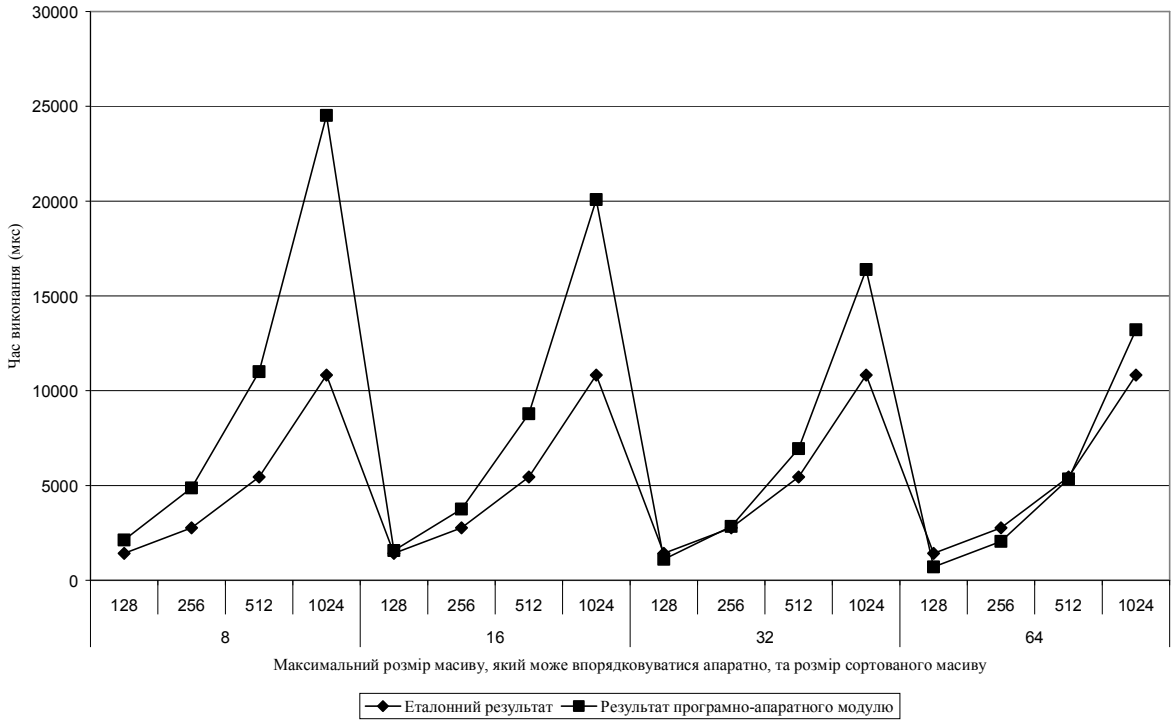


Рисунок 5 - Результати досліджень (процесор Altera NIOS II/e, вхідні дані – попередньо впорядкований масив)

Таблиця 2. Результати досліджень (процесор Altera NIOS II/e, вхідні дані – генерований випадковим чином масив, середнє для 30-х експериментів)

| Максимальний розмір масиву, який може впорядковуватися апаратно | Розмір початкового масиву | Еталонний результат | Результат програмно-апаратного модулю |
|---|---|---|---|
| 8 | 128 | 4792 | 4364 |
| | 256 | 10724 | 10186 |
| | 512 | 23527 | 22882 |
| | 1024 | 51990 | 51758 |
| 16 | 128 | 4792 | 3530 |
| | 256 | 10724 | 7984 |
| | 512 | 23527 | 18341 |
| | 1024 | 51990 | 43062 |
| 32 | 128 | 4792 | 2390 |
| | 256 | 10724 | 6352 |
| | 512 | 23527 | 15235 |
| | 1024 | 51990 | 35616 |
| 64 | 128 | 4792 | 1482 |
| | 256 | 10724 | 4445 |
| | 512 | 23527 | 12419 |
| | 1024 | 51990 | 28075 |

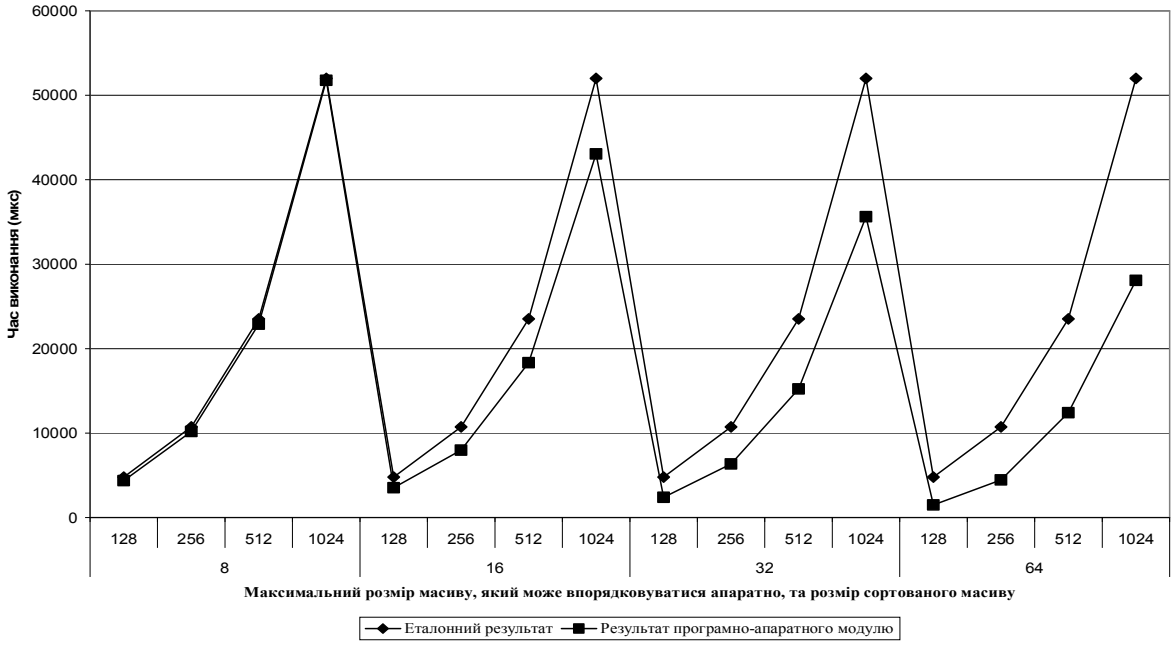


Рисунок 6 - Результати досліджень (процесор Altera NIOS II/e, вхідні дані – генерований випадковим чином масив, середнє для 30-х експериментів)

Таблиця 3. Результати досліджень (процесор Altera NIOS II/f, вхідні дані – попередньо впорядкований масив)

| Максимальний розмір масиву, який може впорядковуватися апаратно | Розмір початкового масиву | Еталонний результат | Результат програмно-апаратного модулю |
|---|---|---|---|
| 8 | 128 | 202 | 327 |
| | 256 | 390 | 748 |
| | 512 | 772 | 1690 |
| | 1024 | 1586 | 3810 |
| 16 | 128 | 202 | 247 |
| | 256 | 390 | 588 |
| | 512 | 772 | 1368 |
| | 1024 | 1586 | 3170 |
| 32 | 128 | 202 | 179 |
| | 256 | 390 | 451 |
| | 512 | 772 | 1093 |
| | 1024 | 1586 | 2617 |
| 64 | 128 | 202 | 119 |
| | 256 | 390 | 333 |
| | 512 | 772 | 854 |
| | 1024 | 1586 | 2143 |

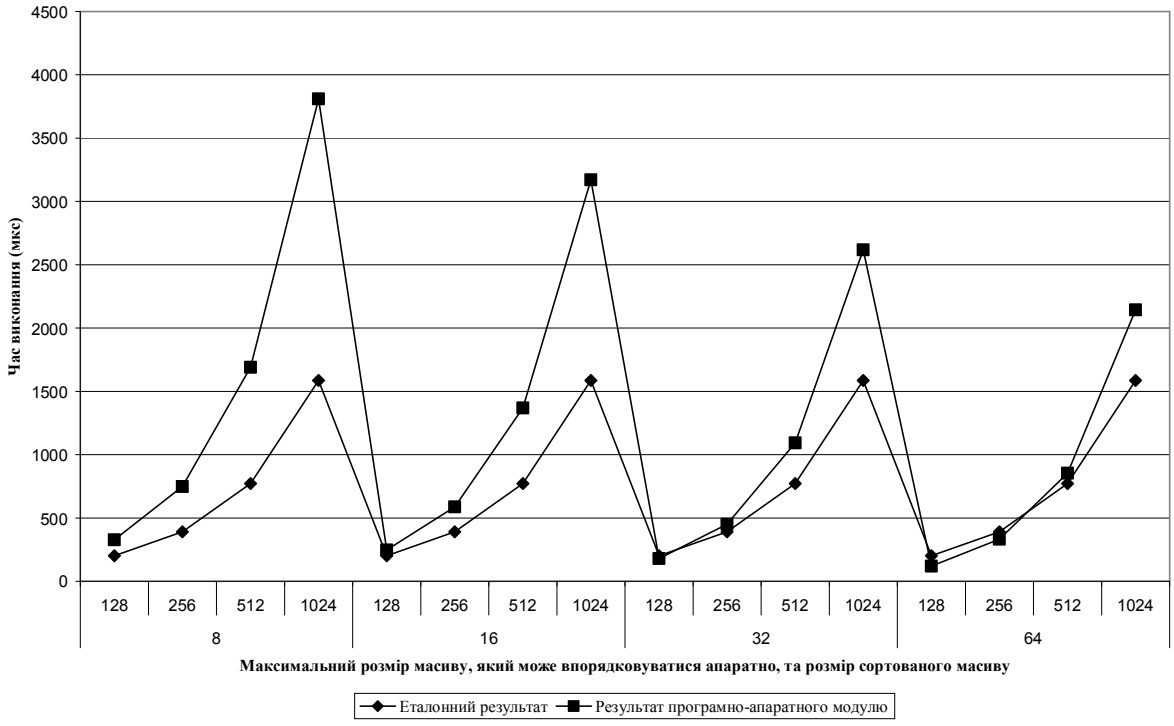


Рисунок 7 - Результати досліджень (процесор Altera NIOS II/f, вхідні дані – попередньо впорядкований масив)

Таблиця 4. Результати досліджень (процесор Altera NIOS II/f, вхідні дані – генерований випадковим чином масив, середнє для 30-х експериментів)

| Максимальний розмір масиву, який може впорядковуватися апаратно | Розмір початкового масиву | Еталонний результат | Результат програмно-апаратного модулю |
|---|---|---|---|
| 8 | 128 | 622 | 597 |
| | 256 | 1427 | 1374 |
| | 512 | 3129 | 3043 |
| | 1024 | 7130 | 6938 |
| 16 | 128 | 622 | 483 |
| | 256 | 1427 | 1076 |
| | 512 | 3129 | 2436 |
| | 1024 | 7130 | 5767 |
| 32 | 128 | 622 | 328 |
| | 256 | 1427 | 861 |
| | 512 | 3129 | 2030 |
| | 1024 | 7130 | 4776 |
| 64 | 128 | 622 | 208 |
| | 256 | 1427 | 613 |
| | 512 | 3129 | 1660 |
| | 1024 | 7130 | 3787 |

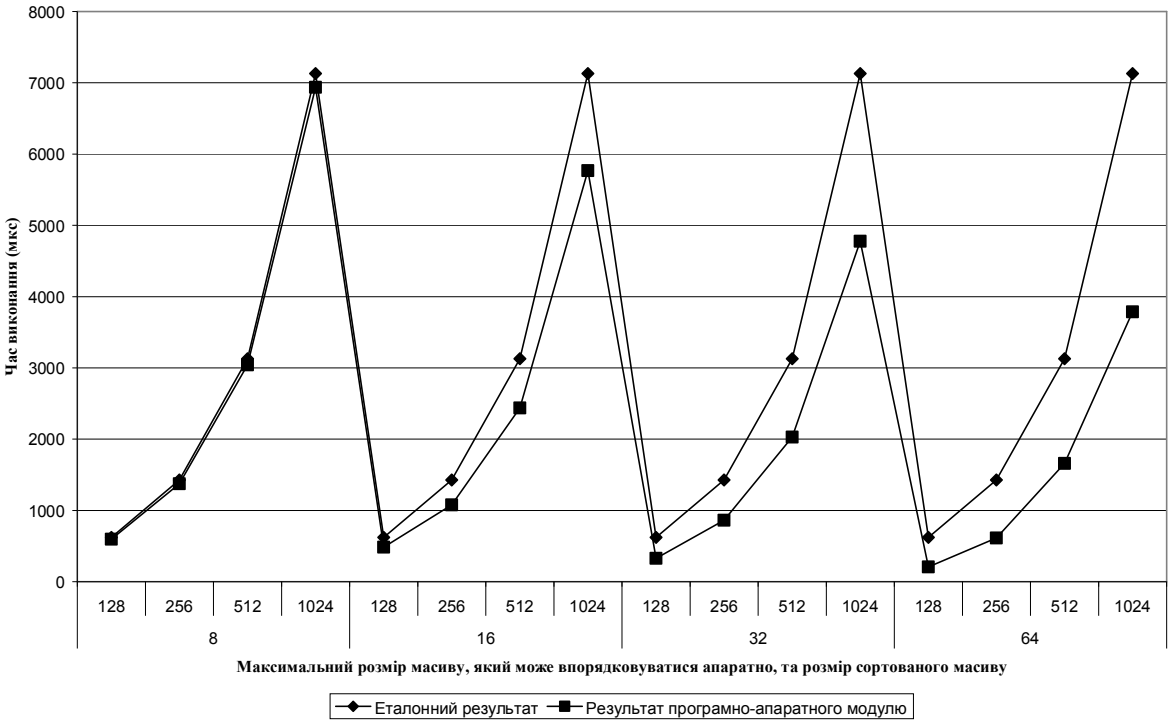


Рисунок 8- Результати досліджень (процесор Altera NIOS II/f, вхідні дані – генерований випадковим чином масив, середнє для 30-х експериментів)

### *Аналіз отриманих результатів*

У табл. 1 та табл. 3 (відповідно на рис. 5 та рис. 7) надано результати досліджень швидкодії запропонованого програмно-апаратного модулю для специфічних даних –попередньо відсортованого масиву. Як і очікувалось, в даному випадку еталона реалізація має перевагу. Причина цього полягає в тому, що, по-перше, еталонний варіант може мати оптимізацію для таких вхідних даних, по-друге, навіть у загальному випадку алгоритм сортування вставками в такому випадку витрачає менше часу [1, 2]. Запропонований варіант не має оптимізації для такого випадку, а тому час роботи апаратної частини не зменшується в залежності від вхідних даних.

З іншого боку, застосування запропонованого методу оптимізації дає можливість отримати зиск у загальному випадку, що показано у табл. 2 та табл. 4 (відповідно на рис. 6 та рис. 8). При цьому необхідно зазначити, що наявна залежність поміж максимальним розміром масиву, який може бути впорядкований апаратно та вихідним розміром вихідного масиву.

Відповідно до проведених досліджень можна сказати, що бажаним є, щоб співвідношення поміж максимальним розміром масиву, сортованого апаратно, та розміром вихідного масиву складало десь 1 до восьми. Дана оцінка не є точною і отримана на основі проведених досліджень.

Використання пропонованого методу оптимізації вигідно незалежно від типу використовуваного мікропроцесора.

### *Висновки*

У цій роботі запропоновано метод оптимізації алгоритму сортування на основі застосування апаратного модулю для систем на ПЛІС. На основі проведених досліджень можна визначити, що пропонований метод можна ефективно застосовувати у реальних системах.

В подальшому передбачається проведення значно більш детальних досліджень роботи пропонованого методу задля визначення усіх аспектів його застосування. Зокрема, мова іде про потребу у розширенні множини критеріїв оцінки ефективності оптимізації, точному визначенні відношення між максимальним розміром масиву, який можна впорядкувати апаратно та вихідним розміром сортованого масиву і таке інше.

### *Список літератури*


1. Кнут Дональд. Искусство программирования: т. 3. Сортировка и поиск / Кнут Дональд; 2-е изд.: пер. с англ. – М. : ООО «И.Д. Вильямс», 2007. – 832с. : ил.
2. Алгоритмы построение и анализ / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. – М. : Издательский дом «Вильямс», 2007. – 1296 с. : ил.
3. Programming languages – C: ISO/IEC 9899:1999(E) – ISO/IEC 9899/1999(E) – [Чинний від 01.12.1999] – ISO/IEC, 1999. – 538 pp. – International Standard.

4. Programming languages – C++: ISO/IEC 14882:2003 – ISO/IEC 14882:2003 – [Чинний від 01.12.2003] – ISO/IEC, 2003. – 650 pp. – International Standard.
5. Sedgewick Robert. Implementing Quicksort programs / R. Sedgewick. – «Brown University», 1987. – 11 pp.
6. Bentley Jon L. Engineering a Sort Function. J. L. Bentley, M. D. Mcilroy. – AT&T «Bell Laboratories», 1993. – 17 pp.
7. Batcher K. E. Sorting networks and their applications / K. E. Batcher. – «Goodyear Aerospace Corporation», 1968. – 8 pp.
8. Палагин А.В. Реконфигурируемые вычислительные системы: Основы и приложения / А. В. Палагин, В. Н Опанасенко. – К.: Просвіта, 2006. – 280 с.: ил.
9. Choy Y. H. Easily reconfigurable VLSI sorter / Int. Journal Electronics. – 1965 –. – ISSN 1362-3060. 1990, Vol. 69, N. 3.
10. GNU C Library [Electronic Resource] – Mode of access: http://ftp.gnu.org/gnu/glibc/ – Title from the screen.
11. Cyclone II Device Handbook Volume 1 [Electronic Resource] – Mode of access : http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf – Title from the screen.
12. Nios II Processor Reference Handbook [Electronic Resource] – Mode of access : http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf – Title from the screen.
13. Avalon Interface Specifications [Electronic Resource] – Mode of access : http://www.altera.com/literature/manual/mnl_avalon_spec.pdf – Title from the screen.
14. Smith Douglas J. HDL Chip Design: A Practical Guide for Designing, Synthesizing & Simulating Asics & FPGAs Using VHDL or Verilog / D. J. Smith. – M. : «Doone Publications», 1996. – 447 pp.
15. Quartus II Handbook v10.0 [Electronic Resource] – Mode of access: http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf – Title from the screen.
16. Ashenden Peter J. The Designer's Guide to VHDL / P. J. Ashenden. – SF. : «Morgan Kaufmann Publishers, Inc.», 1995. – 688 pp.
17. DE2 Development and Education Board [Electronic Resource] – Mode of access : http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html – Title from the screen.

*Надійшла до редакції 28.09.2010* *Рецензент: канд.техн.наук, доц. Дяченко О.Н.*

**Гриценко А.А., Сероштан С.Ю., Зинченко Ю.Е.**
Донецкий национальный технический университет

**Метод оптимизации алгоритма быстрой сортировки для систем на ПЛИС**. Задача сортировки и алгоритмы, которые могут быть использованы для ее решения, являются общеизвестными и рассматриваются в ряде фундаментальных источников. Некоторые из них используются чаще других, и их оптимизация стала темой для известных работ в области компьютерных алгоритмов. Используемые на данный момент реализации этих алгоритмов редко нуждаются в дальнейшей программной оптимизации. Но, при этом, остается возможность для аппаратной оптимизации, особенно, в случае использования этих алгоритмов в системах на ПЛИС. Эти системы отличаются наличием аппаратных ресурсов, которые могут видоизменяться по требованию разработчика.
В этой статье предложен и исследован метод оптимизации общеизвестного алгоритма быстрой сортировки путем использования ресурсов ПЛИС FPGA для использования в рамках развертываемой на ней системы на ПЛИС.
**сортировка, система на ПЛИС, программно-аппаратная система**

**Grytsenko A., Seroshtan S., Zinchenko J.**
Donetsk National Technical University

**Optimization Method for Quicksort on SoPC**. The sorting problem and its solutions are well-known and considered in a number of fundamental sources. Some of appropriate algorithms are used more frequently then others. But still there is a possibility of optimizing these algorithms by the use of hardware, especially, when we use the FPGAs. Such systems allow a designer to configure hardware in accordance with his demands.
In this article we propose an optimization method for the well-known quick sort algorithm. This method is based on the usage of FPGA's reconfigurable resources. It is applicable for modern SoPC which are deployed on these FPGAs.
**sorting problem, SoPC, hardware-software system**