ISSN 2079-0031 (Print) ISSN 2411-0558 (Online)

УДК 004. 383.3 **DOI: 10.20998/2411-0558.2025.01.08**

О. С. ШКІЛЬ, канд. техн. наук, с.н.с., доц., ХНУРЕ,
І. В. ФІЛІПЕНКО, д канд. техн. наук, доц., доц, ХНУРЕ,
А. М. МІРОШНИК, канд. техн. наук, ст. викладач, ХНУРЕ,
В. Р. КОРНІЄНКО, аспірант, ХНУРЕ

АНАЛІЗ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ СПЕЦІАЛІЗОВАНИХ ОБЧИСЛЮВАЛЬНИХ МОДУЛІВ ТА БІБЛІОТЕК ПРИ РЕАЛІЗАЦІЇ АЛГОРИТМІВ ЦИФРОВОЇ ОБРОБКИ СИГНАЛІВ НА ПЛАТФОРМІ SOC

Розглянута реалізація алгоритму швидкого перетворення Фур'є (FFT) на технологічній платформі системи на кристалі SoC Xilinx ZYNQ-7000. Виконано порівняння часових та апаратурних витрат на реалізацію алгоритму FFT для різних варіантів його довжини на PS частині SoC з використанням спеціалізованих бібліотек, та реалізація цього алгоритму на PL частині SoC з використанням IP-core з репозиторію Xilinx. Реалізація алгоритму FFT виконана на мові програмування С з використанням інструментів Vivado/Vitis. Найкращу швидкодію серед різних способів реалізації алгоритму FFT показало використання стандартного IP-core Xilinx HLS FFT IP для PL частини SoC Xilinx ZYNQ-7000. Іл. 7, Бібл. 11 назв.

Ключеві слова: вбудовані системи, системи на кристалі, вбудоване програмне забезпечення, швидке перетворення Фур'є, мова програмування С, швидкодія, пам'ять, оптимізація, спеціалізовані бібліотеки, програмована логіка, IP-core.

Вступ. Вбудовані системи широко використовуються в різних галузях науки і техніки та застосовуються для вирішення завдань управління, обробки сигналів, що надходять з датчиків, цифрової обробки мультимедійних даних та інших завдань. Вони знаходять широке застосування в побутовій електроніці, промисловій автоматиці, на транспорті, телекомунікаційних системах, медичному устаткуванні, у аерокосмічній техніці тощо. При цьому головним критерієм якості функціонування вбудованої системи є продуктивність, а основною вимогою до продуктивності є забезпечення швидкості виконання заданого набору функцій за відведений час, що забезпечує функціонування системи в реальному часі. При цьому швидкодія такої системи є не тільки загальним показником продуктивності, але й показником коректності роботи системи у реальному часі.

© О.С. Шкіль, І.В. Філіпенко, А.М. Мірошник, В.Р. Корнієнко, 2025

Вбудовані системи, які орієнтовані на вирішення конкретної задачі, мають обмежені ресурси з точки зору обчислювальної потужності, пам'яті та енергоспоживання. Ці обмеження вимагають ретельної оптимізації та управління ресурсами (вбудована пам'ять та спеціалізовані обчислювальні блоки) при проектуванні вбудованих систем. Збільшити продуктивність вбудованої системи (час виконання цільової задачі) можна двома способами: апаратним та програмним. Апаратний спосіб – це модернізація апаратної платформи, але це, як правило, входить у протиріччя з ціновими та енергетичними обмеження системи, що проектується. Програмний спосіб збільшення продуктивності вбудованої системи полягає в оптимізації операційної системи та програмного коду, який реалізує цільовий алгоритм. Якщо тип операційної системи обумовлений технічним завданням, то на перше місце виходить оптимізація програмного коду. Серед загальних підходів до оптимізації програмного коду можна виділити наступні: використання ефективніших алгоритмів з усуненням надлишкових операцій та звернень до пам'яті, використання ефективніших структур даних, використання більш низькорівневих систем зі спеціальними бібліотеками, зниження точності обчислювального алгоритму [1]. Не всі ці підходи можуть застосовуватися для вбудованого програмного забезпечення та їх ефективність обумовлена конкретним цільовим алгоритмом. Якщо в якості апаратної платформи вбудованої системи використовуються системи на кристалі (System on Chip, SoC) Xilinx ZYNQ-7000. яка має архітектуру, де поєднано процесорну систему обробки (PS) та програмовану логіку (PL), то доцільно розглядати задачу оптимізації цільового програмного коду як для PS так і PL частин SoC.

При реалізації алгоритмів цифрової обробки сигналів (ЦОС) на платформі SoC Xilinx ZYNQ-7000 для оптимізації програмних обчислень використовується розподіл обчислювальних витрат алгоритмів ЦОС між PL та PS частинами SoC [2]. Але актуальним залишається питання оптимізації швидкодії алгоритмів ЦОС, зокрема перетворення Фур'є, при мінімізації витрат пам'яті з використанням спеціалізованих бібліотек PS частини SoC та репозиторію IP-core Xilinx для PL частини.

Аналіз останніх досліджень та літературних джерел. Питання оптимізації та ефективної реалізації алгоритму перетворення Фур'є на конкретній апаратній платформі є актуальним на поточний момент і йому

присвячено достатньо багато публікацій у спеціальній технічній літературі.

У дослідженні [3] представлено інструментальний засіб AutoFFT для автоматичної генерації високопродуктивних ядер FFT. Підхід поєднує алгоритмічні та апаратні оптимізації для ефективних обчислень. AutoFFT генерує як C-ядра, так і асемблерні ядра, забезпечуючи переносимість і продуктивність.

Дослідження використання HLS для розробки апаратних прискорювачів FFT розглянуто у роботі [4] де авторами підкреслюється важлива роль поведінкових моделей у створенні ефективних архітектур. Також, у роботі виконано аналіз продуктивності і апаратних витрат запропонованого рішення.

Питання оптимізації коду для HLS з цільовим використанням у HPC-додатках розглядається у дослідженні [5]. Авторами запропоновано бібліотеку примітивів для зменшення затримок пам’яті, масштабованості та оптимізації конвеєра. Аналізується вплив окремих технологій на продуктивність апаратних обчислювачів.

У статті [6] запропоновано ефективну апаратну архітектуру для двовимірного ковзного дискретного перетворення Фур’є (SDFT), яка орієнтована на мінімізацію використання апаратних ресурсів та оптимізацію продуктивності для задач реального часу. Запропонована архітектура значно скорочує апаратні витрати на реалізацію (на 25%), використовуючи лише вісім додавань і шість множень.

Метод на основі DL і FFT для покращення якості зображень у медичній візуалізації запропоновано авторами дослідження [7]. Використовується CNN для відновлення втраченої інформації та DDDWT для підвищення роздільної здатності. Метод поєднує OA-FFT, Wallace tree та оптимізацію RDO. Експерименти показали високу точність (до 98 %) і ефективність за PSNR, SSIM та енергоспоживанням.

У статті [8] розглянуто реалізацію FPGA- процесору FFT на 128k-точок для реального часу SAR-зйомки. Використовується чотирьохканальна структура SDF з алгоритмами radix-23 і змішаним radix для ефективної обробки. Реалізовано стратегію коригування довжини слова для підвищення точності. Експерименти показали відносну помилку $\sim 10^{-4}$ у порівнянні з MATLAB-симуляцією , що підтверджує високу точність.

Питання реалізації FPGA-архітектури для багатоканального STFT у системах EW розглядається у дослідженні [9] де реалізація на SoC використовує IP-блоки та Linux для високошвидкісної обробки. Система працює з радіочастотними сигналами та забезпечує швидке перетворення у частотну область. Експерименти на Zynq-7000 показали приріст продуктивності у 22 рази порівняно з процесорним підходом.

Проблеми продуктивності реалізації 3D FFT у багатоядерних процесорах через особливості ієрархії кешу та розрізненість доступів до пам'яті розглянуто у роботі [10]. Авторами запропоновано фреймворк OpenFFT, що оптимізує доступ до пам'яті за допомогою алгоритму Z-OpenFFT для покращення локалізації даних та section-cache-aware алгоритму для оптимізації мережі «метелик» у 1D FFT та моделі адаптивного розподілу потоків з урахуванням кеш-ієрархії.

Дослідження [11] розглядає продуктивність та споживання пам'яті різних відкритих BLAS-бібліотек на вбудованих процесорах, зокрема ARM SoC з SIMD-акселератором та першу комерційну систему на базі RISC-V. Авторами запропоновано масштабне тестування найбільшого на сьогодні набору BLAS-бібліотек з урахуванням особливостей вбудованих Linux-систем. Результатами є підтвердження суттєвого приросту продуктивності оптимізованих BLAS-реалізацій у порівнянні з чистими C-імплементаціями, а також перевага ARM-платформи над RISC-V у більшості тестів.

На основі проведеного аналізу можно зробити висновок, що не існує загальних підходів для оптимальної реалізації алгоритмів (ЦОС) для вбудованих систем. Таким чином актуальною задачею залишається аналіз характеристик спеціалізованих бібліотек SoC при реалізації алгоритмів ЦОС та вибір оптимальної бібліотеки як за параметрами швидкодії, так і за витратами оперативної пам'яті

Постановка задачі. Оскільки вбудована система призначена для виконання обмеженої кількості функцій, розробники можуть її оптимізувати, забезпечуючи ефективність технічного рішення за рахунок підвищення швидкодії та зменшення паняті виходячи зі специфіки задач, на рішення яких орієнтована вбудована система. Аналіз літературних джерел показує, що використання загальних підходів до оптимізації вбудованого програмного забезпечення для реалізації алгоритмів ЦОС не

дає, як правило, належного ефекту. Тому доцільно використовувати спеціалізовані методи оптимізації програмно-апаратного забезпечення SoC, орієнтовані на вирішення конкретної задачі реалізації алгоритмів ЦОС Одним з розповсюджених алгоритмів ЦОС є швидке перетворення Фур'є (Fast Fourier Transform, FFT) яке ефективно обчислює дискретне перетворення Фур'є. Розмір FFT (або довжина FFT) визначає, скільки дискретних точок використовується в перетворенні, що впливає на частотну роздільну здатність, часову роздільну здатність і обчислювальну складність. Виходячи з цього метою дослідження є аналіз та порівняння часових та апаратурних витрат на реалізацію алгоритму FFT для різних варіантів його довжини на PS частині SoC з використанням спеціалізованих бібліотек, та реалізація цього алгоритму на PL частині SoC з використанням IP-core з репозиторію Xilinx.

Математична модель алгоритму FFT.

Розрахунок дійсного FFT є можливим з використанням двох підходів. У першому підході використовується реалізація Complex FFT, де мнима частина комплексного комплексного числа дорівнює нулю і відліки сигналу присвоюються тільки у дійсну частину комплексних вхідних даних. У другому варіанті є можливість розрахувати дійсне FFT використовуючи однорівневу децимацію у часовій області використовуючи комплексне швидке перетворення Фур'є з половинної довжини.

Будемо використовувати малий регістр та індекс n для послідовностей у часовій області, тоді як відповідна літера верхнього регістру разом з індексом k використовується для відповідної послідовності частотної області.

У загальному вигляді вивід розрахунку дійсного перетворення Фур'є використовуючи комплексне перетворення виглядає наступним чином:

$$X[k]=\sum_{n=0}^{N-1}x[n]e^{-j2\pi\frac{kn}{N}}=$$

$$=\sum_{n1=0}^{N/2-1}x[2n_1]e^{-j2\pi\frac{k2n_1}{N}}+\sum_{n2=0}^{N/2-1}x[2n_2+1]e^{-j2\pi\frac{k(2n_2+1)}{N}}=$$

$$=\sum_{n1=0}^{N/2-1}x[2n_1]e^{-j2\pi\frac{k2n_1}{N}}+e^{-j2\pi\frac{kn}{N}}\cdot\sum_{n2=0}^{N/2-1}x[2n_2+1]e^{-j2\pi\frac{kn_2}{N/2}}=$$

$$=X_e[k]+X_0[k]e^{-j2\pi\frac{kn}{N}}.$$

Після чого дійсне перетворення Фур'є може бути розраховано як: $z[n]=X_e[n]+jX_0[n]$.

Вхідні значення відліків сигналу представляються як пари дійсних та мнимих компонент комплексного числа.

Виконуємо реалізацію комплексного швидкого перетворення Фур'є для розрахунку $X_e[k]$ та $X_0[k]$ $Z[n]=DFT_k^{N/2}\{z\}$ розраховується $X_e[k]$ та $X_0[k]$ $X_e[n]=\frac{Z[k]+Z\cdot[N/2-k]}{2}$, $X_k[n]=-j\frac{Z[k]+Z\cdot[N/2-k]}{2}$.

У результаті $X[k]$, $k=0....N-1$ тепер можна розрахувати як

$$X[k]=X_e[k\bmod(N/2)+X_o[k\bmod(N)/2]e^{-j2\pi\frac{k}{N}}.$$

У типовому процесі проектування для платформи ZYNQ як програмної так і апаратної частин є доцільним використовувати інструментарій як Vitis HLS для синтезу IP блоків, так і Vivado для проектування системи. Написання програмного забезпечення як для PS частини так і для PL частини виконується у середовищі Vitis IDE на мові програмування C [2].

Для реалізації і профілювання швидкодії та споживання пам'яті використовується набір інструментів Vivado / Vitis. З метою визначення часу виконання кожного з алгоритмів було реалізовано шаблонну функцію яка використовує системний таймер з високою роздільною здатністю в архітектурі ZYNQ, після чого кожний алгоритм виконується задану кількість разів і виводить статистику по часу виконання з урахуванням мінімального, максимального, середнього часу виконання та стандартного

відхилення. З метою отримання даних по споживанню пам'яті внутрішніми структурами алгоритму було виконано модифікацію вихідного коду кожної зі спеціалізованих бібліотек. У реалізації перетворення з використанням Xilinx HLS IP ядра було використано AXI-DMA інтерфейс з метою коректної роботи з AXI-Stream шиною.

Результати дослідження продуктивності реалізації алгоритму FFT на платформі ZYNQ.

В ході проведених експериментів по реалізації алгоритму FFT на платформі SoC Xilinx ZYNQ-7000 були проаналізовані швидкодія (час реалізації алгоритму) та обсяг використаної оперативної пам'яті для різної довжини алгоритму FFT були розглянуті нативна реалізація FFT на мові програмування С для PS частини (ARM Cortex A9 MPCore), реалізація алгоритму FFT для PS частини з використанням спеціалізованих бібліотек (NE10, PFFFT, KISS_FFT), нативна реалізація для PL частини на мові програмування С з використанням інструментальних засобів САПР Vivado/Vitis та реалізація для PL частини з використанням стандартного IP-core Xilinx HLS FFT IP.

В табл. 1 наведені абсолютні показники швидкодії в мікросекундах (Time μs) та витрат оперативної пам'яті ОЗУ в байтах (Memory) при реалізації алгоритму FFT для його малих розмірів для спеціалізованих бібліотек PS частини SoC.

Табл. 1

Library Name	FFT Size 64 Time | Memory	FFT Size 128 Time | Memory	FFT Size 256 Time | Memory
NE10	1.69 | 1180	3.72 | 2076	7.37 | 3868
PFFFT	2.41 | 404	4.72 | 724	8.75 | 1364
KISS_FFT	5.31 | 916	9.02 | 1556	24.38 | 2836
HANDY	5.63 | 96	12.24| 192	27.19 | 384

Нормалізація даних виконувалась методом min/max, враховуючи абсолютний мінімальний та максимальний час виконання між спеціалізованими бібліотеками. Так само виконувалась нормалізація по використанню бібліотеками ОЗУ платформи ZYNQ.

$$y' = \frac{y - \min(y)}{\max(y) - \min(y)}$$

В табл. 2 наведені нормалізовані показники швидкодії FFT та витрат пам'яті для відповідних бібліотек та малих розмірів FFT, а на рис. 1 та 2 графічне зображення зазначених показників.

Табл. 2

Library Name	FFT Size 64 Time | Memory	FFT Size 128 Time | Memory	FFT Size 256 Time | Memory
NE10	0.036 | 0.296	0.111 | 0.531	0.249 | 1.000
PFFFT	0.067 | 0.093	0.153 | 0.177	0.305 | 0.345
KISS_FFT	0.166 | 0.227	0.301 | 0.395	0.865 | 0.729
HANDY	0.184 | 0.013	0.441 | 0.038	1.000 | 0.088

В табл. 3 наведені абсолютні показники швидкодії FFT в мікросекундах (Time μs) та витрат оперативної пам'яті ОЗУ в байтах (Memory) для середніх розмірів FFT.

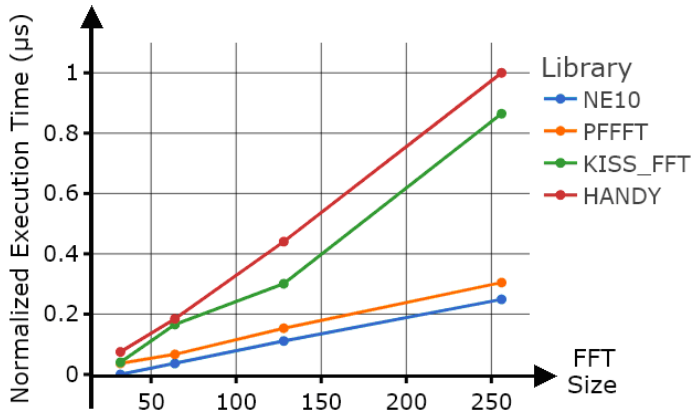

Рис. 1. Нормалізовані показники швидкодії PS частини SoC для малих розмірів FFT

ISSN 2079-0031 (Print) ISSN 2411-0558 (Online)

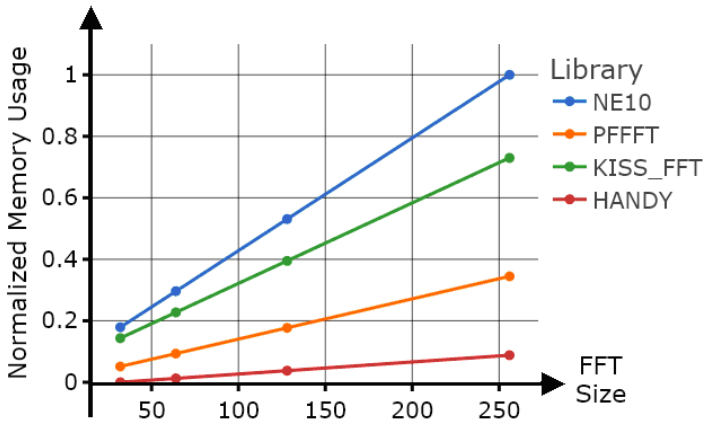

Рис. 2. Нормалізовані показники витрат пам’яті PS частини SoC для малих розмірів FFT

Табл. 3

Library Name	FFT Size 512 Time |Memory	FFT Size 1024 Time | Memory	FFT Size 2048 Time | Memory
NE10	17.00 | 7452	34.88 | 14620	92.53 | 28956
PFFFT	19.53 | 2644	39.87 | 5204	93.54 | 10324
KISS_FFT	42.60 | 5396	111.1 | 10516	202.7 | 20756
HANDY	59.61 | 768	130.5 | 1536	287.0 | 3072

В табл. 4 наведені нормалізовані показники швидкодії FFT та витрат пам’яті для відповідних бібліотек та середніх розмірів FFT, а на рис. 3 та 4 графічне зображення зазначених показників.

Табл. 4

Library Name	FFT Size 512 Time | Memory	FFT Size 1024 Time | Memory	FFT Size 2048 Time | Memory
NE10	0.000 | 0.237	0.067 | 0.491	0.276 | 1.000
PFFFT	0.010 | 0.067	0.085 | 0.157	0.277 | 0.339
KISS_FFT	0.092 | 0.164	0.341 | 0.346	0.673 | 0.709
HANDY	0.160 | 0.000	0.421 | 0.027	1.000 | 0.082

ISSN 2079-0031 (Print) ISSN 2411-0558 (Online)

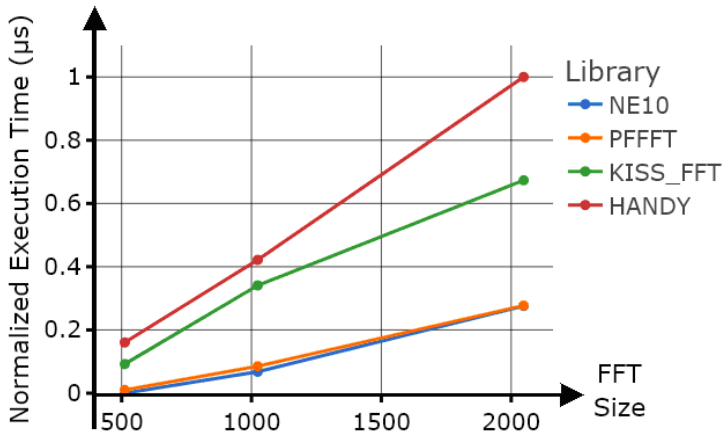

Рис. 3. Нормалізовані показники швидкодії PS частини SoC для середніх розмірів FFT

Співвідношення нормалізованих показників швидкодії та пам'яті для бібліотек PS частини SoC Xilinx ZYNQ-7000 показано на рис. 5. Наведене порівняння показує, що при вимогах високої швидкодії доцільно використовувати спеціалізовані бібліотеки PSБ а нативна реалізація може забезпечити економію оперативної памяті.

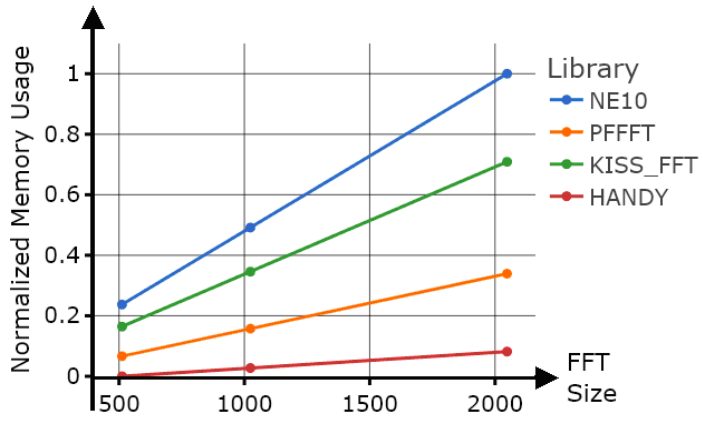

Рис. 4. Нормалізовані показники витрат пам'яті PS частини SoC для середніх розмірів FFT

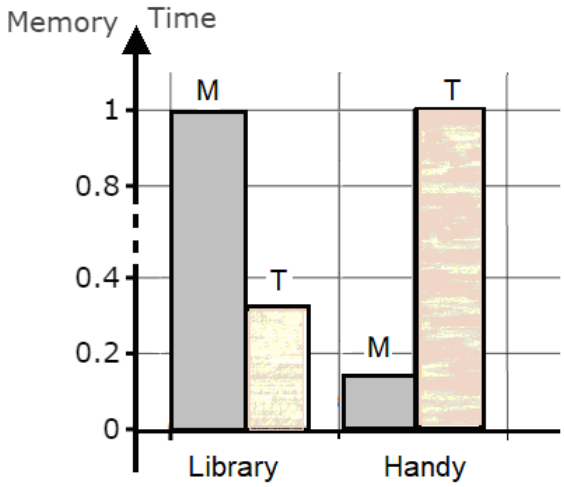

Рис. 5. Співвідношення швидкодії та апаратурних витрат для реалізації FFT на PS частині SoC

Проведемо аналіз витрат апаратних ресурсів (Hardware) для PL-частини SoC. Максимально доступні ресурси PL частини на платформі ZYNQ такі: LUT:53200, FF (Reg):106400, DSP:220, BRAM:140.

В табл. 5 наведені показники швидкодії та Hardware для одного з великих розмірів FFT (16384 відліків) при реалізації на PL частині. В таблиці наведена абсолютна кількість компонентів та процент від їх загальної кількості у SoC.

Табл. 5

PL FFT	FFT Size	Hardware			Time (μs)
		LUT	Reg	BRAM	
IP CORE Xilinx IP	16384	11974 23%	18652 18%	101 72%	527.022
Our HLS	16384	13153 25%	10682 10%	57 40%	9061.06

При цьому середні апаратурні витрати для IP CORE Xilinx IP складають 38%, а для власної реалізації FFT – 25% від наявних апаратних ресурсів.

Співвідношення нормалізованих показників швидкодії та абсолютних показників апаратних витрат для PL частини SoC Xilinx ZYNQ-7000 показано на рис. 6.

ISSN 2079-0031 (Print) ISSN 2411-0558 (Online)

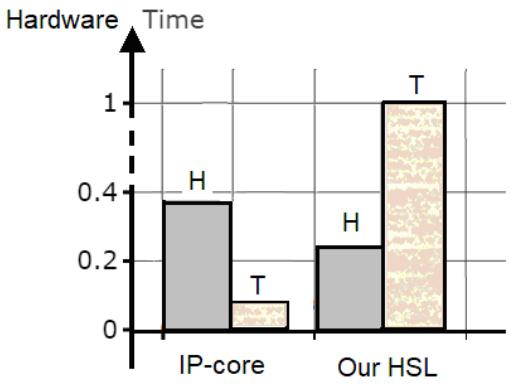

Рис. 6. Співвідношення швидкодії та апаратурних витрат для реалізації FFT на PL частині SoC

Як висновок можна зауважити, що власна реалізація алгоритму FFT для PL частини SoC Xilinx ZYNQ-7000 значно поступається швидкодією IP CORE Xilinx IP, що обумовлено орієнтацією реалізації IP CORE Xilinx IP саме на апаратуру PL частини цієї SoC.

Наприкінці наведемо графік співвідношення абсолютних значень швидкодії алгоритму FFT для різних способів його програмно-апаратної реалізації на платформі SoC Xilinx ZYNQ-7000 (рис. 7).

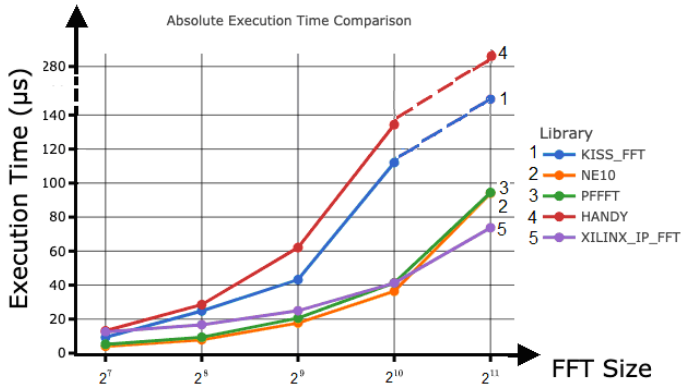

Рис. 7. Співвідношення швидкодії різних способів реалізації алгоритму FFT SoC Xilinx ZYNQ-7000

З наведеного графіку видно, що для малих розмірів FFT (до 2048 відліків) використання спеціалізованих бібліотек PS частини показує кращу швидкодію, а для великих розмірів FFT реалізація на PL частині з

використанням IP-core Xilinx показала найкращу швидкодію при використанні менше 50% власних апаратних ресурсів.

Висновки.

В ході проведених досліджень були проаналізовані швидкодія (час реалізації алгоритму) та обсяг використаної оперативної пам'яті для різної довжини алгоритму швидкого перетворення Фур'є при його реалізації на технологічній платформі SoC Xilinx ZYNQ-7000. Були розглянуті нативна реалізація FFT на мові програмування С для PS частини (ARM Cortex A9 MPCore), реалізація для PS частини з використанням спеціалізованих бібліотек (NE10, PFFFT, KISS_FFT), нативна реалізація для PL частини на мові програмування С з використанням інструментальних засобів САПР Vivado/Vitis та реалізація для PL частини з використанням стандартного IP-core Xilinx HLS FFT IP.

Для PS частини найкращу швидкодію показало використання бібліотеки NE10, але ця бібліотека потребує найбільше оперативної пам'яті. Нативна реалізація в PS частині потребувала найменших витрат оперативної пам'яті, але програє по швидкодії. Для PL частини найкращі показники по швидкодії (серед усіх варіантів реалізацій) показало використання IP-core Xilinx HLS FFT IP. Відносно розміру FFT витрати пам'яті та часу для PS частини зростали приблизно лінійно. Відносно порівняння швидкодії для PS та PL частин, то на малих і середніх розмірах FFT (до 2048 відліків включно) у реалізації на PS частині швидкодія була вище, а для великих розмірів FFT реалізація на PL частині з використанням IP-core Xilinx показала найкращу швидкодію.

Наукова новизна результатів полягає у тому. що набув подальшого розвитку метод оптимального розподілу обчислень між компонентами SoC при програмно-апаратній реалізації алгоритмів цифрової обробки сигналів, який враховує параметри апаратних компонентів SoC та спеціалізованих програмних бібліотек, що дозволило задовольнити вимоги користувачів до швидкодії та обмежень на витрати оперативної пам'яті при реалізації алгоритму швидкого перетворення Фур'є.

Практична значимість дослідження полягає у напрацюванні рекомендацій проектувальникам програмно-апаратної реалізації алгоритму FFT на платформі ZYNQ Zedoard з використанням

інструментальних засобів Vivado/Vitis, що дозволило оптимальним чином розподілити обчислення між програмною та апаратною частинами SoC в залежності від обмежень на час виконання алгоритму FFT, використану при цьому пам'ять та точність обчислень. Також було показано, що оптимальним з точки зору співвідношення швидкодії, точності та використаної вбудованої оперативної пам'яті серед різних способів реалізації алгоритму FFT є використання стандартного IP-core Xilinx HLS FFT IP для PL частини SoC Xilinx ZYNQ-7000.

Напрямок подальших досліджень полягає в розробці метрик оцінювання цільового алгоритму для оптимального розподілу обчислень між PS та PL частинами SoC Xilinx ZYNQ-7000 при сумісному програмно-апаратному проектуванні алгоритмів цифрової обробки сигналів для вбудованих систем.

Список літератури

1. Laurence Tratt. Four Kinds of Optimisation [Electronic resource] / Tratt.net. – Mode of access: https://tratt.net/laurie/blog/2023/four_kinds_of_optimisation.html – Date of access: 10.02.2025.

2. Шкіль О. Автоматизоване проєктування вбудованих систем цифрового оброблення сигналів на платформі SoC / О. Шкіль, Д. Рахліс, І. Філіпенко, В. Корнієнко, Т. Рожнова // Сучасний стан наукових досліджень та технологій в промисловості. – 2024. – № 1 (27). – С. 72-83.
DOI: https://doi.org/10.30837/ITSSI.2024.27.192

3. Automatic Generation of High-Performance FFT Kernels on Arm and X86 CPUs / Z. Li et al. // IEEE Transactions on Parallel and Distributed Systems. – 2020. – Vol.31, no.8. – P. 1925–1941. URL: https://doi.org/10.1109/tpds.2020.2977629 (date of access: 12.01.2025).

4. High-throughput FFT architectures using HLS tools / H. Almorin et al. // Proceedings of 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, United Kingdom, 24–26 October 2022. 2022. URL: https://doi.org/10.1109/icecs202256217.2022.9970886 (date of access: 12.01.2025).

5. Transformations of High-Level Synthesis Codes for High-Performance Computing / J. de Fine Licht et al. IEEE Transactions on Parallel and Distributed Systems. 2021. Vol. 32, no. 5. P. 1014–1029. URL: https://doi.org/10.1109/tpds.2020.3039409 (date of access: 04.04.2025).

6. A Cost-efficient Hardware Accelerator Design for 2D Sliding Discrete Fourier Transform / W.-H. Juang et al. // 2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), PingTung, Taiwan, 17–19 July 2023. 2023. URL: https://doi.org/10.1109/icce-taiwan58799.2023.10227037 (date of access: 12.01.2025).

7. Malathi L., Bharathi A., Jayanthi A. N. FPGA design of FFT based intelligent accelerator with optimized Wallace tree multiplier for image super resolution and quality enhancement. // Biomedical Signal Processing and Control. – 2024. – Vol. 88. – Article 105599. URL: https://doi.org/10.1016/j.bspc.2023.105599 (date of access: 22.03.2025).

8. Li Y., Chen H., Xie Y. An FPGA-Based Four-Channel 128k-Point FFT Processor Suitable for Spaceborne SAR // Electronics. – 2021. – Vol. 10, no. 7. – P. 816. URL: https://doi.org/10.3390/electronics10070816 (date of access: 22.03.2025).

9. Barth C., Cansio E. SoC-Based Multichannel STFT Generator for Digital Electronic Warfare Receivers. // 2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC), Concepcion, Chile, 23–26 October 2023. 2023. – P. 1-6. URL: https://doi.org/10.1109/sccc59417.2023.10315735 (date of access: 22.03.2025).

10. OpenFFT: An Adaptive Tuning Framework for 3D FFT on ARM Multicore CPUs / T. Chen et al. ICS '23: 37th International Conference on Supercomputing, Orlando FL USA. New York, NY, USA, 2023. URL: https://doi.org/10.1145/3577193.3593735 (date of access: 29.03.2025).

11. Evaluation of Open-Source Linear Algebra Libraries targeting ARM and RISC-V Architectures / C. Fibich et al. 2020 Federated Conference on Computer Science and Information Systems, 6 – 9 September 2020. 2020. URL: https://doi.org/10.15439/2020f145 (date of access: 29.03.2025).

References.

1. Laurence Tratt. Four Kinds of Optimisation [Electronic resource] / Tratt.net. – Mode of access: https://tratt.net/laurie/blog/2023/four_kinds_of_optimisation.html – Date of access: 10.02.2025.

2. Shkil′ O. Avtomatyzovane proyektuvannya vbudovanykh system tsyfrovoho obroblennya syhnaliv na platformi SoC / O. Shkil′, D. Rakhlis, I. Filipenko, V. Korniyenko, T. Rozhnova // Suchasnyy stan naukovykh doslidzhen′ ta tekhnolohiy v promyslovosti. – 2024. – № 1. DOI: https://doi.org/10.30837/ITSSI.2024.27.192.

3. Automatic Generation of High-Performance FFT Kernels on Arm and X86 CPUs / Z. Li et al. // IEEE Transactions on Parallel and Distributed Systems. – 2020. – Vol.31, no.8. – P. 1925–1941. URL: https://doi.org/10.1109/tpds.2020.2977629 (date of access: 12.01.2025).

4. High-throughput FFT architectures using HLS tools / H. Almorin et al. // Proceedings of 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, United Kingdom, 24–26 October 2022. 2022. URL: https://doi.org/10.1109/icecs202256217.2022.9970886 (date of access: 12.01.2025).

5. Transformations of High-Level Synthesis Codes for High-Performance Computing / J. de Fine Licht et al. IEEE Transactions on Parallel and Distributed Systems. 2021. Vol. 32, no. 5. P. 1014–1029. URL: https://doi.org/10.1109/tpds.2020.3039409 (date of access: 04.04.2025).

6. A Cost-efficient Hardware Accelerator Design for 2D Sliding Discrete Fourier Transform / W.-H. Juang et al. // 2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), PingTung, Taiwan, 17–19 July 2023. 2023. URL: https://doi.org/10.1109/icce-taiwan58799.2023.10227037 (date of access: 12.01.2025).

7. Malathi L., Bharathi A., Jayanthi A. N. FPGA design of FFT based intelligent accelerator with optimized Wallace tree multiplier for image super resolution and quality enhancement. // Biomedical Signal Processing and Control. – 2024. – Vol. 88. – Article 105599. URL: https://doi.org/10.1016/j.bspc.2023.105599 (date of access: 22.03.2025).

8. Li Y., Chen H., Xie Y. An FPGA-Based Four-Channel 128k-Point FFT Processor Suitable for Spaceborne SAR // Electronics. – 2021. – Vol. 10, no. 7. – P. 816. URL: https://doi.org/10.3390/electronics10070816 (date of access: 22.03.2025).

9. Barth C., Cansio E. SoC-Based Multichannel STFT Generator for Digital Electronic Warfare Receivers. // 2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC), Concepcion, Chile, 23–26 October 2023. 2023. – P. 1-6. URL: https://doi.org/10.1109/sccc59417.2023.10315735 (date of access: 22.03.2025).

10. OpenFFT: An Adaptive Tuning Framework for 3D FFT on ARM Multicore CPUs / T. Chen et al. ICS '23: 37th International Conference on Supercomputing, Orlando FL USA. New York, NY, USA, 2023. URL: https://doi.org/10.1145/3577193.3593735 (date of access: 29.03.2025).

11. Evaluation of Open-Source Linear Algebra Libraries targeting ARM and RISC-V Architectures / C. Fibich et al. 2020 Federated Conference on Computer Science and Information Systems, 6 – 9 September 2020. 2020. URL: https://doi.org/10.15439/2020f145 (date of access: 29.03.2025).

Представив д-р техн. наук, проф., професор кафедри автоматизації проектування обчислювальної техники Харківський Національний Університет Радіоелектроніки Кривуля Генадій Федорович.

Надійшла (received) 12.04.2025

УДК 004.3

Аналіз ефективності використання спеціалізованих обчислювальних модулів та бібліотек при реалізації алгоритмів цифрової обробки сигналів на платформі SoC / О.С. Шкіль, І.В. Філіпенко, А.М. Мірошник, В.Р. Корнієнко // Вісник НТУ "ХПІ". Серія: Інформатика та моделювання. – Харків: НТУ "ХПІ". – 2025. – № 1 (13). – С. 112 – 128.

Розглянута реалізація алгоритму швидкого перетворення Фур'є (FFT) на технологічній платформі системи на кристалі SoC Xilinx ZYNQ-7000. Виконано порівняння часових та апаратурних витрат на реалізацію алгоритму FFT для різних варіантів його довжини на PS частині SoC з використанням спеціалізованих бібліотек, та реалізація цього алгоритму на PL частині SoC з використанням IP-core з репозиторію Xilinx. Реалізація алгоритму FFT виконана на мові програмування С з використанням інструментів Vivado/Vitis. Найкращу швидкодію серед різних способів реалізації алгоритму FFT показало використання стандартного IP-core Xilinx HLS FFT IP для PL частини SoC Xilinx ZYNQ-7000. Іл. 7, Бібліогр.: 11 назв.

Ключові слова: вбудовані системи, системи на кристалі, вбудоване програмне забезпечення, швидке перетворення Фур'є, мова програмування С, швидкодія, пам'ять, оптимізація, спеціалізовані бібліотеки, програмована логіка, IP-core

УДК 004.3

Analysis of the effectiveness of using specialized computing modules and libraries when implementing digital signal processing algorithms on the platform SoC / O.S. Shkil, I.V. Filipenko, A.M. Miroshnyk, V.R. Kornienko // Herald of the National Technical University "KhPI". Series of "Informatics and Modeling". – Kharkov: NTU "KhPI". – 2025. – № 1 (13). – P. 112 – 128.

The implementation of the Fast Fourier Transform (FFT) algorithm on the Xilinx ZYNQ-7000 SoC system-on-chip technology platform is considered. The time and hardware costs for implementing the FFT algorithm for different options for its length on the PS part of the SoC using specialized libraries and the implementation of this algorithm on the PL part of the SoC using the IP-core from the Xilinx repository are compared. The implementation of the FFT algorithm is performed in the C programming language using the Vivado/Vitis tools. The best performance among different ways to implement the FFT algorithm was shown by using the standard Xilinx HLS FFT IP IP for the PL part of the SoC Xilinx ZYNQ-7000. Fig.: 6, Refs.: 13 titeles.

Keywords: embedded systems, systems on a chip, embedded software, fast Fourier transform, C programming language, performance, memory, optimization, specialized libraries, programmable logic, IP-core