УДК 004.415.5:519.8

ЕФЕКТИВНІСТЬ ОПТИМІЗАЦІЇ КОДУ В СЕРЕДОВИЩІ ВИСОКОРІВНЕВОГО СИНТЕЗУ НА ПЛАТФОРМІ SOC ZYNQ-7000 У ПОРІВНЯННІ З СУЧАСНИМИ ПРОЦЕСОРНИМИ АРХІТЕКТУРАМИ

Кармanський Б.Ю.
email: karmanskybogdan@gmail.com
Науковий керівник – к.т.н., доц. Шкіль О.С.
Харківський національний університет радіоелектроніки
(61166, Харьков, просп. Науки,14, каф. АПОТ, тел. (057) 702-13-26)
email: oleksandr.shkil@nure.ua

The thesis investigates the efficiency of code optimization on the SoC Zynq-7000 platform in comparison with modern CPU architectures such as AMD Ryzen 9, Apple M1 and Raspberry Pi. Experiments were carried out for matrix multiplication (classical and Winograd), fast Fourier transform and wavelet transforms using Haar and Daubechies wavelets. Several optimization models were analyzed, including memory access optimization, loop unrolling, function inlining and their combinations. The results demonstrate a significant reduction of execution time. The obtained recommendations can be used to design high-performance SoC-based systems for signal processing and other computationally-loaded algorithms.

Вступ. Платформа SoC Zynq-7000 поєднує двоядерний процесор ARM Cortex-A9 та програмовану логіку FPGA, що дозволяє реалізовувати як програмні, так і апаратні прискорювачі в єдиній системі. Така архітектура є особливо актуальною для задач цифрової обробки сигналів та лінійної алгебри, де важливо забезпечити високу продуктивність при обмежених апаратних ресурсах. Було виконано експериментальне дослідження ефективності оптимізації коду на платформі Zynq-7000 та проведено порівняння з сучасними процесорними архітектурами, такими як AMD Ryzen 9, Apple M1 та Raspberry Pi. Основна увага приділялася зменшенню часу виконання обчислювально-навантажених алгоритмів за рахунок оптимізації доступу до пам’яті, розкриття циклів, інлайнінгу та їх поєднання.

Алгоритми дослідження. Експерименти проводилися для кількох класів алгоритмів: множення матриць (традиційний алгоритм та алгоритм Віноградa), пряме та зворотне швидке перетворення Фур’є, а також пряме та зворотне вейвлет-перетворення з використанням вейвлетів Гаара та Добеші. Алгоритми множення матриць є базовими для численних задач лінійної алгебри та машинного навчання, тому їх продуктивність безпосередньо впливає на швидкодію багатьох прикладних систем. Швидке перетворення Фур’є використовується для аналізу частотних характеристик

сигналів і є класичним прикладом обчислювально складного алгоритму, чутливого до організації доступу до пам’яті. Вейвлет-перетворення Гаара та Добеші застосовуються для багаторівневого аналізу сигналів та зображень; вейвлет Гаара має простішу структуру, тоді як Добеші характеризується більшою обчислювальною складністю та кращими апроксимаційними властивостями.

Результати оптимізації на платформі Zynq-7000. Для традиційного множення матриць базовий час виконання на Zynq-7000 становив 45,52 мс; оптимізація доступу до пам’яті зменшила його до 40,96 мс, а поєднання оптимізацій дало найкращий результат 36,74 мс. Для алгоритму Віноградa базовий час 38,69 мс було знижено до 34,24 мс за рахунок оптимізації доступу до пам’яті та до 31,45 мс при комбінуванні оптимізацій. Пряме вейвлет-перетворення Гаара з 9 мс скоротилося до 8,17 мс при оптимізації пам’яті та до 6,78 мс при поєднанні оптимізацій, тоді як зворотне перетворення виявилося менш чутливим і майже не змінювало час виконання, а комбіновані оптимізації навіть погіршили його до 8,11 мс. Для вейвлета Добеші пряме перетворення було прискорено з 9,6 мс до 6,25 мс (оптимізація пам’яті) і до 5,02 мс (поєднання оптимізацій), а зворотне – з 9,41 мс до 6,17 мс і 6,01 мс відповідно. Найбільший виграш спостерігався для перетворення Фур’є: пряме Фур’є було прискорено з 54,45 мс до 8,91 мс, а зворотне – з 55,94 мс до 9,8 мс завдяки поєднанню оптимізацій, тоді як проста оптимізація доступу до пам’яті в окремих випадках навіть погіршувала результат.

Порівняння платформ за зведеними результатами. Приріст швидкодії від оптимізацій для всіх алгоритмів і платформ у відсотковому прискоренні щодо оригінального виконання відображено у таблиці 1. Загальний аналіз показує, що оптимізація доступу до пам’яті забезпечує найбільш помітне покращення для алгоритмів множення матриць, особливо на платформах із відносно слабшими процесорними ресурсами, таких як Raspberry Pi та Zynq-7000. Розкриття циклів має неоднозначний ефект і в ряді випадків для вейвлет- та Фур’є-перетворень на Raspberry Pi та Zynq не призводить до очікуваного прискорення через перевантаження кешу інструкцій. Інлайнінг виявився найефективнішим для перетворення Фур’є, особливо на потужних процесорах AMD Ryzen 9 та Apple M1, де зменшення кількості викликів функцій суттєво скорочує час виконання. Майже в усіх випадках найкращий результат дає поєднання кількох оптимізацій, що дозволяє одночасно зменшити кількість звернень до пам’яті, покращити використання кешу та знизити накладні витрати на управління обчисленнями.

Таблиця 1 – Зведені результати ефективності оптимізацій у відсотках

Оптимізація	Алгоритм	Ефективність оптимізації на процесорі в %			
		AMD Ryzen 9	Apple M1	Raspberry PI	Zynq-7000
Оптимізація доступу до пам'яті	Множення матриць	33,2	0,6	17,4	10,7
	Вейвлет	3,4	0	11,7	21,3
	Фур'є	-7,3	-67,8	1,9	-4,7
Розкриття циклів	Множення матриць	2,7	1,3	-0,3	5,8
	Вейвлет	7,1	-64,3	-21,6	-1,6
	Фур'є	-0,6	-38,4	0,6	1,8
Інлайнінг	Вейвлет	-9,1	14,3	11,1	9,2
	Фур'є	63,1	55,3	87	82
Поєднання оптимізацій	Множення матриць	37,2	3,4	18,8	19
	Вейвлет	19,2	-14,3	14,8	25,5
	Фур'є	62,8	65,5	86,7	83

Висновки. На основі експериментального дослідження було реалізовано та проаналізовано низку моделей оптимізації, орієнтованих на підвищення продуктивності алгоритмів множення матриць, перетворення Фур’є та вейвлет-перетворень на платформі SoC Zynq-7000. Отримані результати продемонстрували суттєве скорочення часу виконання завдяки зниженню кількості звернень до пам’яті, особливо при комбінуванні різних оптимізаційних підходів. Використання інструментів Xilinx Vivado та Vitis забезпечило інтеграцію реалізованих алгоритмів з апаратною частиною платформи та спростило процес розробки й тестування оптимізаційних моделей. Розроблені підходи й отримані рекомендації можуть бути застосовані під час створення високопродуктивних систем у галузях обробки сигналів, аналізу зображень, телекомунікацій та мультимедіа, а також слугувати основою для подальших досліджень у напрямі оптимізації коду для SoC-платформ.

Список використаних джерел:

1. The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC / L. H. Crockett та ін. Glasgow, Scotland, UK : Department of Electronic and Electrical Engineering University of Strathclyde, 2014. 460 с.

2. Шкіль О.С. Оптимізація програмного коду для високорівневого синтезу при апаратній реалізації обчислювально-навантажених алгоритмів / О.С. Шкіль, О.І. Філіпенко, Д.Ю. Рахліс, І.В. Філіпенко, В.Р. Корнієнко // Сучасний стан наукових досліджень та технологій в промисловості. – 2025. – No 3 (33). – С. 189-202. DOI: https://doi.org/10.30837/2522-9818.2025.3.189