УДК 004.312.466-048.34 **doi: 10.32620/aktt.2023.5.07**

В. О. КУЛАНОВ, А. Є. ПЕРЕПЕЛИЦИН

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна

МЕТОД СТВОРЕННЯ І ВПРОВАДЖЕННЯ FPGA ПРОЄКТІВ СТІЙКИХ ДО ЗМІН ВИМОГ І СЕРЕДОВИЩ РОЗРОБЛЕННЯ ДЛЯ ХМАРНИХ ІНФРАСТРУКТУР

Предметом *вивчення в даній статті є сучасні технології програмовної логіки, історія змін продукції провідних виробників, включаючи середовища розроблення, а також процеси оптимізації проєктів.* ***Метою*** *роботи є вдосконалення сучасних методів, технологій і інструментальних засобів розроблення та інтеграції FPGA-as-a-Service (FaaS) в якості сервісів у складі хмарних інфраструктур, дата-центрів і бортових високопродуктивних систем, з урахуванням умов і вимог, що постійно змінюються.* ***Завдання:*** *проаналізувати історію змін продукції провідних виробників елементної бази програмовної логіки; проаналізувати динаміку змін функціональності середовищ розроблення проєктів для систем програмовної логіки з урахуванням існуючих вимог і обмежень з боку компаній-розробників програмно-апаратних платформ; виконати порівняльний аналіз сучасних макетних плат і карт прискорювачів для розроблення і тестування проєктів, побудованих на базі мікросхем програмовної логіки; виконати наліз використання існуючих рішень на базі FPGA технологій в складі хмарних сервісів сучасних Cloud-провайдерів; запропонувати практичні кроки щодо розроблення систем на базі програмовних логічних інтегральних схем (ПЛІС), стійких до зміни вимог; запропонувати послідовність для розроблення та оптимізації високопродуктивних систем з реалізацією їх на базі ПЛІС; навести приклад практичного застосування запропонованого методу. Відповідно до поставлених завдань, були отримані наступні* ***результати****. Проаналізовано історію змін провідних компаній виробників програмовної логіки, та змін версій середовищ розроблення і набору рішень однієї із найбільших компаній. Детально проаналізовано використання існуючих рішень на базі FPGA в складі хмарних сервісів. Надані дві послідовності для оптимізації проєктів і підвищення їх продуктивності та зниження трудовитрат під час їх створення і перенесення на нові версії програмних та апаратних засобів. На основі результатів дослідження був розроблений і протестований прототип, який дозволив застосувати запропонований метод на практиці для адаптації проєкту FPGA as a service під час перенесення на іншу версію карти прискорювача.* ***Висновки:*** *Головний внесок і наукова новизна отриманих результатів полягає в тому, що проведено експериментальне дослідження парадигми неперервного перепрограмування програмовної логіки, яке дозволило сформулювати елементи нового методу створення проєктів в якості сервісу для хмарних інфраструктур, дата-центрів і систем штучного інтелекту. Запропоновано набір практичних кроків розробки систем, які є стійкими до змін умов та вимог. Застосування запропонованого методу дозволяє уникнути витрат на підтримку проєкту у разі зміни вимог.*

Ключові слова: *FPGA; ПЛІС; FPGA як сервіс; хмарні сервіси; середовища розроблення; зміна вимог; апаратні прискорювачі.*

Вступ

У сучасному світі, де стрімко розвивається технологічний прогрес, важко переоцінити важливість наукових досягнень та технологічних інновацій у галузі техніки і технології. Перехід до нових рішень в частині технології створення мікросхем програмовної логіки або ПЛІС, які в рамках роботи термінологічно еквівалентні FPGA, безпосередньо впливає на багато галузей критичного застосування, включаючи бортові рішення аерокосмічних систем [1, 2].

Для систем у цій сфері надійність є критичним фактором. Це викликає зацікавленість в якнайшвидшому переході до сучасних рішень у розробці програмного-апаратних платформ. Зміни в вимогах до навігації, комунікації, вимірювань і автоматизації потребують постійного вдосконалення технічних підходів та гнучкості у реагуванні на зміни.

Життєвий цикл багатьох сучасних систем може становити дуже короткий термін – до двох років. Це обумовлено швидкістю зростання сучасного маркету, зміни вимог з боку користувачів (замовників) та експоненційним розвитком технологій [3].

Сьогодні, розроблення нових рішень може займати час, який можна порівняти з часом їх життя. У зв'язку з цим необхідно відповідним чином ставитися до вибору виробника, інструментальних засобів та апаратної платформи. Такий вибір є критично важливим, адже може виявитися, що в найближчі кілька років вкладені кошти в побудову проєктів під

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

певну програмно-апаратну платформу можуть бути втрачені. Одна з причин – виробник припиняє підтримку донедавна поширеного сімейства мікросхем або програмних продуктів та заміняє його на наступне покоління – більш сучасне [4, 5].

Важлива річ, яку слід враховувати перед початком старту нового проєкту – це потенційна залежність (Vendor Lock) від постачальника програмно-апаратної платформи. В цьому випадку виробник продукту може значно вплинути на інфраструктуру або архітектуру проєкту через функціональні обмеження або інші чинники з технологічної точки зору.

Компаніям розробникам програмової логіки надто складно підтримувати паралельно кілька інструментальних засобів, тому набір політик цих компаній диктує обмеження можливості використання попередніх версій [6]. При цьому технології програмовної логіки з використанням цих засобів дають змогу здійснювати швидку розробку з використанням гетерогенних обчислень [7]. Це робить можливим неймовірно швидке прототипування та моделювання алгоритмів і складових частин при побудові систем штучного інтелекту [8, 9].

Альтернативний спосіб створення проєктів на програмовній логіці, який передбачає використання парадигми FPGA-as-a-Service (FaaS) – надання в оренду апаратної платформи ПЛІС через постачальників хмарних послуг (Cloud Providers) [10].

Цей підхід дозволяє розробникам створювати проєкти на програмовних логічних інтегральних схемах без значних витрат на придбання та обслуговування обладнання. Замість цього, інженери можуть орендувати FPGA-ресурси за потребою [11, 12] та масштабувати їх відповідно до вимог проєкту. Цей підхід також допомагає уникнути проблеми залежності від виробника та забезпечує доступ до сучасних FPGA-ресурсів у хмарному середовищі. Використання FaaS може бути особливо корисним у випадках, коли швидкість прототипування та можливість масштабування проєкту є критичними факторами успіху.

Таким чином, пошук стратегій проєктування в умовах постійних змін вимог, технологій та програмно-апаратних засобів є вкрай важливим чинником на шляху оптимізації витрат для створення кінцевого продукту.

Метою даної роботи є вдосконалення сучасних методів, технологій і інструментальних засобів розроблення та інтеграції FPGA-as-a-Service в якості сервісів у складі хмарних інфраструктур, дата-центрів і бортових високопродуктивних систем, з урахуванням умов і вимог, що постійно змінюються.

Для досягнення поставленої мети, в рамках даної роботи, розглядаються та вирішуються наступні **задачі**:

1) проводиться аналіз історії змін продукції провідних виробників елементної бази мікросхем програмовної логіки;

2) аналізується динаміка змін функціональності середовищ розроблення проєктів для систем програмовної логіки з урахуванням вимог та існуючих обмежень з боку компаній-розробників програмно-апаратних платформ;

3) проводиться порівняльний аналіз сучасних макетних плат для розроблення і тестування проєктів, побудованих на базі мікросхем програмовної логіки, для створення єдиної моделі таких систем;

4) аналізуються використання існуючих рішень на базі FPGA технологій в складі хмарних сервісів сучасних Cloud-провайдерів;

5) пропонуються практичні кроки щодо розроблення систем на базі ПЛІС, стійких до зміни вимог;

6) пропонується послідовність розроблення та оптимізації високопродуктивних систем на ПЛІС;

7) наводиться приклад застосування методу створення і впровадження FPGA проєктів стійких до змін вимог і середовищ розроблення.

1. Аналіз історії зміни продукції виробників програмовної логіки

Системи, які використовують ПЛІС передбачають перепрограмування безпосередньо в процесі їх експлуатації. В першу чергу це обумовлено тим, що перехід до нових технологічних процесів – до 7 нм включно, вимагає дуже великих фінансових витрат на розроблення.

Набагато дешевше створювати перепрограмовані рішення, які можуть бути налаштовані відповідним чином, відповідаючи на виклики зміни вимог з боку замовника та/або умов функціонування тощо. Наявність серійного випуску таких рішень, із наданням користувачеві можливості програмування та перепрограмування їх під свої потреби, відкриває багато можливостей та впливає на супутні процеси випуску цифрових платформ.

Серед провідних фірм розробників елементної бази ПЛІС можна відслідковувати зміну продуктів чотирьох із них: Intel, AMD (Xilinx), Microsemi, Lattice Semiconductor [3].

Lattice і Microsemi займаються виробництвом елементної бази мікросхем, які можуть бути запрограмовані для різноманітних інтегрованих рішень. Рішення цих компаній досить прості, що забезпечує можливість глибокого аналізу їх мікросхем на предмет закладних пристроїв та наявності дефектів.

Вони знаходять застосування під час побудови критичних систем у всіх країнах світу. При цьому їх функціональні можливості дозволяють реалізовувати досить серйозні проєкти.

Середовища розроблення таких виробників має значно обмежені можливості. При цьому вони функціонально цілком прийнятні для створення сучасних проєктів та мають багато спільного з інструментарієм провідних фірм виробників з більш ніж тридцятирічним досвідом. До лав провідних фірм-виробників (флагманів) можна було віднести дві компанії: Xilinx і Altera.

Цікаво зазначити, що боротьба між двома гігантами сучасності в галузі створення мікропроцесорної електроніки – компаніями Intel та AMD продовжується і в царині виробництва мікросхем програмовної логіки.

Так, наприклад, компанія AMD виробляє продукцію Xilinx [13], а компанія Intel роками раніше поглинула компанію Altera, яка зараз отримала новий бренд і називається Intel-Altera.

Поява нових типів процесорів, включаючи M1, виконаного на основі архітектури ARM також говорить про динаміку на ринку випуску напівпровідникової техніки. Саме ця архітектура використовується в FPGA в якості процесорного ядра. Для вбудованих рішень такі процесори здатні надати баланс продуктивності та низького енергоспоживання.

Наявність конкуренції є можливою умовою зниження цін і, навіть, прискорення в процесі втілення нових ідей. В рамках цього дослідження основний фокус робиться на продукцію компанії Xilinx, оскільки програмовані рішення Intel-Altera були розглянуті в рамках попередніх досліджень [2].

2. Аналіз динаміки змін середовищ розроблення ПЛІС-проєктів

Однією з найважливіших цілей для розробника є зниження трудомістких витрат у процесі створення продукту. Досягти цієї мети прагне переважна більшість, а тому доступний арсенал засобів і стратегій дуже великий. До них можна віднести і додавання «надлишкових» ресурсів, що можливо завдяки збільшенню кількості доступної оперативної пам'яті для засобів розроблення.

При цьому для самого розробника складність розробки суттєво знижується. Все це показує загальну тенденцію для середовищ розроблення, і це відповідає тренду автоматизації процесу створення проєктів загалом.

На певних етапах розвитку компанія Xilinx приділяла багато часу та пропонувала середовище розроблення ISE. В якості порівняння, популярним рішенням для компанії Altera тих років було середовище розроблення Quartus II, яке, навіть, після ребрендингу залишилося з тим самим ім'ям у лінійці продуктів Intel.

Компанія Xilinx відмовилася від ISE. З 2014 року на зміну йому прийшло середовище Vivado, яке пізніше увійшло до складу лінійки потужніших продуктів [14]. Це дозволили здійснювати синтез проєктних рішень з можливістю поєднання класичного процесу розроблення FPGA з використання більш високорівневих сутностей. До них варто віднести використання мови C і C++, OpenCL та підтримки інших технологій, що відповідають усім сучасним запитам автоматизації процесу розроблення.

У період після 2016 року аж до 2018 року на ринку існував і існує досі SDx пакет, який включав SDAccel. Це потужне рішення для програмування програмовної логіки класу FPGA з використання всіх її переваг, включаючи нову парадигму програмування – програмування «на льоту» [15].

Зв'язки між середовищами розробки під час переходу між ними показують паралельне існування кількох середовищ розроблення (рис. 1).

Рис. 1. Історія змін середовищ розроблення компанії Xilinx, а далі AMD, за минуле десятиліття

Паралельно з середовищем SDAccel компанія розробляла протягом 5 років середу розроблення Vitis [16], яка включила більшість функцій, у тому числі функції SoPC (System on Programmable Chip), платформ для побудови штучного інтелекту і з перенесенням різних мов на кінцеву платформу у вигляді RTL (Register Transfer Level) описів для програмування FPGA [17].

Після закінчення цього періоду підготовки та випуску середовища Vitis, підтримка середовища розроблення SDAccel та сумісних пакетів розробки було припинено. Такий швидкий перехід до нових середовищ розроблення призвів до необхідності адаптації всіх проєктів, що існували. І це означає, що слід підготуватися до наступної зміни.

Незважаючи на зав'ялену підтримку інтеграції git, заявлену підтримку систем контролю версій, обидва середовища не можуть цього робити навіть за заявою технічної підтримки компанії Xilinx. Технічно SDAccel та Vitis передбачають такі інструменти, але на практиці вони не працюють.

Між цими середовищами розроблення неможливо перенести проєкти без повного покрокового їх

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

відтворення – неможливо виконати експорт та імпорт, щоб після цього проєкт можна було зібрати.

Але при цьому зовнішні відмінності (графічний інтерфейс) майже не відрізняється, а існування зворотної сумісності декларується виробником, але, на практиці, не працює. Відсутність зворотної сумісності суттєво ускладнює підтримку існуючих проєктів.

У зв'язку з цим необхідно дійти зрозумілої стратегії ведення процесу розроблення та підтримки проєктів з урахуванням динамічної зміни середовищ проєктування. Це вимагає безперервного моніторингу існуючих рішень для готовності до можливих модифікацій.

Ще одним продуктом Xilinx є фреймворк XRT (Xilinx Runtime), який використовується для забезпечення зв'язку з FPGA та безпосереднього програмування. Цей фреймворк являє собою інтерфейс між FPGA рішеннями у форматі PCIe карт та host-рішеннями на базі виключно деяких версій операційних систем Ubuntu або CentOS.

Середовище розроблення Vitis не підтримується в операційній системі Windows, при тому що складові компоненти, включаючи Vivado, працюють без зайвих проблем. Цей фреймворк також розвивався паралельно з середовищами розроблення та демонструє аналогічні проблеми сумісності FPGA рішень різних випусків.

Фреймворк XRT здійснює моніторинг всіх процесів, пов'язаних з FPGA-картою (board), моніторинг температури, моніторинг доступу (фаєрвол) тощо. Основне його призначення пов'язано зі здійсненням операцій обміну даних, і програмуванням «на льоту», що дозволяє змістити фокус із класичних FPGA систем.

При переході до розробки з використанням нових середовищ розробки необхідно змінити парадигму сприйняття місця FPGA в системі, що розробляється. Як не дивно, цей перехід є особливо складним для досвідчених розробників.

При програмуванні «на льоту» найскладніша FPGA система представляється як функція С-подібною мовою, яка може бути безпосередньо викликана з програми на host-комп'ютері, що виконується.

При цьому вся система є викликом в одному рядку коду з передачею параметрів. Для програміста весь процес взаємодії з FPGA представляє просто виклик функції та очікування прапора результату готовності оброблених даних та можливості їх читання з буферів. Такий спосіб взаємодії зручний, але вимагає правильного розуміння місця FPGA у такій системі. XRT дозволяє практично миттєве програмування.

Для програмування на льоту рекомендується підготувати набір заздалегідь скомпільованих IP-ядер (безпосередньо підготовлених бінарних файлів необхідних проєктів для завантаження FPGA).

Вони можуть вказуватися як параметри в host-програмі і викликатися в будь-якій послідовності практично миттєво. Тривалість виконання інструкцій на кожному з IP-ядер залежить від розв'язуваних завдань і ця частина близька до звичної парадигми роботи FPGA систем.

Слід зауважити, що особливості взаємодії PCIe не дозволяють підтримувати виконання ядер більше кількох десятків секунд, тому слід здійснювати обробку даних порціями.

3. Аналіз сучасних макетних плат для розроблення ПЛІС-проєктів

Аналіз змін середовища розроблення передбачає розгляд історії розвитку інструментального засобу від початку його створення до сьогоднішнього часу.

Розгляд, безпосередньо, технологій та макетних плат, які використовуються дата-центрами для побудови систем обробки даних в тому числі в складі систем класу FPGA-as-a-Service, показує наявність перебування їх в стані постійних змін [13].

Власне, всі ці плати виконуються у форматі стандартних PCIe розширень, які часто містять додаткове живлення, активне охолодження та додаткові інтерфейси для комунікації зі зовнішнім світом. Також не виключається і більш прості варіанти реалізацій, які використовуються як бюджетна альтернатива за рахунок зменшення додаткової апаратної периферії та ресурсів (об’єм оперативної пам’яті, тип інтерфейсу взаємодії з користувачем/host-системою тощо).

На сьогодні існує багато таких альтернатив. Одним з таких рішень є VCU1525, потужна макетна плата, яка не містить динамічної пам'яті і була орієнтована на побудову різних класичних обробників, відеоконверторів тощо. Також були впроваджені плати U200, U250, які дозволили працювати з динамічною пам'яттю класу DDR4 (розширювана) [12].

Потреби в збільшенні швидкості обміну даними між системою та її компонентами прискорило перехід до технології HBM (High Bandwidth Memory). Це загальний тренд перенесення в середину кристала всіх можливих перемичок і підкладок, задля компонування системи в одному корпусі, і відповідно, як сумарний результат, отримання переваг в частині зниження енергоспоживання, підвищення швидкодії тощо.

Технологія HBM має кілька версій, включно з HBM2e. Компанія Hunix розпочала виробництво таких чіпів надшвидкої динамічної пам'яті HBM2e. В одному чіпі 16 ГБ пам'яті пропускна здатність такої пам'яті до 460 гігабайт в секунду [15].

Це недосяжна швидкість у багатьох практичних реалізацій, оскільки вона порахована на основі пропускної спроможності інтерфейсу. На практиці досяжна швидкість у режимі лінійного читання може досягати 400 гігабайт, у разі довільного читання це може бути близько 260-280 гігабайт на секунду.

В основі таких макетних плат є чіп FPGA з категорії UltraScale, який може запропонувати велику кількість ресурсів за немалі фінансові витрати.

Альтернативний варіант – використання Engineering Sample, які дозволяють працювати з платою до виходу її на основний ринок, що, насправді, ніяк не гарантує, що кінцевий варіант не буде мати зовсім іншу елементну базу. А отже процес подальшої підтримки прошивок залишається відкритим.

У зв'язку з цим компанії розробники розглянути варіанти здешевлення продукції, які дозволяють знизити вартість макетних плат. Один з таких підходів – побудова функціонально дуже близького та схожого на флагманський варіант, але з певною апаратною оптимізацією, жертвуючи продуктивністю, в 3 рази за деякими пунктами, 1,5 рази за деякими пунктами.

У практичному сенсі, за такого підходу, такі плати можна задіяти лише на чверть від їхньої потенційної продуктивності. При цьому деякі умільці просто додають активне охолодження до таких рішень і це дозволяє суттєво підвищити цей результат за тієї ж вартості.

Через таку політику компаній існує ціла низка плат, які швидко перестають підтримуватися, незважаючи на те, що вони мають розширену номенклатуру апаратних ресурсів, достатньо розповсюджені та дорого коштують, але вони просто не підтримуються самими засобами розроблення.

Наприклад, стара версія середовища розроблення не дозволяє скомпілювати проєкт із тактовою частотою для заданої плати, нова – дозволяє, але вона вже не підтримує цю плату. При тому, що функціональність між ними немає особливої різниці. Такі плати, в результаті, не потрапляють у масове використання, а застосовуються виключно на етапі розроблення.

4. Аналіз використання існуючих рішень на базі FPGA в складі хмарних сервісів

На сьогоднішній день відомі Cloud-провайдери та компанії розробники елементної бази мікросхем програмовної логіки надають доступ до FPGA-ресурсів, розгорнутим в хмарному середовищі, у вигляді сервісу (табл. 1).

Розглянемо особливості застосування та особливостей кожного з сервісів.

Amazon Web Services (AWS) надає FPGA-ресурси в рамках Amazon EC2 F1. Використовується для різних задач, включаючи машинне навчання, обробку великих об’ємів даних, аналітику тощо. Однією з ключових особливостей є можливість розроблення власних апаратних прискорювачів за допомогою FPGA, використовуючи мови програмування, такі як Verilog і VHDL.

Таблиця 1

FaaS в складі хмарної інфраструктури відомих провайдерів

Провайдер	**Назва сервісу**	**Особливості**
Amazon Web Services (AWS)	Amazon EC2 F1	Попередньо зібрані апаратні прискорювачі, інструменти розроблення FPGA
Microsoft Azure	Платформа Azure FPGA	Високопродуктивні ресурси FPGA, підтримка декількох фреймворків
Google Cloud Platform	Обчислювальний механізм з FPGA	Екземпляри FPGA, які можна налаштовувати, доступні засоби розроблення
Alibaba Cloud	FPGA Elastic Compute Service (ECS)	Прискорювачі FPGA для штучного інтелекту та аналітики даних
AMD (Xilinx)	Карти FPGA прискорювачів Alveo для дата-центрів	Власні ресурси, виробництво FPGA, та інструменти розроблення
Intel	DevCloud	Ресурси та засоби розроблення FPGA

Microsoft Azure пропонує FPGA-прискорення через Azure FPGA Platform. Цей сервіс широко використовується для великих обчислень та інтенсивних обчислень у галузі наукових досліджень. Однією з технічних особливостей є підтримка різних фреймворків для розробки FPGA-програмного забезпечення, включаючи Intel oneAPI Toolkits та OpenCL.

Google Cloud Platform (GCP). FPGA-прискорення надається через Compute Engine з FPGA, застосовується для аналізу даних, обробки медіа, Edge Computing та розгортання штучного інтелекту на пристроях IoT (Internet of Things). Технічною особливістю є можливість користувачів налаштовувати FPGA-екземпляри для конкретних завдань та розробляти власні прискорювачі за допомогою високорівневих мов, таких як C, C++, або Python.

Alibaba Cloud пропонує FPGA-прискорення через FPGA Elastic Compute Service (ECS). В Азії та

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

Китаї цей сервіс широко використовується для розробки та розгортання рішень у сфері штучного інтелекту, фінансів та інших галузей. Технічною особливістю є підтримка FPGA-розробки з використанням інструментів, наданих Alibaba Cloud.

AMD (Xilinx), як один з провідних виробників FPGA, надає FPGA-ресурси через Alveo Data Center accelerator cards. Цей сервіс користується популярністю серед розробників FPGA-програмного забезпечення та спеціалістів у сфері вбудованих систем. Технічною особливістю є доступ до ресурсів та інструментів розроблення FPGA від Xilinx, включаючи високорівневі мови програмування, такі як C/C++ та Python.

Intel також надає FPGA-ресурси через DevCloud. Сервіс широко використовується в галузі обробки даних – особливо в фінансових організаціях та дослідницьких лабораторіях для оптимізації обчислень та аналізу даних. Технічною особливістю є підтримка FPGA-розробки з використанням інструментів та бібліотек від Intel.

5. Практичні кроки для побудови систем на ПЛІС, що довго підтримуються

За результатами проведеного аналізу, існує можливість запропонувати практичні кроки щодо пошуку можливих рекомендацій для побудови систем, що довго підтримуються, із зазначенням для розробників FPGA as a Service з використанням інструментальних засобів від провідних виробників, що необхідно будувати ці рішення з урахуванням цих рекомендацій.

Таким чином, для побудови рішень FPGA as a Service для впровадження та підтримки на FPGA ресурсах існує необхідність взяти до уваги проблеми, пов'язані з періодичністю зміни середовищ розробки, а значить скористатися кроками:

1. Наслідування хорошого тону, пов'язаного з незалежністю реалізації ПЛІС проєкту від конкретики даного інструментального засобу.

2. Передбачати можливість модифікації складових частин та можливість перенесення таких рішень (спрощеного портування таких рішень) до оновлених модифікованих середовищ розробк.

3. Використовувати можливість організації процесів ведення підтримки проєкту без використання графічної оболонки середовища, а з використанням інтерфейсу командного рядка, який має зворотну сумісність між такими інструментальними засобами розробки та середовищами.

4. Використання безперервного механізму зворотного зв'язку з розробником такого середовища розроблення для усунення проблем, пов'язаних із сумісністю поточних та інструментальних засобів, що розробляються, оскільки вони мають інформацію про впровадження нових або підтримки команд, які використовуються такими середовищами розроблення ще до того, як ця інформація з'явиться у публічному доступі у вигляді специфікації і набуде чинності та підтримки в новому випущеному середовищі розроблення.

6. Послідовність розробки і оптимізації високопродуктивних систем на ПЛІС

В ПЛІС існує проблема, пов'язана з прив'язкою до окремих регіонів кристала, вони називаються SLR. Це регіони усередині кожної ділянки кристала з фіксованим набором міжз'єднань. Існує можливість прив'язки таких рішень до окремих SLR, пов'язаних зі специфікою проєкту.

Наприклад, одна з версій інструментального середовища розроблення надає можливість прив'язки окремих частин проєкту до окремих ділянок на кристалі SLR. Для одного з кристалів така можливість є, для іншого кристала – такої можливості немає, незважаючи на те, що це та сама версія плати. Причина дуже проста – одна з плат є Engineering Sample, яка є рядовим, масово виробленим екземпляром.

Оскільки основний проєкт розробляється на інженерній версії під час перенесення його на основну плату відкривається можливість її тривалої підтримки. При цьому можливість прив'язки окремих ділянок, окремих елементів регіону до конкретних частин кристала.

Проблема пов'язана з неможливістю використання цього інструменту на етапі розробки та необхідністю повторного проєктування для оптимізації отриманих рішень.

З урахуванням таких обмежень можна запропонувати наступну стратегію підвищення продуктивності та спрощення трасування для ПЛІС карт, що надають свої ресурси як сервіс, а саме, використання розділених вузлів, за допомогою набору ланцюжків регістрів на підставі рекомендацій від компаній виробника, щоб надати можливість інструментальними засобами здійснювати трасування таких рішень.

Для організації такої архітектури рекомендується додавати набір з двох або трьох буферів з послідовним зв’язком, з можливістю конвеєризації процесу обробки даних.

У цьому випадку, одним з унікальних способу побудови таких систем є можливість додавання тега до оброблюваних даних, що може дозволити зв'язати в часі деяку унікальну мітку, яка міститиме досить компактне уявлення друкарської помилки з декількох десятків біт (у разі обробки широких масивів даних).

Це дозволить істотно спростити процес конвеєризації та надати можливість здійснити повноцінну прив'язку та асоціацію даних з унікальними ідентифікаторами.

Далі розділяти окремі проєкти та переміщати їх в інший блок SLR (Super Logic Region).

Такий підхід дозволяє здійснювати синхронізацію оброблюваних даних на кінцевих вузлах обробки з додаванням довільної кількості проміжних ступенів у вигляді набору регістрів, що може спростити процес трасування таких рішень, для отримання кінцевої частоти такого ренію, а значить і продуктивності і спростити процес трасування.

Дотримання цих рекомендацій дозволяє з одного боку зменшити час компіляції таких проєктів, з іншого боку це може дозволити підвищити шанс отримання бажаної фінальної частоти за підсумком компіляції для такого рішення.

Крім цього, слід виділити можливість обмеження кількості паралельно оброблюваних рішень, паралельно оброблюваних даних в рамках одного набору даних для зменшення розрядності окремих рішень через обмежену кількість вхідних та вихідних фізичних ліній комутацій у складі окремих SLR регіонів.

Дотримання цих рекомендацій може дозволити знизити трудовитрати при перенесенні одного проєкту на нові плати, нові карти ПЛІС, без додаткових трудовитрат на модифікацію таких проєктів.

Іноді вимоги змінюються з боку середовищ розробки та цільових карток, і це привносить уточнення до вимог замовника.

7. Практичний приклад застосування запропонованого методу

Результати досліджень були застосовані при проєктуванні сервісу обробки зображень з використанням FPGA карт прискорювачів Alveo U280 у середовищі розробки SDAccel 2018. Далі проєкт було перенесено у середу розробки Vitis 2019, а далі перенесено до Vitis 2020. Усі три названих середовища несумісні і звичайне перенесення вимагає покрокового створення всіх частин проєкту та налаштування параметрів компілятора. Далі виконано перенесення проєкту у Vitis 2021 та Vitis 2022.

У разі зміни середовища розробки змінюється можливість реалізації окремих елементів.

Наприклад, якщо на окремих платах була можливість зробити паралельну шину шириною 2048 біт, то в іншої плати через неможливість перенесення таких рішень з однієї області SLR до іншої виникає необхідність. або змінити кількість модулів, які будуть обробляти дані паралельно для звільнення певної кількості ліній зв'язку, або змінити розрядність такої шини до 1024 або 512. Ця зміна вимог до проєкту може виникнути через зміни середовища або цільового чіпа.

У випадку U280 чіп включає три окремі області SLR. До SLR0 підключені два кристали HBM через 16 наборів портів по 256 біт. Для підвищення продуктивності компоненти проєкту пов'язані з пам'яттю були поміщені в SLR0. При цьому відповідно до запропонованої послідовності бралася до уваги доступна для використання кількість вільних виводів у кожній області SLR (рис. 2).

Слідування запропонованим крокам дозволило перенести проект до карти Alveo U50 без додаткової модифікації проекту, у тому числі з урахуванням іншого числа виводів в SLR.

Висновки

Виконаний аналітичний огляд історії розвитку продукції провідних виробників програмовної логіки, включно з аналізом динаміки зміни середовищ розроблення систем програмовної логіки. Також виконаний порівняльний аналіз макетних плат для розроблення і тестування проєктів програмовної логіки. Розглянуто практичний досвід процесу переходу до нових рішень розроблення систем програмовної логіки, яке продиктовано швидкістю зростання технологій.

Виконаний порівняльний аналіз макетних плат для розроблення і тестування проєктів програмовної логіки. Розглянуто практичний досвід процесу переходу до нових рішень розроблення систем програмовної логіки, яке продиктовано швидкістю зростання технологій.

Також було проведено порівняльний аналіз макетів для розробки та тестування програмованих логічних проєктів. Розглянуто практичний досвід процесу портування проєктів у нові середовища розробки для програмовної логіки. Було показано, що така потреба продиктована швидкістю розвитку технологій. Показано, що внутрішньосистемне динамічне програмування рішень значно спрощує процес побудови ПЛІС як сервісу.

Розглянуто етапи побудови проєктів для систем штучного інтелекту з використанням програмовної логіки, засоби розробки, проаналізувати базові елементи, а також елементна база, що випускається, для їх підтримки.

Встановлено, що динаміка зміни середовищ розробки повинна братися до уваги під час планування проєктів FPGA для зниження ризиків. Програмування рішень на льоту істотно спрощує процес побудови FPGA як сервіс, а також розробку та перевірку роботи компонентів систем штучного інтелекту.

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

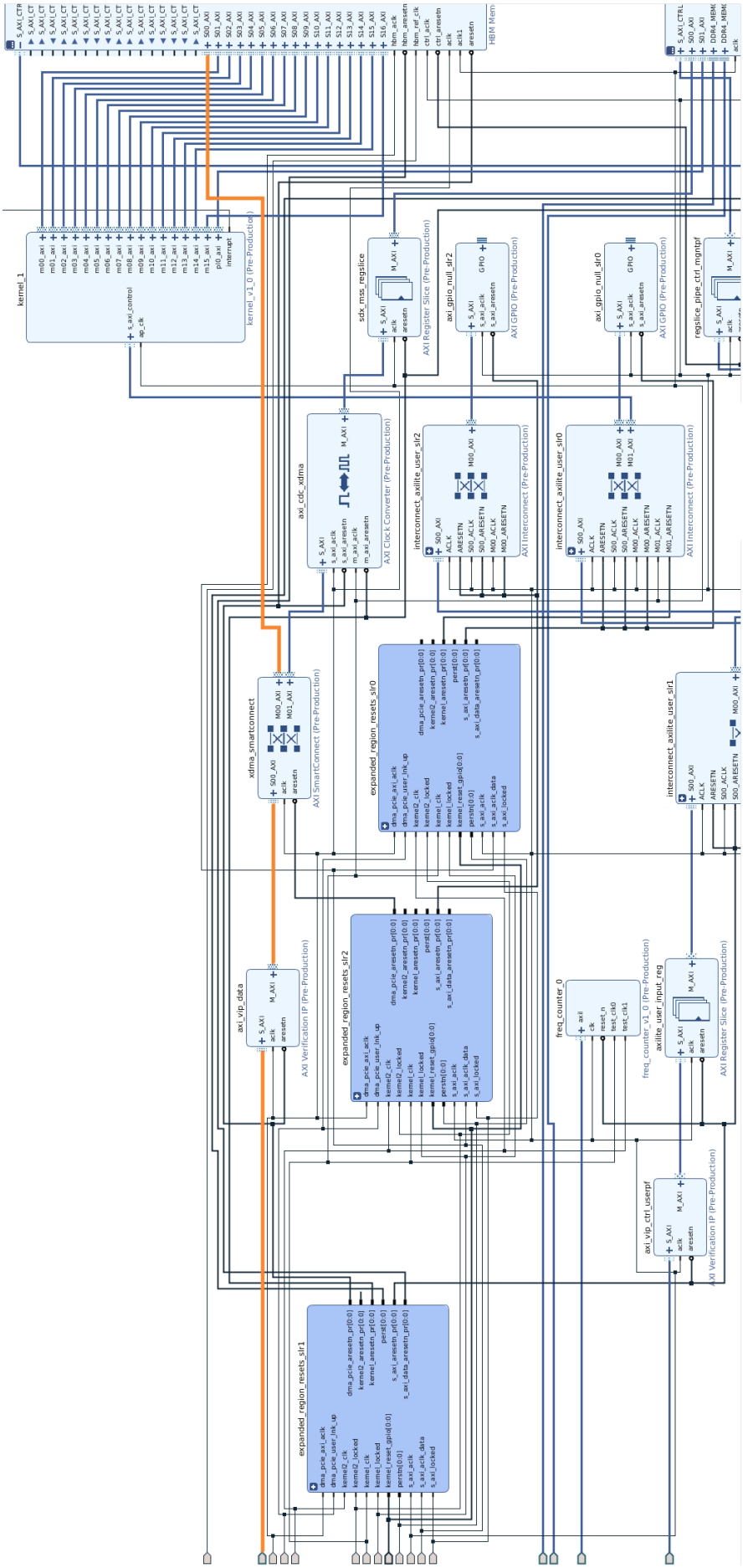

Рис. 2. Графічне представлення динамічної області проєкту та зв'язку трьох складових частин SLR з HBM

Застосування масштабування та розпаралелювання на верхньому рівні, а також конвеєризації та параметризації на рівні RTL дозволяє ефективно розгортати такі FPGA системи на базі ресурсів комерційних хмарних інфраструктур і дата центів.

Таким чином, динаміка зміни середовищ розроблення повинна братися до уваги під час планування проєктів FPGA для зниження ризиків зміни програмного та апаратного оточення. Програмування рішень на льоту істотно спрощує процес побудови FPGA як сервіс, а також розробку та перевірку роботи компонентів систем штучного інтелекту.

Застосування масштабування та розпаралелювання на верхньому рівні, а також конвеєризації та параметризації на рівні RTL дозволяє ефективно розгортати такі FPGA системи на базі ресурсів комерційних хмарних інфраструктур і дата центів.

Проведено експериментальне дослідження парадигми неперервного перепрограмування програмовної логіки, яке дозволило сформулювати елементи нової послідовності створення проєктів в якості сервісу для хмарних інфраструктур і дата центрів.

Використання запропонованої послідовності для побудови масштабованих конвеєризованих систем дозволяє знизити трудовитрати та тривалість процесу розробки подібних проектів для високопродуктивних FPGA. Виконання перенесення частини компонентів проекту з однієї області SLR до іншої дозволила підвищити частоту основного тактування системи з 200 МГц до 230 МГц.

Практичне застосування запропонованого методу дозволяє знизити трудовитрати на модифікацію проєкту при переході на нову версію середовища розробки або при безперервному довизначенні вимог до такої системи.

Напрямком подальших досліджень може бути посилення методологічної бази для опису складових параметрів таких систем для надання аналітичної можливості виконання чисельної оцінки характеристик на основі пріоритетних параметрів.

Внесок авторів: формулювання завдань дослідження – **В. О. Куланов, А. Є. Перепелицин**; огляд та аналіз інформаційних джерел – **В. О. Куланов**, **А. Є. Перепелицин**; аналіз рішень в складі хмарних сервісів – **В. О. Куланов**; побудова прототипу – **А. Є. Перепелицин**; аналіз отриманих результатів – **В. О. Куланов, А. Є. Перепелицин**; формулювання висновків – **В. О. Куланов, А. Є. Перепелицин**.

Усі автори прочитали та погодилися з опублікованою версією рукопису.

Література

1. Kulanov, V. Parameterized IP Infrastructures for Fault-Tolerant FPGA-Based Systems: Development, Assessment, Case-Study [Text] / V. Kulanov, V. Kharchenko, A. Perepelitsyn // Proceedings of IEEE East-West Design & Test Symposium. – 2010. – P. 452–455. DOI: 10.1109/EWDTS.2010.5742075.

2. Перепелицин, А. Є. Застосування параметризовних IP інфраструктур для розробки вбудованих відмовостійких систем на ПЛІС [Текст] / А. Є. Перепелицин // Радіоелектронні і комп'ютерні системи. – 2016. – № 5 (79). – С. 104–112.

3. Perepelitsyn, A. Technologies of FPGA-based projects Development Under Ever-changing Conditions, Platform Constraints, and Time-to-Market Pressure [Text] / A. Perepelitsyn, V. Kulanov // Proceedings 2022 IEEE 12th International Conference on Dependable Systems, Services and Technologies, DESSERT 2022. – 2022. – 5 p. DOI: 10.1109/DESSERT58054.2022.10018828.

4. 7 Series DSP48E1 Slice. User Guide, Xilinx, UG479 (v1.10) [Online]. – Available at: https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1. – 27.03.2018.

5. UltraScale Architecture Memory Resources. User Guide, Xilinx, UG573 (v1.10) [Online]. – Available at: https://docs.xilinx.com/v/u/en-US/ug573-ultrascale-memory-resources. – 4.02.2019.

6. Vivado Design Suite Properties Reference Guide, Xilinx, UG912 (v2022.1) [Online]. – Available at: https://docs.xilinx.com/r/2022.1-English/ug912-vivado-properties/USER_CROSSING_SLR. – 28.02.2023.

7. Vitis HLS User Guide, Xilinx, UG1399 (v2021.1) [Online]. – Available at: https://docs.xilinx.com/r/en-US/ug1399-vitis-hls. – 5.08.2021.

8. Technological Stack for Implementation of AI as a Service based on Hardware Accelerators [Text] / A. Perepelitsyn, H. Fesenko, Y. Kasapien, & V. Kharchenko // Proceedings 2022 IEEE 12th International Conference on Dependable Systems, Services and Technologies, DESSERT 2022, – 2022. – 5 p. DOI: 10.1109/DESSERT58054.2022.10018615.

9. Zynq DPU Product Guide, Xilinx, PG338 (v3.3) [Online]. – Available at: https://docs.xilinx.com/r/3.3-English/pg338-dpu. – 28.02.2023.

10. Perepelitsyn, A. FPGA as a Service Solutions Development Strategy [Text] / A. Perepelitsyn, I. Zarizenko, V. Kulanov // Proceedings 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies, DESSERT 2020. – 2020. – P. 376–380. DOI: 10.1109/DESSERT50317.2020.9125017

11. Method of QoS evaluation of FPGA as a service [Text] /A. Perepelitsyn, V. Kulanov, & I. Zarizenko // Radioelectronic and Computer Systems. – 2022. – No. 4. – P. 153–160. DOI: 10.32620/reks.2022.4.12.

12. Alveo U50 Data Center Accelerator Card Data Sheet, Xilinx, DS965 (v1.7.1) [Online]. – Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/data_sheets/ds965-u50.pdf. – 19.05.2021.

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

13. Alveo Product Selection Guide, Data Center Accelerator Cards, Xilinx. – Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/selection-guides/alveo-product-selection-guide.pdf. – 28.02.2023.

14. UltraFast Design Methodology Guide for the Vivado Design Siute, Xilinx, UG949 (v2019.2) [Online]. – Available at: https://docs.xilinx.com/v/u/2019.2-English/ug949-vivado-design-methodology. – 6.12.2019.

15. Alveo U280 Data Center Accelerator Card User Guide, Xilinx, UG1314 [Online]. – Available at: sandycast.com/support/documentation/boards_and_ kits/accelerator-cards/ug1314-u280-reconfig-accel.pdf. – 27.02.2020.

16. Vitis Unified Software Platform Documentation: Application Acceleration Development, Xilinx, UG1393 (v2019.2) [Online]. – Available at: https://docs.xilinx.com/r/en-US/ug1393-vitis-application-acceleration. – 28.02.2020.

17. Перепелицин, А. Є. Технології реалізації штучного інтелекту як сервісу на основі апаратних прискорювачів [Текст] / А. Є. Перепелицин, Є. В. Касапєн, Г. В. Фесенко, В. С. Харченко // Авіаційно-космічна техніка і технологія. – 2022. – № 6. – С. 57–65. DOI: 10.32620/aktt.2022.6.07.

References

1. Kulanov, V., Kharchenko, V., & Perepelitsyn, A. Parameterized IP Infrastructures for Fault-Tolerant FPGA-Based Systems: Development, Assessment, Case-Study. *Proceedings of IEEE East-West Design & Test Symposium*, 2010, pp. 452–455. DOI: 10.1109/EWDTS.2010.5742075.

2. Perepelitsyn, A. E. Ispol'zovaniye paramet-riziruyemykh IP infrastruktur dlya razrabotki vstroyennykh otkazoustoychivykh sistem na PLIS [Usage of parametrizable IP infrastructures for FPGA-based fault-tolerant onboard systems development]. *Radioelektronni i komp'uterni sistemi – Radioelectronic and computer systems*, 2016, vol. 5, pp. 104–112.

3. Perepelitsyn, A., & Kulanov, V. Technologies of FPGA-based projects Development Under Ever-changing Conditions, Platform Constraints, and Time-to-Market Pressure. *Proceedings 2022 IEEE 12th International Conference on Dependable Systems, Services and Technologies, DESSERT 2022*, 2022. 5 p. DOI: 10.1109/DESSERT58054.2022.10018828.

4. *7 Series DSP48E1 Slice. User Guide, Xilinx, UG479*. Available at: https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1. (accessed March 27, 2018).

5. *UltraScale Architecture Memory Resources. User Guide, Xilinx, UG573 (v1.10)*. Available at: https://docs.xilinx.com/v/u/en-US/ug573-ultrascale-memory-resources. (accessed February 4, 2019).

6. *Vivado Design Suite Properties Reference Guide, Xilinx, UG912 (v2022.1)*. Available at: https://docs.xilinx.com/r/2022.1-English/ug912-vivado-properties/USER_CROSSING_SLR. (accessed February 28, 2023).

7. *Vitis HLS User Guide, Xilinx, UG1399 (v2021.1)*. Available at: https://docs.xilinx.com/r/en-US/ug1399-vitis-hls. (accessed August 5, 2021).

8. Perepelitsyn, A., Fesenko, H., Kasapien, Y., & Kharchenko, V. Technological Stack for Implementation of AI as a Service based on Hardware Accelerators. *Proceedings 2022 IEEE 12th International Conference on Dependable Systems, Services and Technologies, DESSERT 2022*, 2022. 5 p. DOI: 10.1109/DESSERT58054.2022.10018615.

9. *Zynq DPU Product Guide, Xilinx, PG338 (v3.3)*. Available at: https://docs.xilinx.com/r/3.3-English/pg338-dpu. (accessed February 28, 2023).

10. Perepelitsyn, A., Zarizenko, I., & Kulanov, V. FPGA as a Service Solutions Development Strategy. *Proceedings 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies, DESSERT 2020*, 2020, pp. 376-380. DOI: 10.1109/DESSERT50317.2020.9125017.

11. Perepelitsyn, A., Kulanov, V., & Zarizenko, I. Method of QoS evaluation of FPGA as a service. *Radioelectronic and Computer Systems*, 2022, no. 4, pp. 153–160. DOI: 10.32620/reks.2022.4.12.

12. *Alveo U50 Data Center Accelerator Card Data Sheet, Xilinx*, DS965 (v1.7.1). Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/data_sheets/ds965-u50.pdf. (accessed May 19, 2021).

13. *Alveo Product Selection Guide, Data Center Accelerator Cards, Xilinx*. Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/selection-guides/alveo-product-selection-guide.pdf. (accessed February 28, 2023).

14. *UltraFast Design Methodology Guide for the Vivado Design Siute, Xilinx, UG949 (v2019.2).* Available at: https://docs.xilinx.com/v/u/2019.2-English/ug949-vivado-design-methodology. (accessed December 6, 2019).

15. *Alveo U280 Data Center Accelerator Card User Guide, Xilinx, UG1314 (v1.3).* Available at: https://www.sandycast.com/support/documentation/boards_and_kits/accelerator-cards/ug1314-u280-reconfig-accel.pdf. (accessed February 27, 2020).

16. *Vitis Unified Software Platform Documentation: Application Acceleration Development, Xilinx, UG1393 (v2019.2).* Available at: https://docs.xilinx.com/r/en-US/ug1393-vitis-application-acceleration. (accessed February 28, 2020).

17. Perepelitsyn, A., Kasapien, Y., Fesenko, H., & Kharchenko, V. Technologies for Implementing of Artificial Intelligence as a Service based on Hardware Accelerators. *Aviacijno-kosmicna tehnika i tehnologia – Aerospace technic and technology*, 2022, no. 6, pp. 57-65. DOI: 10.32620/aktt.2022.6.07.

Надійшла до редакції 10.08.2023, розглянута на редколегії 20.09.2023

METHOD OF CREATION AND DEPLOYMENT OF FPGA PROJECTS RESISTANT TO CHANGE OF REQUIREMENTS AND DEVELOPMENT ENVIRONMENTS FOR CLOUD INFRASTRUCTURES

Vitaliy Kulanov, Artem Perepelitsyn

The subject of study in this article is the modern technologies of programmable logic devices, the history of product changes of leading manufacturers, including development environments, and project optimization processes. The **goal** is to improve modern methods, technologies, and software tools for the development and integration of FPGA-as-a-Service in the form of services in cloud infrastructures, data centers, and on-board high-performance systems, with taking into account the ever-changing conditions and requirements, and the constraints of platforms. **Task**: to analyze the history of product changes of the leading vendors and manufacturers of programmable logic devices; analyze the dynamics of changes in the functionality of project development environments for FPGA-based systems, with taking into account the existing requirements and restrictions from vendors of software and hardware platforms and components; perform a comparative analysis of modern development boards and accelerator cards for the prototyping and testing of projects based on the chips of programmable logic; analyze the use of an existing services and solutions based on FPGA technologies as part of cloud services from modern cloud providers; propose practical steps for the development of systems based on FPGA resistant to project requirements change; propose the sequence for the development and optimization of high-performance systems with their implementation based on FPGA; and to provide practical example of the use of the proposed method. According to the tasks, the following **results** were obtained. The history of changes in the leading companies of programmable logic manufacturers, as well as changes in versions of the development environments and the products of one of the largest companies, is analyzed. The use of existing FPGA-based solutions as part of cloud services is analyzed in detail. Two sequences for optimizing projects and increasing their productivity with reducing of the prototyping efforts during their creation and porting to new versions of software and hardware platforms are provided. Based on the results of the research, a prototype was developed and tested, which allowed the application of the proposed method in practice for adapting and porting the FPGA as a service project during the transfer to another version of the accelerator card. **Conclusions.** The main contribution and scientific novelty of the obtained results is that an experimental study of the paradigm of runtime reprogramming of programmable logic was performed, which made it possible to formulate the elements of a new method of creation projects as a service for cloud infrastructures, data centers, and artificial intelligence systems. A set of practical steps for the development of systems that are tolerant to changes in conditions and requirements is proposed. The application of the proposed method allows to avoid costs of project support in the case of changes in requirements.

Keywords: FPGA; PLD; FPGA as a service; cloud services; development environment; project requirements change; hardware accelerator.

Куланов Віталій Олександрович – канд. техн. наук, доц., доц. каф. комп'ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Перепелицин Артем Євгенович – канд. техн. наук, доц., доц. каф. комп'ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Vitaliy Kulanov – PhD, Associate Professor at the Computer Systems, Networks and Cybersecurity Department, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,
e-mail: v.kulanov@csn.khai.edu, ORCID: 0000-0002-9312-0735, Scopus Author ID: 54911941800.

Artem Perepelitsyn – PhD, Associate Professor at the Computer Systems, Networks and Cybersecurity Department, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,
e-mail: a.perepelitsyn@csn.khai.edu, ORCID: 0000-0002-5463-7889, Scopus Author ID: 56332607800.