DOI 10.31891/2307-5732-2023-319-1-360-368
УДК 004.93

ЛИСЕНКО СЕРГІЙ
Хмельницький національний університет
https://orcid.org/0000-0001-7243-8747
АТАМАНЮК ОЛЬГА
https://orcid.org/0009-0003-7937-3677
БОХОНЬКО ОЛЕКСАНДР
Хмельницький національний університет

# МЕТОД ПОБУДОВИ АПАРАТНОЇ АРХІТЕКТУРИ ДЛЯ СИСТЕМА КОМП'ЮТЕРНОГО ЗОРУ НА ОСНОВІ FPGA

*У цій роботі розглянуто методологію розробки апаратної архітектури для систем комп'ютерного зору на основі програмованої логіки, зокрема FPGA. У роботі проведено дослідження методів розробки архітектур для комп'ютерного зору та встановлено переваги використання FPGA у порівнянні з традиційними процесорами загального призначення. У роботі також розглянуто основні аспекти проектування апаратури на FPGA, зокрема вибір відповідного засобу розробки, проектування логіки, синтез та валідація розробленої апаратури. Досліджено можливості FPGA у забезпеченні високої продуктивності та ефективності систем комп'ютерного зору, що робить їх привабливими для використання у візуальних системах.*

*Ключові слова: FPGA, апаратна архітектура, системи комп'ютерного зору, алгоритми обробки зображень програмовані логічні пристрої, платформа для обробки візуальної інформації, низькорівневе програмування, платформа для дослідження алгоритмів зору, високоякісне відео та зображення з використанням FPGA.lexander*

LYSENKO SERGII,, ATAMANIUK OLGA, BOKHONKO OLEXANDER
Khmelnytskyi National University

## METHOD OF CONSTRUCTING HARDWARE ARCHITECTURE FOR COMPUTER VISION SYSTEM BASED ON FPGA

*In this work, the methodology of hardware architecture development for computer vision systems based on programmable logic, in particular FPGA, is considered.*

*In the work, the methods of developing architectures for computer vision are studied and the advantages of using FPGA compared to traditional general-purpose processors are established. The paper also considers the main aspects of hardware design on FPGA, in particular, the selection of a suitable development tool, logic design, synthesis, and validation of the developed hardware. The capabilities of FPGAs in providing high performance and efficiency of computer vision systems have been investigated, which makes them attractive and popular for use in visual systems. We discover that FPGA technology offers a high degree of flexibility and configurability, allowing for the creation of custom hardware architectures that can be tailored to specific computer vision applications.*

*In addition, to studying the benefits of using FPGA technology for computer vision, we also consider the main aspects of hardware design on FPGA, including the selection of a suitable development tool, logic design, synthesis, and validation of the developed hardware. By carefully considering these aspects, we can ensure that the hardware architecture we develop is both efficient and effective. Our research shows that the capabilities of FPGAs in providing high performance and efficiency of computer vision systems are truly remarkable. This makes them an attractive choice for use in visual systems, particularly in scenarios where high speed and accuracy are critical. Overall, our work serves to shed light on the many benefits of FPGA technology for computer vision and lays the foundation for further research and development in this exciting field.*

*Keywords: FPGA, hardware architecture, computer vision systems, image processing algorithms, programmable logic devices, visual information processing platform, low-level programming, vision algorithm research platform, high-quality video and imaging using FPGA.*

## Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Програмовані вентильні матриці (FPGA) — це нове доповнення до світу прискорення центрів обробки даних. Хоча базова технологія існує десятиліттями, її застосування в центрах обробки даних поступово починає набирати обертів. Однак існує безліч проблем, які перешкоджають широкому застосуванню FPGA в центрах обробки даних. Ланцюжки інструментів із закритим вихідним кодом призводять до блокування постачальника та нестабільних потоків інструментів. Мови, що використовуються для програмування FPGA, вимагають різних процесів проектування, які нелегко освоїти розробникам програмного забезпечення. Порівняно з стандартними рішеннями, що використовують центральні та графічні процесори, FPGA є дорожчими та потребують більше часу для розробки. Усе це та багато іншого робить FPGA важко продати людям, які потребують прискорення завдань.

Тим не менш, FPGA також пропонують можливість розробки швидших прискорювачів з меншою енергетичною оболонкою для швидко мінливих програм [1].

FPGA також можна використовувати для підвищення ефективності мережі в центрі обробки даних шляхом заміни компонентів центральної мережі розумними комутаторами. Робота, представлена тут, досягає 7-кратного прискорення порівняно з класичною реалізацією розподіленого програмного забезпечення за сценарієм хеш-з'єднання. Крім того, FPGA можна використовувати для впровадження

нових технологій зберігання в центр обробки даних шляхом надання високоефективних консенсусних послуг безпосередньо в мережі.

Коли справа доходить до продуктивності, вона виглядає незадовільно. У світі програмного забезпечення існує багато добре налаштованих систем трасування, щоб з'ясувати, де саме втрачається продуктивність.

Такі методи, як лічильники продуктивності, пропонують певний спосіб визначити, чи щось не так. Однак зазвичай це лише дуже розпливчасті інструменти, які можуть лише надавати підказки, але не давати певності. Проблеми продуктивності, на які вказують лічильники продуктивності, потребують ще одного тривалого моделювання, потоку бітів, повторного процесу, щоб привести до результатів.

Відповідно, перехід до нового покоління часто є дуже повільним і схильним до помилок процесом. Усі зміни необхідно перевірити ще раз за допомогою описаного вище процесу. Зрештою, перехід від одного покоління FPGA до іншого часто є дуже дорогим [2].

Окрім проблем, пов'язаних із налаштуванням, також можуть виникнути нові функції, які потрібно використовувати для оптимальної продуктивності. У світі програмного забезпечення ці функції зазвичай автоматично використовуються компілятором. З іншого боку, для FPGA нові технології можуть вимагати глибоких змін будь-якої частини конструкції, що знову запускає вищезгаданий цикл повторного синтезу.

Останні роки показали, що існує багато програм, які не так легко відображаються на ЦП або ГП. Навіть такі проблеми, які добре підходять для графічних процесорів, як нейронна мережа, FPGA пропонують значну економію електроенергії та підвищену пропускну здатність. Такі методи, як відтворення з точністю до одного біта, легко виконати на FPGA, але неможливо на традиційних обчислювальних пристроях.

**Відомі методи побудови апаратної архітектури для систем комп'ютерного зору**

Відомі методи побудови апаратної архітектури для систем комп'ютерного зору є однією з ключових галузей розвитку комп'ютерних технологій. Ці методи дозволяють підвищити точність розпізнавання об'єктів, зменшити час обробки даних та забезпечити надійність роботи системи комп'ютерного зору.

Одним з відомих методів побудови апаратної архітектури для систем комп'ютерного зору є використання програмованих логічних пристроїв (FPGA). FPGA можуть бути сконфігуровані та програмовані, щоб створювати апаратні архітектури, спеціально призначені для обробки зображень. Ця методика забезпечує швидке виконання завдань, пов'язаних з обробкою зображень, та зменшення завантаження на центральний процесор.

Іншим відомим методом побудови апаратної архітектури є використання графічних процесорів (GPU). Ці процесори розроблені для обробки графіки, але також можуть бути використані для обробки зображень. Графічні процесори мають велику кількість ядер, які можуть працювати паралельно, що забезпечує швидку обробку зображень та велику продуктивність.

Крім того, існують методи побудови апаратної архітектури на основі ASIC (застосуванням спеціалізованих інтегральних схем). Ці інтегральні схеми можуть бути розроблені спеціально для конкретного завдання з обробки зображень, що забезпечує найвищу продуктивність та швидкість роботи системи [3].

Усі ці методи побудови апаратної архітектури для систем комп'ютерного зору мають свої переваги та недоліки, і ви вибираєте той метод, який найбільше відповідає вашим потребам та вимогам.

Наприклад, FPGA є відмінним вибором для систем комп'ютерного зору, які потребують високої продуктивності та надійності роботи. FPGA можуть бути програмовані для виконання специфічних завдань, що забезпечує ефективну обробку зображень та високу швидкість роботи системи.

Графічні процесори також є відмінним вибором для систем комп'ютерного зору, особливо для завдань, які вимагають паралельної обробки зображень. GPU мають велику кількість ядер та високу продуктивність, що дозволяє їм швидко виконувати складні завдання з обробки зображень.

ASIC, з іншого боку, забезпечують найвищу продуктивність та швидкість роботи системи, оскільки вони спеціально розроблені для конкретного завдання. Однак, розробка та виготовлення ASIC можуть бути дорогими та часовими затратами [4].

Таким чином, вибір методу побудови апаратної архітектури для систем комп'ютерного зору залежить від конкретних потреб та вимог системи.

**Метод побудови апаратної архітектури для систем комп'ютерного зору на основі FPGA**

Дозволити комп'ютерам сприймати навколишнє середовище все ще залишається одним із найскладніших завдань комп'ютерного зору. Особливо стереозір, сприйняття глибини за допомогою двох камер, важливий для багатьох сфер, таких як робототехніка та автономне водіння. Стереозйомка використовує дві камери, розташовані на деякій відстані одна від одної по горизонталі, але на одному рівні по вертикалі. Таким чином, пікселі на зображеннях, знятих двома камерами, зміщуються лише в горизонтальному напрямку, причому максимальне зміщення пікселя (традиційно називається невідповідністю) обмежується відстанню між двома камерами.

Потім обчислену невідповідність для пікселя можна використовувати для отримання інформації про глибину зі стереозображень за допомогою тріангуляції (пікселі, розташовані ближче до камер, мають більшу невідповідність) [5].

Алгоритм у центрі цієї роботи називається Semi-Global Block Matching (SGBM), який є одним із найшвидших алгоритмів, який також показує високу точність у тестах стереозору. Основою запропонованого методу є синтез нової апаратну архітектуру програмно-технічного засобу для комп'ютерного зору, що застосовує FPGA, а також використовує алгоритм SGBM.

Метод забезпечує синтез побудови параметризованої архітектури для систем комп'ютерного зору на основі FPGA архітектура, яка має високу масштабованість, що дозволяє легко впроваджувати її на малих малопотужних пристроях (наприклад, в автономних роботах), а також на великих високопродуктивних чіпах (наприклад, у стаціонарних випадках використання для обробки кількох відеопотоків високої роздільної здатності). Як показано на рисунку 1, основна увага приділяється обчисленню невідповідності.

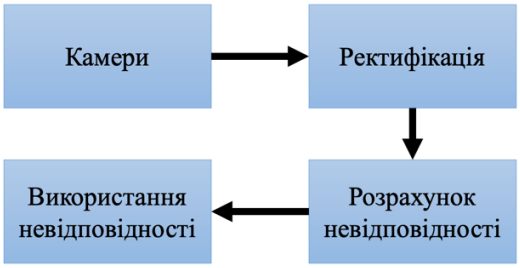



**Рис. 1. Узагальнена схема програмно-апаратного засобу побудови апаратної архітектури для систем комп'ютерного зору на основі FPGA. Типова система стереозору**

Метод побудови апаратної архітектури для систем комп'ютерного зору на основі FPGA використовує алгоритм SGBM. Він забезпечує хорошу точність при керованих обчислювальних зусиллях і надійний щодо вибору параметрів конфігурації.

Ключовою відмінністю від простих підходів на основі блоків, які зазвичай обчислюють збіги, прагнучи до мінімальних сум різниць між інтенсивністю пікселів у базовому та відповідних зображеннях, є використання більш складної функції вартості. Ця функція вартості не тільки має розширений діапазон (враховуючи не лише окремі пікселі чи локальні околиці, але й пікселі вздовж шляхів по всьому зображенню), але також розглядає характеристики вищого рівня (наприклад, взаємну інформацію та перепис), а не інтенсивність пікселів [6].

Перевага використання цієї характеристики, яку ввели Віола та Уеллс, полягає в її якості навіть у випадку невиправлених даних зображення. Однак він погано масштабується для зображень з більшою глибиною (більші координати S, особливо за глибинами, представленими як 8-бітні значення).

Використовується різновид цієї ідеї, яка розглядає різницю між прямими кількостями пікселів, що задовольняють відношення, а не відстань Хеммінга як вартість. Цей підхід, який називається непараметричним ранговим перетворенням, має якість відповідності, подібну до Census, але його легше реалізувати для високої продуктивності обчислень.

Функція $\mathrm{C}(\boldsymbol{p},\mathrm{d}) \in N_0$ використовується для позначення вартості відповідності пікселя $\boldsymbol{p} = (x,y)$ у координатах $(x,y)$ у базовому зображенні за припущеної невідповідності (зміщення) $d$ у координатах $(x - d,y)$ на відповідному зображенні.

Відмінності в кількості пікселів, темніших за центральний піксель (параметричне рангове перетворення), можна використовувати для реалізації $\mathrm{C}$.

Щоб визначити фактичну найкращу відповідність, ці витрати розраховуються для всіх потенційних розбіжностей $d < D_{max}$, де $D_{max}$ є верхньою межею потенційної невідповідності (через фізичну відстань встановлення двох камер) [7].

У першому наближенні передбачається, що відповідність із найнижчою вартістю вказує на справжню невідповідність $arg\ min_{d<D_{max}}\ C(\boldsymbol{p},d)$ між базовим і відповідним зображеннями для окремого пікселя $\boldsymbol{p}$.

Щоб досягти кращої точності відповідності, (напів)глобальні підходи, такі як SGBM, обчислюють ці витрати на потенційні відповідності не лише між окремими (або околицями) пікселів, але й уздовж багатопіксельних шляхів, що простягаються через усе зображення.

Вартість узгодження вздовж усього шляху, описана відносним зсувом елементів шляху $r = (\Delta x, \Delta y)$, для припущеної невідповідності $d$ позначається як $L_r(\boldsymbol{p},d)$.

Ці шляхи рівномірно розподіляються для загального перегляду збігів:

$$Lr'(\boldsymbol{p},d) = C(\boldsymbol{p},d) + min \begin{cases} Lr'(\boldsymbol{p} - \boldsymbol{r},d) \\ Lr'(\boldsymbol{p} - \boldsymbol{r},d-1) + P_1 \\ Lr'(\boldsymbol{p} - \boldsymbol{r},d+1) + P_1 \\ min_{\,i}\, Lr'(\boldsymbol{p} - \boldsymbol{r},i) + P_2 \end{cases} \quad (1)$$

Кількість і розташування шляхів має прямий вплив не тільки на точність узгодження, але також на обчислювальні зусилля і, в цьому випадку, на фактичну архітектуру апаратного прискорювача SGBM.

Метою є компроміс між продуктивністю та точністю. Скорочення від восьми до чотирьох шляхів $0°\,(r = (1{,}0)),\ 45°\,(r = (1{,}1)), 90°\,(r = (0{,}1))$ і $135°\,(r = (-1{,}1))$ призводить до втрати точності лише на 1,7% (збільшення кількості неправильно позначених розбіжностей) у тесті Middlebury, але дозволяє високоефективну апаратну архітектуру обчислень $L_r$ для всіх цих шляхів паралельно [8].

Якщо навіть обмежена втрата точності є неприйнятною, запропоновану апаратну архітектуру можна використати для виконання другого проходу по зображенню, обчислюючи решту шляхів $180°\,(r = (-1{,}0)), 225°\,(r = (-1,-1)), 270°\,(r = (0,-1))$ і $315°\,(r = (1,-1))$, починаючи з протилежного кута. Це зменшить частоту кадрів для відповідності вдвічі. Оскільки шляхи у вибраному розташуванні більше не розподіляються рівномірно по зображенню, у деяких тестах можна виміряти деякі незначні (неізотропні) ефекти згладжування, але вони не повинні впливати на зручність використання в сценаріях реального світу.

Необроблена вартість $\boldsymbol{Lr'}\,(\boldsymbol{p}, d)$ для узгодження пікселів $\boldsymbol{p}$ уздовж шляху $r$ для припущеної невідповідності $d$ обчислюється за формулою. Ці необроблені витрати на шлях розраховуються для всіх вибраних шляхів, для всіх потенційних розбіжностей $d$ до обмеження $D_{max}$. Для кожного пікселя оцінюється як локальна вартість $C$, так і напівглобальний компонент. Остання враховує чотири характеристики, які спостерігаються в реальних зображеннях, мінімальна з яких додається до локальної вартості: перша характеристика – це вартість попереднього пікселя на шляху, другий і третій компоненти штрафують за невелику розбіжність. Зміни $|\Delta d| = 1$ за $P_1$, тоді як останній член штрафує більші зміни невідповідності (так звані розриви) за $P_2$. $P_1$ зазвичай визначається в автономному режимі експериментально шляхом аналізу вхідних зображень, типових для реального випадку використання стереозору. $P_2$, з іншого боку, динамічно коригується під час виконання: оскільки невідповідність часто також представляє розриви, коли інтенсивність пікселя змінюється, обчислення $P_2 = \frac{P_2'}{|\mathrm{I}_p - \mathrm{I}_{p-r}|}'$ компенсує різні інтенсивності пікселів $\mathrm{I}_p$ та $\mathrm{I}_{p-r}$ уздовж шляху $r$. Що стосується $P_1$, $P_2'$ є константою, визначеною експериментально на основі репрезентативних зразків зображень офлайн. Для подальшого обговорення розрахунку вартості шляху зверніться до оригінальної роботи Гіршмюллера. Щоб визначити (напів) глобальну вартість узгодження, вартість шляху підсумовується по всіх шляхах. Однак для апаратної реалізації варто розглянути дещо змінене формулювання [9].

В апаратному забезпеченні ключовою характеристикою є ширина слова (у бітах) арифметичних операторів і типів даних. Оскільки шляхи проходитимуть по всьому зображенню, вони можуть бути досить довгими (залежно від роздільної здатності камери), і підсумовування їх вартості може призвести до великих значень, які потребують широких слів для обчислення та зберігання. Цьому можна протистояти, віднімаючи від необроблених витрат шляху для пікселя $\boldsymbol{Lr'}\,(\boldsymbol{p}, d)$ мінімальну вартість шляху для всіх припущених розбіжностей $d$ для попереднього пікселя $p - r$ уздовж шляху $\boldsymbol{r}$.

Ефект кодування лише відмінностей між попередніми та поточними пікселями призводить до зменшення величини значень, які вимагають відповідно вужчих слів даних для зберігання та обчислення.

Невідповідність $d$ з мінімальною відповідністю $cost\ arg\ min_d\ S(\boldsymbol{p}, d)$ вважається виграшною диспропорцією для пікселя $\boldsymbol{p}$. Ці виграшні відмінності виводяться прискорювачем для кожного пікселя як вхідні дані для подальшого обчислення фактичної глибини (положення осі $Z$, тут не обговорюється).

На практиці необхідні додаткові обмеження накладені для очищення викидів і позначення недійсних розбіжностей: результат аргументу $S(\boldsymbol{p}, d)$ може бути багатоелементним набором, що означає, що мінімальна вартість відповідності для пікселя $\boldsymbol{p}$ виникає для різних потенційних розбіжностей $d$. З такою неунікальною вартістю алгоритм не може визначити одну виграшну невідповідність, а натомість реєструє невідповідність для цього пікселя як «недійсний» [8].

Крім того, виконується так звана перевірка ліворуч/праворуч, яка порівнює результати алгоритму під час його виконання з поміняними ролями базового та відповідного зображень. Цю перевірку також можна ефективно здійснити (уникаючи перерахунку всіх розбіжностей для попереднього зображення збігу, яке зараз використовується як основа), повторно використовуючи попередньо обчислене $S(\boldsymbol{p}, d)$ уздовж епіполярної лінії як аргумент $S((x(\boldsymbol{p}) + d, y(\boldsymbol{p})), d)$, щоб вибрати виграшну невідповідність $d$ для другого зображення. Перевірка ліворуч/праворуч встановлює для невідповідності значення «недійсне», якщо відповідні невідповідності вихідного проходу та пропуску зі зміненими ролями відрізняються більш ніж на одиницю. Цей крок усуває фантомні диспропорції, що є результатом закритих поверхонь, які видно на одному зображенні, але приховані на іншому.

Запропонований метод побудови апаратної архітектури для систем комп'ютерного зору на основі FPGA уможливлює здійснити синтез архітектуру програмно-технічного засобу, яка підвищує швидкодію систем комп'ютерного зору за рахунок покращення не лише введення додаткового рівня детального паралелізму (наприклад, паралельне обчислення диспропорції та сортування), а й також завдяки реалізації з використанням найсучаснішого стилю розробки, нечутливого до затримок, у наступному – генерація мови опису обладнання. Як результат, він є значно більш масштабованим, легшим для розширення, але також

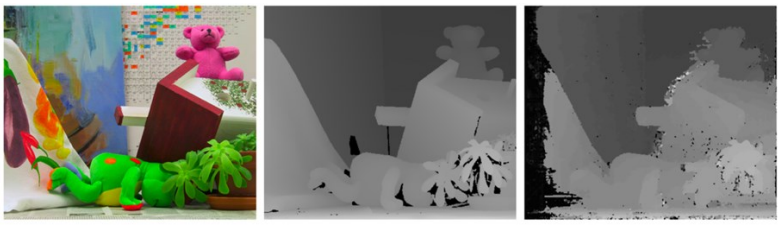
набагато швидшим, ніж оригінальна робота (навіть якщо компенсувати відмінності в цільових технологіях FPGA) [10].

У конструкції з декількома рядковими процесорами обчислені диспропорції буферизуються в FIFO, доки їх не витягне модуль виводу. Потім модуль виводу об'єднує виходи процесорів рядків, використовуючи ту саму циклічну схему, що й модуль введення. Потім медіанний фільтр 3 × 3 застосовується до об'єднаного потоку для видалення викидів у обчислених диспропорціях.

Максимальне значення будь-якого $L$ завжди менше $C_{max} + P_2$. Це обмежує ширину слова, необхідну для зберігання даних і арифметичних операторів в апаратній реалізації [11].

Цей підхід оцінюється на трьох рівнях: першим критерієм є точність, потім модельована незалежна від цілі продуктивність апаратної архітектури в термінах тактових циклів і, нарешті, продуктивність настінного годинника на трьох фактичних платформах FPGA, що охоплюють вбудовану систему та центр обробки даних.

Використовуючи тест Middlebury, алгоритм створює в середньому 8,4 % розбіжностей, що перевищує поріг помилки в один піксель. Різниця між результатами запропонованої архітектури та основною правдою в незакритих областях становить 9,5 % для Конусів, 13,3 % для Тедді, 6,8 % для Цукуби та 4,1 % для Венери. Зразок результату для тестового набору Тедді показано на рисунку 2.



а) б) в)

**Рис. 2. Порівняння невідповідності для набору зображень Тедді:**
**а) Набір зображень Тедді, б) Набір зображень Тедді, в) Невідповідність, спричинена зображенням**

За потреби більшої точності можна досягти, виконавши повну перевірку ліворуч/праворуч або виконавши два проходи по зображенню для використання восьми шляхів. Обидва ці підходи приблизно вдвічі зменшують продуктивність прискорювача, але навіть тоді він буде корисним для обробки в реальному часі.

Моделювання архітектури з точністю до циклу використовувалося для визначення характеристик часу виконання прикладів реалізацій. Порівняння з фактичними реалізаціями апаратного забезпечення показує, що ці симуляції насправді є репрезентативними для кінцевої продуктивності.

Ядро оцінюється за трьома роздільними здатностями зображення: 640 × 480 пікселів (VGA), 1280 × 720 пікселів (720p) і 1920 × 1080 пікселів (1080p). Зображення з роздільною здатністю VGA оцінюються за Dmax = 64, для вищої роздільної здатності Dmax = 128. Для всіх роздільних здатностей кількість тактів, необхідних для завершення одного кадру, визначається шляхом моделювання [12].

Експертиза містить різні композиції крупно- та дрібнозернистого паралелізму. Реалізація описується парою (#p, #d), яка вказує на використання рядкових процесорів #p, причому кожне обчислення #d передбачає невідповідності паралельно. Для кожної з роздільних здатностей використовується автоматичне дослідження простору розробки для генерації 250 альтернатив реалізації, відображених на осі X у порядку збільшення площі або продуктивності. Через обмеження простору лише підмножину альтернатив можна позначити тут за допомогою (#p, #d). Як показано на рисунку 3 для зображень VGA, архітектура добре масштабується зі збільшенням кількості процесорів рядків, аж до нижньої межі 654644 циклів, після чого один піксель обчислюється за 2,11 циклу, а одна диспропорція вимагає 0,033 циклу. Ця конструкція обмежена швидкістю заповнення вхідних буферів, яку можна збільшити ще більше, застосовуючи також методи дрібного розпаралелювання [13]. При передбачуваній тактовій частоті 200 МГц архітектура досягне до 306 кадрів в секунду, як показано на рисунку 4. Такі великі та швидкі системи можна використовувати для розрахунку розбіжностей між кількома камерами для створення об'ємного огляду сцени. Для додатків з низьким енергоспоживанням більш цікавим є мінімальна частота, необхідна для досягнення 30 кадрів на секунду (типова вимога для обробки в реальному часі) [14]. Як показано на рисунку 5, архітектура здатна задовольнити цю вимогу на частоті до 30 МГц для зображень VGA.

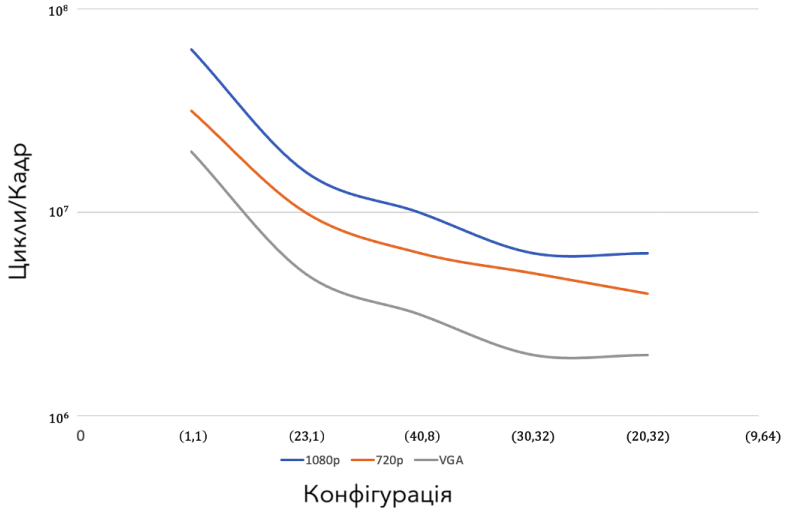



**Рис. 3. Цикли, необхідні для обробки однієї карти невідповідності для різних ступенів паралельності**

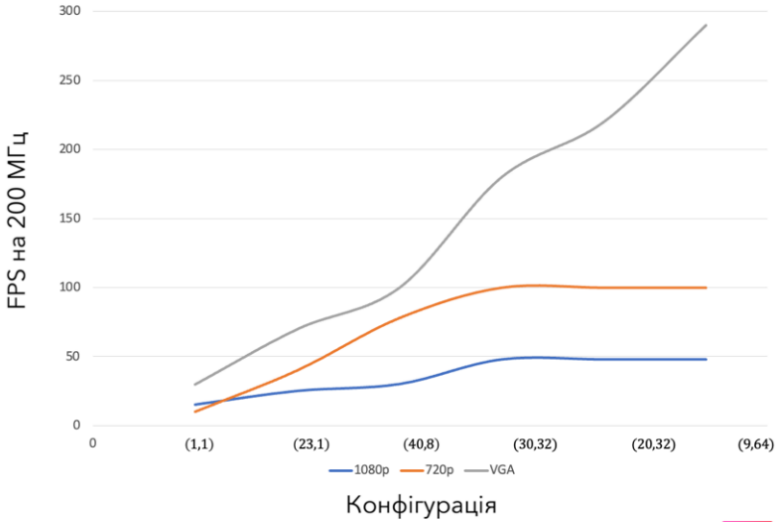



**Рис. 4. Кількість кадрів в секунду досягається запропонованою архітектурою на тактовій частоті 200 МГц**

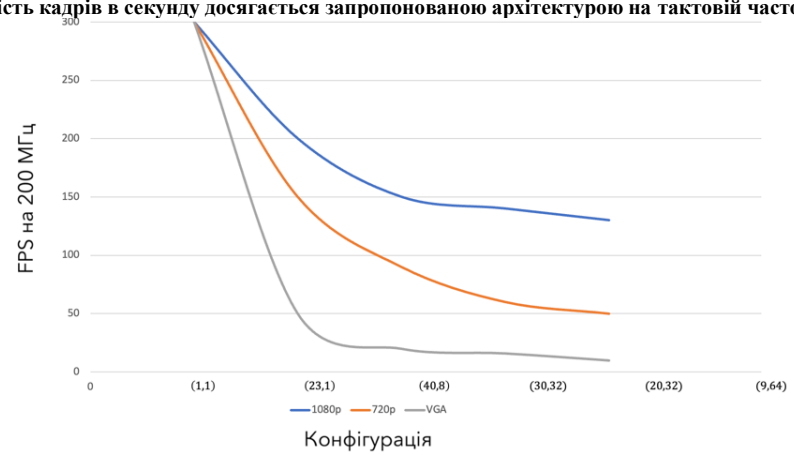



**Рис. 5. Частота, необхідна для досягнення 30 кадрів на секунду на запропонованій архітектурі для різного ступеня паралелізму**

### Експериментальні дослідження методу та програмно-апаратна реалізація

Проектування апаратного забезпечення чимось схоже на програмування програмного забезпечення на мові асемблера, у якому простий розрахунок складається з безлічі інструкцій машинного коду, а з іншого боку, багато логічних блоків утворюють єдиний апаратний модуль. Рисунок 6 показує основну концепцію того, як працюють блоки FPGA. Нижче будуть розглянуті всі логічні блоки, які вводяться для реконфігурації часткового алгоритму [15].

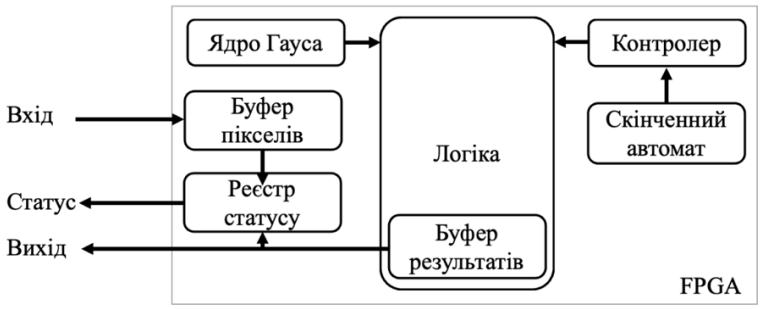



**Рис. 6. Блок-схема реалізації FPGA**

На рисунку 7 показано ядро Гауса, яке використовується для отримання матриці Гарріса A. Цей блок є найпростішим, він не вимагає введення та виведення постійних змінних ядра.

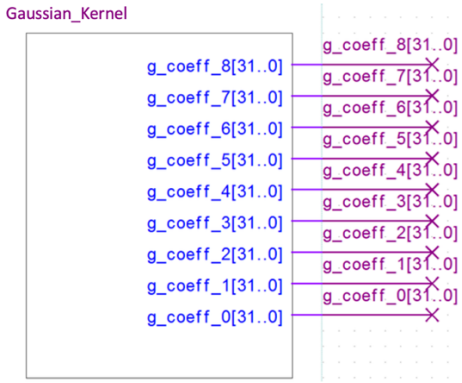



**Рис. 7. Ядро Гауса**

Кожен результат відповідає одному елементу матриці, коефіцієнт 8 представляє перший елемент 0,0001, коефіцієнт 5 є центральним елементом 0,3989 і так далі. Варто зауважити, що кожен параметр є числом з плаваючою точкою, і він отримує 32 біти пам'яті для збереження, тому всі виходи мають 32 біти.

Щоб максимізувати швидкість, було реалізоване апаратне забезпечення якомога паралельнішим. Всі IP-адреси FIFO, надані Altera, є механізмом «один вхід і один вихід», що не зовсім те, чого потрібно досягти, тому FIFO «один вхід і дев'ять виходів» реалізовано, щоб задовольнити потребу, щоб можна було запускати за один прохід цілу згортку з матрицею Гауса 3*3 [16]. Сигнали вхідного FIFO подано в таблиці 1.

Таблиця 1.

**Сигнали вхідного FIFO**

| Назва сигналу | Опис |
|---|---|
| clk | Сигнал годинника |
| data flag | Перевертається, коли вводяться нові дані |
| sw | Перемикач, який керує функцією скидання |
| source[31..0] | Вхідні пікселі, які є 32-розрядним числом типу float |
| FIFO x[31..0] | Вихідний піксель No.x у парі з відповідним елементом матриці Харріса |
| done | Встановлюється на високий рівень, коли FIFO заповнений, інакше залишається низьким |

Спочатку всі 9 виходів встановлювались на нуль, а дані надсилались через один до дев'яти. Коли надсилаються 10-ті дані, усі вихідні дані отримують змінні в певному порядку, найстаріша розташована внизу, а найновіша – зверху. Для цього FIFO № 8 є верхнім, а № 0 – нижнім.

Коли 1-е дані (1010) отримані після трьох тактів, то в наступному такті найстаріші дані (0001) відкидались, а 10-ті дані додавались до FIFO та зберігались в регістрі № 8, і так далі.

Варто зауважити, що кожного разу, коли надходили нові дані, сигнал прапора даних перевертався, цей механізм гарантував, що система не переплутає два послідовних даних, які мають абсолютно однакове значення [17]. Згортка включає як FIFO, так і ядро Гауса, кожен вихід цих двох блоків було об'єднано в дев'ять різних груп. Оскільки алгоритм методу побудови апаратної архітектури для систем комп'ютерного зору на основі FPGA використовує 32-розрядне число типу float для представлення пікселів, звичайні блоки обчислення не могли бути застосовані тут через складність значення float. Altera надає мегафункцію, яка повністю підтримує обчислення плаваючого значення, що робить її ідеальним вибором під час вибору методу побудови апаратної архітектури для систем комп'ютерного зору на основі FPGA [18] (рис.8).

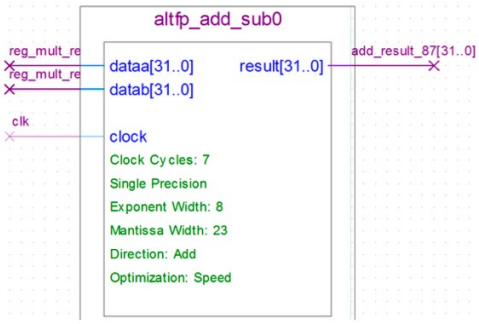



**Рис. 8. Суматор**

На рисунку 8 показано блок додавання після налаштування майстра мегафункцій. Оптимізацію швидкості вибрано, тому що, що є головним золотом дисертації, будь-які функції та можливості, крім чистого додавання, вимкнено з метою економії ресурсів. Такий блок потрібно дублювати багато разів, щоб сформувати функціональний алгоритм. 32-розрядне число з плаваючою точкою означає одинарну точність і має 8-розрядний експонент і також 23-розрядну мантису. Призначення портів досить зрозуміле, і глобальний сигнал clk буде підключений до суматора [19].

Для однієї згортки двох матриць 3*3 реалізовано дев'ять множень і вісім суматорів. Кожен із блоків має період виконання, і останній обчислення може виконуватися лише тоді, коли перший виконає свою роботу. Для надійності всього цього процесу введено кінцевий контейнер.

На рисунку 9 представлена схема кінцевого скінченного автомату. Якщо отриманий сигнал є «високим», тоді спрацьовує арбітр, він починає рахувати всередині та виводить значення на роздільник, розділювач використовує це значення, щоб визначити, яким є поточний стан, і встановити відповідний вихід на «високий» під час налаштування інші на «низький» [20].

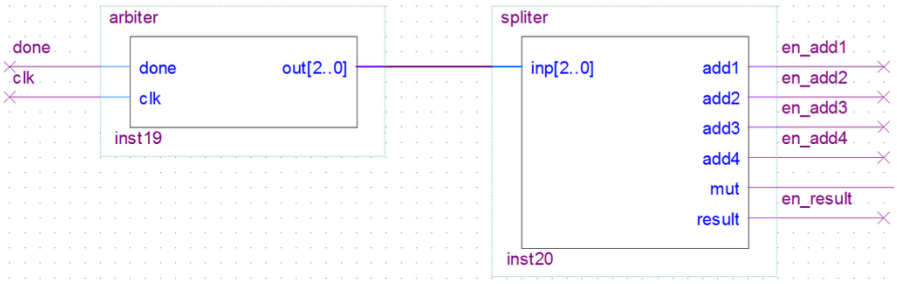



**Рис. 9. Скінченний автомат**

Є чотири регістри додавання та регістр результату. Взявши, регістр add1, входи підключені до попереднього етапу обчислення, незалежно від того, які значення є, регістр завжди зберігатиме те, що він отримує, коли отримано сигнал дозволу «позитивний фронт», вихід порти негайно виведуть дані, які наразі зберігаються, і залишатимуться виводом до наступного «позитивного фронту». Останній етап обчислення, яким буде add2, використовує ці виходи як джерела вхідних даних і передає результат у регістр add2, а процес продовжує виконувати роботу для add3 і add4 [21].

Регістр результату взаємодіє з центральним процесором, який, в свою чергу, є повністю відокремленою системою. Центральний процесор міг запитувати результат кілька разів, навіть якщо було обчислено лише один результат. Через різницю в частоті між процесорами та FPGA, процесори завжди на рівні ГГц, а FPGA – у МГц, що досить повільно порівняно з процесорами [22].

Отже, має бути спосіб захисту кожного запиту, що стосується нових даних. Сигнал прапора повідомляє центральному процесору про те, чи є поточні дані придатними та безпечними для отримання, і це завжди перший сигнал, який буде взаємодіяти в процесі зв’язку між центральним процесором і FPGA, коли процесор отримав «високий» сигнал [23]. Сигнал прапора почне отримувати дані, тим часом результат отримання буде надіслано до FPGA, що вказує на початок процесу. Після отримання результату ЦП знову встановлює «низький» сигнал отримання результату, повідомляє FPGA, що процес завершено. Потім FPGA також встановлює сигнал прапора результату на «низький», щоб, дозволивши ЦП отримати сигнал прапора першим, він міг змусити ЦП чекати, поки не буде отримано новий результат. Такий механізм може значно підвищити надійність системи [24].

Таким чином, цей процес може значно сповільнити швидкість обробки зображень, і поки що це неможливо покращити або уникнути.

**Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі**

В роботі представлені результати дослідження, зокрема, розроблено метод побудови апаратної архітектури для систем комп’ютерного зору на основі FPGA.

В результаті досліджень було встановлено, що метод побудови апаратної архітектури є ефективним підходом до розроблення апаратних прискорювачів для обробки відеоданих.

Даний метод дозволяє значно збільшити швидкість обробки відеоданих, знизити витрати енергії та ресурсів, що забезпечує ефективну інтеграцію з системами розпізнавання образів, навігації, контролю якості та безпеки. Основні положення, що використовуються для розв'язання задач по даній темі, полягають у вивченні особливостей комп'ютерного зору, засобів обробки відеоданих, а також принципів роботи та можливостей FPGA.

Також в роботі подано аспекти практичногго застосування розробленої апаратної архітектури та програмного забезпечення для комп'ютерного зору. Для реалізації систем комп'ютерного зору на основі FPGA використовувались відповідні програмні засоби, що дозволяють створювати програмне забезпечення для FPGA-пристроїв. В роботі представлено програмно-апаратний засіб для систем комп'ютерного зору на основі FPGA, що включає в себе відповідні програмні засоби та апаратну архітектуру. Застосування такого програмно-апаратного засобу дозволяє реалізувати різноманітні завдання обробки зображень та комп'ютерного зору з високою швидкодією та точністю.

## Література


1. Donald G. Bailey. Boca Raton. FPGA-Based Implementation of Signal and Image Processing Systems. FL: CRC Press, 2017. 352 p.
2. Dirk Koch. Designing Embedded Systems with FPGAs. Berlin: Springer, 2018. 284 p.
3. Doug Amos, Austin Lesea, and René Richter. FPGA-based Prototyping Methodology Manual: Best Practices in Design-for-Prototyping. New York, NY: Springer, 2018. 340 p.
4. Zhao Zhang and Xiaojun Liu. Hardware Architecture Design for Real-Time Image Processing on FPGA. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5439629/ (дата звернення: 10.04.2023).
5. Georgios Karakonstantis. FPGA-Based Accelerators for Financial Applications. New York, NY: Springer, 2018. 256 p.
6. Tsung-Hsien Lee and Shuvra S. Bhattacharyya. High-Level Synthesis for Real-Time Digital Signal Processing. New York, NY: Springer, 2018. 276 p.
7. Amirhossein Rezaei and Mohamed Zahran. FPGA-Based Acceleration of Deep Convolutional Neural Networks for Computer Vision: A Survey. URL: https://ieeexplore.ieee.org/document/8393873 (дата звернення: 10.03.2023).
8. Yibin Liang, Yiqiang Chen, and Ming Liu. Design and Implementation of a High-Performance FPGA-Based Embedded Computer Vision System. URL: https://ieeexplore.ieee.org/document/7273657 (дата звернення: 10.03.2023).
9. A. Sudhakar, P. Ravinder Reddy, and P. Rajesh Kumar. FPGA Implementation of Image Processing Algorithms: A Review. URL: https://www.sciencedirect.com/science/article/pii/S2212671620303608 (дата звернення: 11.03.2023).
10. Ahmad Ibrahim and Asim El-Sheikh. FPGA-Based Implementation of Neural Networks. New York, NY: Springer, 2018. 220 p.
11. L. M. Bergasa, O. Yebes, and R. Arroyo. An Efficient FPGA Implementation of a Real-Time Object Tracking System. URL: https://ieeexplore.ieee.org/document/6867583 (дата звернення: 11.03.2023).
12. Hamed Habibi Aghdam, Fatemeh Saberian, and Seyed Saeid Moosavi. FPGA-Based Embedded System for Real-Time Object Recognition. URL: https://www.mdpi.com/2079-9292/9/6/917 (дата звернення: 12.03.2023).
13. FPGA-based Embedded Vision System Design. URL: https://www.xilinx.com/products/design-tools/vitis/embedded-vision.html (дата звернення: 12.03.2023).
14. Mark Zwolinski. Designing Digital Systems with SystemVerilog. New York, NY: Springer, 2018. 368 p.
15. Pong P. Chu. Hoboken. FPGA Prototyping by VHDL Examples: Xilinx MicroBlaze MCS SoC". NJ: Wiley-IEEE Press, 2019. 392 p.
16. Mohammad G. Rastegari and Mohammad H. Kahani. FPGA-Based Computer Vision: A Survey. URL: https://ieeexplore.ieee.org/document/8563013 (дата звернення: 10.03.2023).
17. Marek Gorgon and Radim Burget. Designing FPGA-based Computer Vision Systems. URL: https://www.mdpi.com/2079-9292/8/7/724 (дата звернення: 11.03.2023).
18. George A. Constantinides, Peter Y. K. Cheung, and Wayne Luk. Boca Raton. High-Level Synthesis for FPGA Design: From Prototyping to Deployment. FL: CRC Press, 2018. 436 p.
19. Joseph J. Lee and Ayesha Fatima. Deep Learning with FPGA: From Device to Algorithm. Cham, Switzerland: Springer, 2021. 188 p.
20. F. D. B. Pereira and R. M. A. Pereira. FPGA-based Hardware Accelerator for Object Detection in Computer Vision. URL: https://ieeexplore.ieee.org/document/8607876 (дата звернення: 11.03.2023).
21. J. M. Rodriguez-Ramos, J. M. Rodriguez-Delgado, and J. M. Santana. An FPGA-based real-time vision system for detecting and tracking vehicles. URL: https://ieeexplore.ieee.org/document/8525406 (дата звернення: 12.03.2023).
22. Wayne Wolf. FPGA-based System Design. Boca Raton, FL: CRC Press, 2019. 350 p.
23. Pong P. Chu. FPGA Prototyping Using Verilog Examples: Xilinx Spartan-6 Version. Hoboken, NJ: Wiley-IEEE Press, 2018. 352 p.
24. N. C. Kumar, V. Anand, and M. C. Padma. An FPGA-Based Smart Vision System for Object Detection and Tracking. URL: https://ieeexplore.ieee.org/document/8880966 (дата звернення: 14.03.2023).