УДК 519.6:512.972, 004.27

Клименко І.А., к.т.н.

КЛАСИФІКАЦІЯ ТА АРХІТЕКТУРНІ ОСОБЛИВОСТІ ПРОГРАМОВАНИХ МУЛЬТИПРОЦЕСОРНИХ СИСТЕМ-НА-КРИСТАЛІ

Національний авіаційний університет

Надано загальну інформацію щодо убудованих мультипроцесорних систем-на-кристалі на базі ПЛІС (FPGA-MPSoC). Виконано всебічний аналіз архітектурних особливостей та надана широка класифікація FPGA-MPSoC. Приведено огляд останніх досліджень в області розробки FPGA-MPSoC. Представлено широкий круг таких систем з метою дослідження всіх тенденції архітектури та вирішуваних задач.

Вступ

Мультипроцесорні системи-на-кристалі (*MPSoC*, *Multiprocessor System on Chip*) належать до класу убудованих реконфігурованих мультипроцесорних систем (МПС) та очолюють найсучасніші тенденції розвитку цифрових убудованих електронних систем. Основна ознака таких систем – наявність більш ніж одного процесору. Потужні задачі для вирішення яких призначені сучасні убудовані електронні системи потребують саме мультипроцесорної архітектури для досягнення режиму функціонування в реальному часі за забезпечення інших критичних обмежень, таких як споживна потужність, фізичні розміри та вартість. *MPSoC* є рішенням для реалізації таких складних систем. Широке коло задач, таких як обслуговування мереж, мультимедіа, управління можуть досягнути значного підвищення продуктивності від застосування систем такого класу. Один з розповсюджених прикладів – мобільні телефони. Сучасні моделі мають пропонувати низьку споживану потужність та інтегрувати велику кількість функціоналу, такого як аудіо- та відеокодування, генерація графіки, доступ до мережі Інтернет. Ще приклад поширеного застосування – системи управління в реальному часі, як складними технологічними процесами так і різноманітним сучасним устаткуванням: бортовими системами, радарами, супутниковими системами, робототехнікою, біологічними системами, тощо. Убудовані електронні системи також успішно застосовуються для побудови високоінтелектуальних нейронних систем. *MPSoC* пропонують кращі можливості для рішення означеного кола задач у порівнянні з однопроцесорними убудованими електронними системами.

Традиційно основний спосіб збільшення продуктивності однопроцесорних систем було покращення характеристик системи за рахунок збільшення тактової частоти процесору. У подальшому традиційна тенденція змінилася у напрямку виконання паралельних обчислень на більш низьких тактових частотах, але з низькою споживаною потужністю [1 – 3]. Такий спосіб підвищення продуктивності в першу чергу зводиться до вдосконалення архітектури, а використання технології замовлених інтегральних схем (*ASIC*) дозволяє отримати максимально можливу продуктивність та швидкодію [4].

Сучасний рівень розвитку програмованої елементної бази обумовлює найновіші та найперспективніші тенденції в оглянутій області застосування убудованих електронних систем, якими є програмовані мультипроцесорні системи (*Soft MPSoC*, *Soft Multiprocessors*). Інші відомі назви таких систем – реконфігуровані мікропроцесорні системи *(RMPSoC, Reconfigurable Multiprocessor Systems*), мультипроцесорні системи-на-кристалі на основі програмованих логічних інтегральних схем ПЛІС (*FPGA-MPSoC*, *FPGA-based Multiprocessor System-on-Chip*), мультипроцесори на програмованому кристалі (*MPoPC*, *Multiprocessor-on-Programmable Chip*).

Програмована технологія спрощує швидке створення прототипів електронних систем та дозволяє проводити дослідження нових архітектур і технологій без проблем, пов'язаних з реалізацією *MPSoC* на *ASIC* [4]. Кількість публікацій стосовно розробки та дослідження реконфігурованих *MPSoC* за останні роки значно збільшилось як за кордоном, так і в нашій країни, що показує велику актуальність цього напрямку. За даними бібліографічної бази даних наукової та технічної інформації *Inspec* кількість публікацій за 2008-2010 роки виросла у 8-10 разів у порівнянні з 2000 роком [5], пошук здійснювався за ключовими словами «*multiprocessor*» та «*FPGA*».

До останніх років реконфігуровані *MPSoC* зазвичай презентували, як спосіб створення

прототипів систем для подальшої реалізації на *ASIC*. Сьогодні і кінцеву розробку приладу доцільно виконувати на базі ПЛІС. Ріст можливостей сучасних ПЛІС [4] дозволяє розробникам реалізувати складні мультипроцесорні системи на одному кристалі, а ведучі компанії розробники ПЛІС в свою чергу пропонують можливості застосування програмованих ядер процесорів спеціально розроблених для використання в ПЛІС, та апаратних процесорних ядер. Крім того ПЛІС устатковані убудованими блоками пам'яті, периферією та схемами зв'язку. Динамічна реконфігурація є однією з потужніших якостей *FPGA-MPSoC*, яка забезпечує гнучкість проектування та дозволяє мультипроцесорним системам адаптуватися до конкретних вимог вирішуваної задачі.

Фактори, що обумовлюють ефективність технології FPGA-MPSoC

Як вже було зазначено реалізація мультипроцесорної архітектури на *ASIC* надає високі показники продуктивності для вирішування різноманітних спеціалізованих задач. Таким чином слід визначити основний сенс реалізації мультипроцесорних систем за застосування реконфігурованих технологій ПЛІС. Головний недолік систем цього класу у порівнянні з технологією *ASIC* це зниження продуктивності, що обумовлено відмінностями використання внутрішніх ресурсів мікросхем. Розміщення логічних вентилів мікросхеми ПЛІС строго визначено і розташування логічної конфігурації розроблюваного пристрою та трасування внутрішніх з'єднань не завжди носить оптимальний характер, що відображається на швидкодії та споживаної потужності [4]. Одначе сучасні ПЛІС мають в першу чергу порівняні технічні характеристики з мікросхемами *ASIC*, та сама технологія *FPGA-MPSoC* має ряд переваг, що обумовлюють вибір саме цього способу реалізації убудованих мультипроцесорних систем. Далі наведені основні переваги технології *FPGA-MPSoC*, що обумовлюють її сучасний прогрес.

Гнучкість та реконфігурованість. Кількість програмних процесорних ядер, що можуть бути включені в систему, обмежена тільки об'ємом внутрішніх ресурсів кристалу ПЛІС. Також розробник має можливість незалежно конфігурувати кожний процесор додаючи кеш-пам'ять та додаткові обчислювальні потужності, наприклад, блок обробки даних з плаваючою комою (*FPU*, *Floating Point Unit*) та різноманітні співпроцесори для спеціалізованих обчислень [6], або змінюючи функціональність будь яких блоків. В цьому контексті важливо те, що пристрій може програмуватися та модифікуватися внутрисистемно, тобто функціональність пристрою вже убудованого в електронну систему може бути легко змінено, одноразово або багаторазово, забезпечуючи динамічну реконфігурацію системи, так кажучи «на-льоту».

Зменшення часу розробки. Процес розробки не включає безпосередньо виробництво інтегральної схеми, що значно скорочує час розробки. Під розробкою пристрою мається на увазі програмування заздалегідь підготовленого кристалу. Крім того на скорочення строків виходу на ринок готового пристрою значно впливає просте внесення змін у запрограмовану конфігурацію, а саме можливість зручної та швидкої модифікації, налагоджування та тестування, як готового пристрою цілком, так і його окремих ланцюгів, без залучування технологічного процесу виробництва мікросхем.

Низька вартість. Зараз ПЛІС масового виробництва відносно дешеві, і можуть бути застосовані як для промислового виробництва електронних пристроїв, так і в лабораторних умовах невеличкими командами розробників, в тому числі науковцями. Звичайно відсутність технологічного процесу виробництва і вільнопоширювані та широкодоступні засоби автоматизації проектування від провідних виробників ПЛІС також відображаються на ціну готового пристрою.

Масштабованість. Збільшення кількості процесорів, об'єму пам'яті та периферії можливо настільки наскільки дозволяють ресурси кристалу.

Ділі слід означити основну галузь ефективного застосування технології *FPGA-MPSoC*. Убудовані мультипроцесорні системами в контексті виконуваних ними специфічних задач підпадають під розряд нестандартного, спеціалізованого електронного устаткування. Основний показник, що визначає ефективність застосування будь якої технології це досягнення найкращої відповідності «вартість–продуктивність». Під вартістю розробки мають на увазі витрачені часові та людські ресурси й ціну виробництва. Враховуючи означені вище переваги технології *FPGA-MPSoC* можемо конкретизувати деякі випадки, коли застосування цієї технології більш переважніше і дає найкращий баланс вартості розробки і продуктивності.

– Невеликого об'єму випуску, вузькоспеціалізовані, критично важливі в відповідних галузях проекти.

– Наукові дослідження: розробка нових архітектур, ієрархій пам'яті, способів міжпроцесорної взаємодії, і таке інше.

– Швидка розробка нових класів реконфігурованих мультипроцесорів.

– Системи, що мають перспективи розростання у подальшому використанні в залежності від збільшення складності вирішуваних задач.

Для означених випадків застосування ПЛІС пропонує велику кількість переваг під час розробки убудованих електронних систем. Ці переваги включають можливість швидкого виправлення помилок, вдосконалення та модернізації (*Upgrade*) системи, зміни апаратних функцій без зміни фізичної структури системи на платі [4, 5, 7]. Сьогодні опит розробки пристроїв на базі ПЛІС вже показав, що вони мають по-перше переваги у часі виходу на ринок готових розробок, у порівнянні з *ASIC* [8 – 10], та по-друге потужні сучасні інструменти орієнтовані на розробку *FPGA* високого класу, доступні для широкого класу розробників [11–13].

На підставі вищесказаного та на основі аналізу літературних джерел, в яких обговорюють життєздатність реконфігурованих мультипроцесорних систем [6, 8, 9, 14], слід означити основні проблеми досліджуваної області, що мають бути вирішені:

– обґрунтування життєздатності та перспектив, чи можуть *FPGA-MPSoC* за рівнем продуктивності конкурувати з мультипроцесорними системами широкого застосування, зокрема з *MPSoC*-рішеннями на *ASIC*;

– підвищення продуктивності *FPGA-MPSoC*, потребує розробки та втілення нових методів, засобів, технологій, тощо;

– проектування ефективних *FPGA-MPSoC* для рішення цільових задач – потребує розробки ефективних методологій та засобів проектування.

Необхідність вирішення означених проблем обґрунтовують актуальність та доцільність наукових досліджень в галузі розробки убудованих мультипроцесорів на базі ПЛІС.

Огляд класів та основних виробників процесорних ядер на ПЛІС

Основна конфігурація архітектури *FPGA-MPSoC* це процесорне ядро. Доцільно використовувати готові, вже налагоджені мікропроцесорні ядра, що дозволяє скоротити тривалість циклу проектування, та забезпечити більш високу продуктивність, ніж за розробки деяких власних конфігурацій. Найбільш широко застосовують програмовані процесорні ядра, що випускаються відомими компаніями *Xilinx* та *Altera*.

Xilinx широко застосований бренд за розробки *FPGA-MPSoC*. Компанія пропонує три класи процесорів: програмовані процесорні ядра *MicroBlaze*, *PicoBlaze* та убудоване апаратне ядро *PowerPC*. Кількість апаратних *PowerPC* ядер обмежено сімейством та моделлю ПЛІС, компанія пропонує мікросхеми максимум з чотирма убудованими процесорними ядрами. Кількість програмованих процесорних ядер *MicroBlaze* або *PicoBlaze*, що можуть бути застосовані, обмежено тільки логічними ресурсами кристалу ПЛІС.

Процесорне ядро *MicroBlaze* це тридцятидвох розрядний *RISC* програмований процесор з гарвардською архітектурою, за якої процесор має розділені пам'ять команд і пам'ять даних [15]. Щодо продуктивності, *MicroBlaze* може формувати нову командну інструкцію в кожному такті і підтримувати таку продуктивність у більшості випадків. Для реалізації вводу/виводу застосовується загальна шина *CoreConnect On-Chip Peripheral Bus*, яка може функціонувати в режимах «*master*» або «*slave*» та забезпечувати безпосереднє підключення великої кількості периферії та *IP*-блоків як компанії *Xilinx*, так і сторонніх виробників. Кожний процесор MicroBlaze має засоби *Fast Simplex Link* (*FSL*) для реалізації ефективного з'єднання крапка-крапка (*point-to-point*) для додавання спеціально створених співпроцесорів для прискорення обчислень [6].

Мультипроцесорні системи з ядром *MicroBlaze* конфігуруються за застосування інтегрованого пакету компанії *Xilinx Embedded Development Kit* (*EDK*) [15]. Цей пакет включає програмі засоби, документацію та *IP*-ядра і застосовується для розробки убудованих систем на основі кристалів ПЛІС компанії *Xilinx* з убудованими апаратними ядрами процесора *PowerPC* та програмованими процесорами *MicroBlaze*.

Процесорне ядро *PicoBlaze* представляє собою компактний, продуктивний, економічно обґрунтований, убудований програмований процесор [16]. Це восьмирозрядний *RISC* мікроконтролер, оптимізований для всіх сімейств мікросхем *Xilinx*. Компанія безкоштовно поставляє процесорні ядра у вигляді

вихідного коду на мові *VHDL* разом із спеціальною середою для розробки, але надана ліцензія дозволяє їх використання тільки з пристроями *Xilinx*.

В якості прикладу, версія процесорного ядра *PicoBlaze KCPSM*3 займає всього 96 логічних блоків *ПЛІС Spartan*-3. Це 12,5% площини кристалу *XC3S50*, та 0,3% – кристалу *XC*3*S*5000. У типовій реалізації один блок оперативної пам'яті вміщує 1024 програмних інструкції, які автоматично завантажуються під час конфігурації процесорного ядра на кристалі. Навіть така проста конфігурація дозволяє досягати швидкості від 44 до 100 *MIPS* (мільйон команд за секунду) в залежності від сімейства ПЛІС [16].

Для проектування, розробки та комплексного налагоджування пристроїв на ПЛІС, в тому числі систем-на-кристалі компанії виробники мікросхем пропонують системи автоматизації проектування (САПР). САПР це інтегрована середа розробки, яка реалізує всі етапи створення цифрового пристрою на базі ПЛІС, включаючи розробку проекту, синтез та моделювання, трасування, завантаження в кристал, тестування та налагоджування, в тому числі внутрісистемне. Зазвичай інтегровані програмні середовища підтримують всі сімейства мікросхем відповідного виробника. Так компанія *Xilinx* поставляє свої продукти разом з САПР *Xilinx ISE* (*Integrated Synthesis Environment*) [11].

Процесорне ядро *Nios* II (*N*II) компанії *Altera* це наступний за популярністю продукт, що використовується [17, 18]. Тридцятидвохрозрядне програмоване процесорне ядро *Nios* II це еволюція попередньої шістнадцятирозрядної версії програмованого процесору *Nios*. Для взаємодії процесорів між собою та периферією на кристалі застосовується достатньо продуктивне шинне рішення *Avalon* [19].

Основні особливості шини *Avalon* – простота архітектури, відповідно чого для реалізації її логіки задіяні мінімальні ресурси кристалу. Шина повністю синхронна, не підтримує передачі даних пакетами і можливість роботи в режимі «*multi-master*». В одному такті роботи один з процесорів працює в режимі «*master*» («Головний» процесор), інші пристрої системи – в режимі «*slave*» («Підлеглий» процесор). За одну трансакцію шина *Avalon* передає один байт, слово або подвійне слово (8, 16, або 32 біта) між одним з пристроїв «*slave*» і пристроєм «*master*».

Програмований процесор *Nios* II підходить для реалізації широкого кола обчислювальних додатків, від цифрових сигнальних процесорів (*DSP*, *Digital Signal Processor*) до систем управління.

Nios II має три непараметризовані вариації. *Nios* II/*e* (*Economy*) – 6-*CPI* процесор (*CPI*, *Cycles Per Instruction*, кількість тактів, необхідних для виконання однієї інструкції), мінімалізований, неконвеєрний оптимізований за розміром, з програмно-реалізованим множенням; *Nios* II/*s* (*Standart*) – 5-ступеневий конвеєр, представляє собою баланс між розміром та функціональністю, з апаратно реалізованим множенням, кешем команд; *Nios* II/f (*Fast*) – потужний 6-ступеневий конвеєр, оптимізований для досягнення максимальної продуктивності, з динамічним розгалуженням, кешем команд та даних, опціонально ділильним пристроєм, обробкою зовнішніх переривань, опціонально пристроєм управління пам'яттю (*MMU*, *Memory Management Unit*) або пристроєм захисту пам'яті (*MPU*, *Memory Protection Unit*) [20].

Розробникам систем-на-кристалі на базі ядер *Nios* II потрібна система автоматизації проектування *EDA* (*Electronic Design Automation*) *SOPC Builder* (*System on a Programmable Chip Builder)* [12, 13]. *SOPC Builder* – спеціалізоване програмне забезпечення компанії *Altera*, призначене для автоматичного проектування, безпосередньо орієнтованого на розробку систем на кристалі, і представляє собою набір різноманітних *HDL*-модулів: контролери пам'яті, інтерфейси, периферійне устаткування, тощо, на базі яких конфігурується система-на кристалі. Модулі процесорного ядра *Nios* II та шини *Avalon* включають засоби взаємодії з програмним середовищем *SOPC Builder* для автоматичної генерації ядра системи, шини і периферійних пристроїв.

Всі спеціалізовані програмні засоби автоматичного проектування, такі як *SOPC Builder*, утиліти, додаткові модулі та *IP* вузли, в тому числі модуль *Avalon* та процесорне ядро *Nios* потребують використання системи автоматизації проектування *Qwartus* II, яку компанія *Altera* поставляє з усіма своїми продуктами. САПР *Qwartus* II, забезпечує повний цикл проектування та розробки електронних пристроїв на кристалах ПЛІС, підтримуючи всі можливості продуктів компанії.

Слід також зазначити, що завдяки загальної стандартизації всі системи автоматичного проектування, як компанії *Altera*, так і компанії *Xilinx* дозволяють конфігурувати системи на базі компонентів сторонніх розробників.

Окрім застосування готових процесорних ядер широковідомих виробників ПЛІС є можливість застосування програмованих процесорів з відкритим кодом (*open cource*). Найбільш розповсюджені серед відкритих ядер програмуємих процесорів *OpenRisc* компанії *OpenCores* [21] та *Leon* (3, 4) компанії *Gaisler* [22]. Програмовані процесорні ядра *Leon* мають більш широку функціональність, ніж процесорні ядра *MicroBlaze* та *Nios* II. *IP* ядра *Leon* реалізовані за архітектурою *SPARC* (*Scalable Processor Architecture* – масштабуєма процесорна архітектура), але, у зв'язку з цим, мають значні обмеження пов'язані з великою кількістю задіяних логічних ресурсів, що ускладнює реалізацію великої кількості обчислювальних модулів в одній ПЛІС. Порівняні характеристики оглянутих класів убудованих процесорних ядер наведені у роботах [5, 20].

Особливості архітектури систем-на-кристалі

Цільові функції та класи задач, що мають бути вирішені *FPGA-MPSoC,* визначають їх архітектуру, яка є важливим чинником загальної класифікації (рис. 1). У зв'язку з цим за даними літературних джерел зазвичай розглядають такі основні класи архітектур:

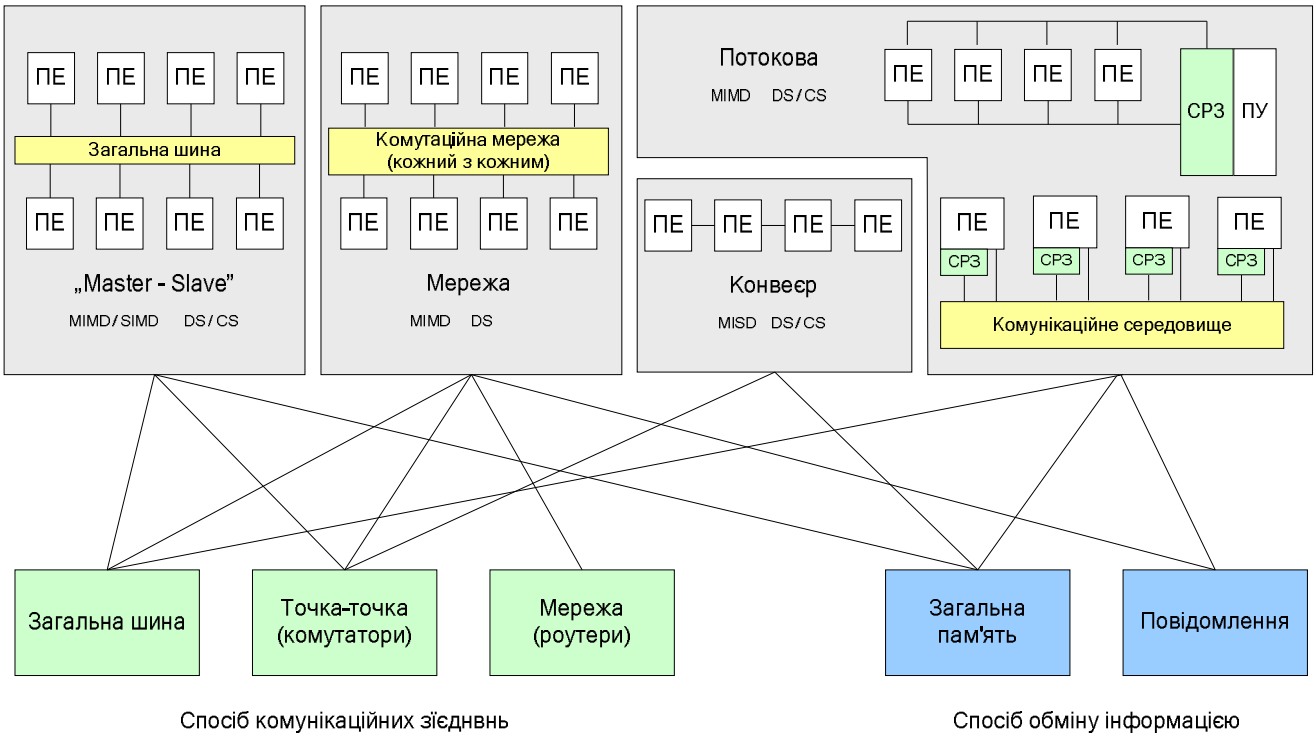


Рис. 1. Архітектурна класифікація *FPGA-MPSoC*

мультипроцесорна архітектура із загальним комунікаційним середовищем («*Master-Slave*»), конвеєрна архітектура («*Pipeline*»), мережева архітектура («*Net*»). Також можливі комбіновані варіанти, наприклад, «*Master-Slave*» + «*Pipeline*». В роботах [23 – 25] запропоновано потокову архітектуру, яка в даному огляді архітектур представлена окремим класом.

1. *Мультипроцесорна архітектура із загальною комунікаційною середою* «*Master-Slave*» («Головний» процесор – «Підлеглий» процесор) є система з різними способами управління, де один процесор – централізована система (*CS*, *centralized system*), або декілька чи всі процесори – децентралізована система (*DS*, *decentralized system*), виконують функції управління системою та контроль поведінки підлеглих процесорів. Обчислювальні вузли поєднуються загальним комунікаційним середовищем, наприклад, загальною шиною.

2. *Конвеєрна архітектура* застосовується для реалізації конвеєрних обчислень, архітектура складається з ланцюга процесорів і кожний процесор діє, як ступень конвеєра. Під час рішення потоку однотипних задач на конвеєрі значно збільшується продуктивність, якщо це адекватно задачам, що виконуються у системі.

3. *Мережева архітектура* застосовується в мультипроцесорних системах, де немає ієрархії між процесорами. Особливості комунікаційної середи в цьому випадку – забезпечення можливості спілкуватися кожного процесора з будь-яким іншим процесором, якщо це необхідно. Обчислювальні вузли поєднуються загальним комунікаційним середовищем різної архітектури та реалізують різноманітні топології – гіперкуб, метелик, дерево, тощо [26 – 28]. Ще одна назва таких систем комутуєма (*crossbar*) архітектура.

4. *Потокова архітектура* передбачає наявність процесорних вузлів та середи розподілу задач між процесорами. В потоковій архітектурі можуть бути реалізовані різні принципи управління: централізоване управління – за наявністю одного «*Master*» процесора, який здійснює загальне управління системою та пересилку команд та даних у загальну середу розподілу задач, із якої сформовані задачі поступають на процесорні елементи; децентралізоване управління передбачає розподілену середу розподілу задач, та рівноправність процесорів з точки зору вирішення задач управління. Особливістю функціонування систем з потоковою архітектурою є не послідовний перелік виконуваних команд, а наявність даних для команд. Готові команди передаються на обчислювальні блоки, виконуються, обчислювальні вузли по виконанні задачі повертають результати у середу розподілу задач, які можуть бути застосовані у подальших обчисленнях, або виводяться на пристрої вводу/виводу в якості кінцевого результату. Потокова архітектура із загальною середою розподілу задач запропонована колективом дослідників НТУУ «КПІ» під керівництвом професора Жабіна В.І. [23 – 25]. Модель архітектури реалізована на кристалі ПЛІС *Cyclon* II *Altera* [29], та за результатами моделювання отримані високі показники ефективності.

Другим важливим питанням опису архітектури є фізична реалізація способів комунікаційних з'єднань. Розглядають три загальні підходи:

1. *З'єднання крапка-крапка* (*Point-to-Point*), коли процесори поєднані безпосередньо. Перевагою є взаємозв'язок із високою пропускною здатністю, тому що немає необхідності розподіляти загальну комунікаційну середу. Але це рішення не є ефективним за розростання системи – комунікаційна середа значно збільшується у розмірі, це спричинює використання великої кількості логічних ресурсів кристалу та ускладнює загальне управління системою. Інша проблема балансування навантаження, з'єднання крапка-крапка ефективне за умови рівномірного розподілення потоків даних комунікаційною мережею та запобігання черг [26 – 29].

2. *Загальна шина* (*Bus*), традиційний підхід, який запозичений у монопроцесорних систем. Це один з найкращих механізмів комунікації процесорів, з точки зору продуктивності, здійснення управління та синхронізації, але не ефективний за застосування великої кількості процесорів. Шина є загальним системним ресурсом, але в один момент часу може здійснювати тільки одну передачу даних від джерела до приймача. Для забезпечення безконфліктного доступу до загального системного ресурсу декількох процесорів одночасно необхідні спеціальні додаткові засоби, які в загалом реалізують деяку систему черг. Таким чином за розростання системи збільшується час очікування процесорами доступу до загальної шини, що впливає на продуктивність системи в першу чергу. Окрім того додаткові засоби звичайно використовують апаратні ресурси кристалу та потребують розробки та застосування спеціальних ефективних методів або адаптації універсальних засобів обміну даними в системі, для забезпечення вимог вирішуваної задачі.

3. *Мережа* (*Network*). Один з прогресивних підходів сьогодні, носить назву мережа-на-кристалі (*NoC*, *Network-on-Chip*). В основі цього способу взаємозв'язку процесорів лежить застосування мережевих засобів зв'язку в системі-на-кристалі. Це рішення найкращим чином поєднує галузь застосування та продуктивність. Ідея складається в застосуванні спеціальних роутерів всередині кристалу для створення зв'язків між всіма ядрами системи з низькими затримками.

Також на рівні опису архітектури важливо визначити можливі способи обміну інформацією між процесорами:

1. *Загальна пам'ять* – під час обміну інформацією між двома процесорами використовуються область загальної пам'яті, куди процесор-передавач інформації має записати дані, а процесор одержувач має зчитати ці дані. В системах класу система-на-кристалі загальна пам'ять використовується досить часто, однією з причин цього є обмежена

внутрікристальна пам'ять кристалів ПЛІС, таким чином цей метод дозволяє зберігати пам'ять. До недоліків застосування даного способу по-перше слід віднести одночасні звернення до загального системного ресурсу з боку різних процесорів, що впливатиме на продуктивність системи незалежно від структури комунікаційної середи. Зазвичай системи із загальною пам'яттю застосовують загальну комунікаційну середу, але можливі також системи з загальною пам'яттю та з *NoC*-комунікаціями. Системи із загальною пам'яттю також потребують розробки спеціальних засобів для синхронізації роботи процесорів. Це одна з найважливіших проблем дослідників та розробників мультипроцесорних систем-на-кристалі на базі ПЛІС, це пов'язано з тим, що навіть широкозастосовані процесорні ядра відомих виробників не мають універсального рішення цієї проблеми [31 – 33]. Зрозуміло, що особливості фізичної реалізації механізму обміну даними через загальну пам'ять є обмін машинними словами за паралельними комунікаційними, розрядність яких відповідає розрядності даних системи. Це зумовлює ще одну проблему – значне використання ресурсів кристала ПЛІС, а також його виводів, на реалізацію каналів передачі даних [34].

2. *Обмін повідомленнями*. Обмін пакетами найбільш часто застосовується в системах з розподіленою пам'яттю [26, 27, 30, 35] та складається з обміну повідомленнями між процесорами. Для здійснення цього способу обміну необхідні спеціальні протоколи обміну пакетами, що мають бути вирішені на програмному рівні процесорних ядер. З точки зору продуктивності слід відмітити часові витрати на формування пакетів даних, послідовну передачу пакетів даних каналами зв'язку, які, окрім всього несуть значну кількість службової інформації. Але послідовні канали передачі даних значним чином зберігають апаратний ресурс та виводи кристалу ПЛІС. Такий спосіб обміну найбільш приємний для систем з мережевою архітектурою з *NoC*-комунікаціями.

Класифікація систем-на-кристалі

Традиційна класифікація *FPGA-MPSoC* стосується структури та призначення процесорних ядер що складають загальну обчислювальну структуру системи. Розрізняють два класи систем – *гетерогенні*, коли у склад системи входять різні за призначенням та структурою процесори, прискорювачі, співпроцесори, тощо, та *гомогенні*, коли всі процесори в системі ідентичні. Зазвичай проблемно-орієнтовані системи є гетерогенні. Це є найбільш розповсюджений клас серед обчислювальних систем взагалі, що базуються на ПЛІС. Причина в тому, що *FPGA-MPSoC* зазвичай реалізуються для убудованих додатків, які внутрішньо поводяться гетерогенним чином і потребують різних типів процесорів.

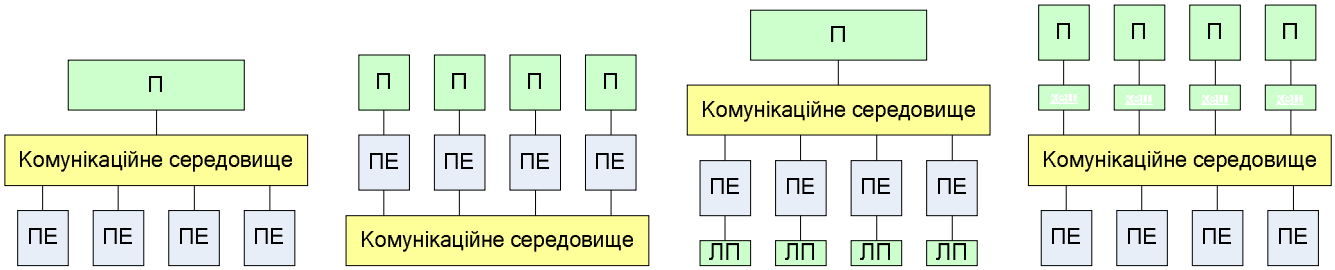
Гомогенні *FPGA-MPSoC* є звичайними для систем загального призначення, де всі процесори ідентичні. Значною перевагою таких систем є висока масштабованість як апаратного так і програмного забезпечення збільшення кількості процесорів не потребує зміни архітектури, програмне забезпечення розробляти значно простіше.

Інша важлива класифікація стосується архітектури пам'яті системи. Розглядають два класи архітектури пам'яті: *загальна пам'ять* та *розподілена пам'ять*.

В системах із загальною пам'яттю всі процесори розділяють єдиний ресурс пам'яті (рис. 2, *а*). Таким чином всі зміни що зроблені процесором в деякій області пам'яті видимі для інших процесорів в системі.

З точки зору ефективності архітектури системи із загальною пам'яттю мають обмежені можливості для масштабування по причині обмеженої пропускної здатності пам'яті. Окрім того, системи із загальною пам'яттю потребують реалізації спеціальних механізмів синхронізації міжпроцесорної взаємодії. Одним із способів розпаралелювання задач в мультипроцесорній системі є розподіл виконуваної задачі на декілька процесів. При цьому виникає потреба в реалізації таких засобів синхронізації, як монітори, семафори, прапори, бар'єри та замки, що встановлюються протягом існування зв'язку [30]. Також необхідно окремо забезпечити управління доступом до загальних системних ресурсів, до яких належить загальна пам'ять. Другим способом є організація потоків, або ниток. Найбільш популярні відомі моделі для організації багатопоточних обчислень з архітектурою загальної пам'яті це *POSIX* (*POSIX threads* (*потоки, нитi* – стандарт для реалізації потоків, механізм породження ниток) [36, 37] та *OpenMP* (*Open Multi-Processing* – відкритий стандарт для розпаралелювання програм в системах із загальною пам'яттю) [38, 39]. За застосування цих моделей в архітектурах із загальною пам'яттю різні процеси можуть легко обмінюватись інформацією через спеціальну зага-

льну змінну, одначе зазначені технології здійснюють лише організацію потоків та окремо ще потребують реалізації синхронізації, управління доступом до загальної пам'яті та засобів її захисту [37, 39].



а *б* *в* *г*

Рис. 2. Архітектурні особливості організації пам'яті в FPGA-MPSoC:
а – загальна пам'ять, *б* – розподілена пам'ять, *в, г* – засоби підвищення ефективності загальної пам'яті

В якості збільшення продуктивності архітектури із загальною пам'яттю застосовують системи із комбінованою архітектурою (рис. 2, *в*) – додавання у структуру кожного процесорного модуля локальної пам'яті (ЛП) або комунікаційної пам'яті [23]. Такі архітектури актуальні в системах побудованих на базі загального комунікаційного середовища, коли звернення до загальної пам'яті великої кількості процесорів приводить до значних затрат часу за очікування доступу та виникнення конфліктних ситуацій. Застосування локальної пам'яті зменшує частоту звернень до загальної пам'яті, що сприяє зменшенню конфліктів. Локальна пам'ять процесора може застосовуватись для зберігання програм та проміжних даних під час виконання окремих гілок паралельного алгоритму. Системні програми та дані зберігаються в основній загальній пам'яті, обмін інформацією між процесорами відбувається через загальну пам'ять, або через спеціальну комунікаційну пам'ять, що входить до складу кожного процесора.

В якості підвищення пропускної здатності загальної пам'яті пропонують [26, 34] модульну структуру пам'яті, при цьому пам'ять розбивається на модулі й доступ до окремих модулів з боку різних процесорів відбувається одночасно, така структура передбачає наявність кеш-пам'яті у структурі кожного модуля (рис. 2, *г*). Для реалізації такої пам'яті можуть бути застосовані відомі механізми розшарування пам'яті, які традиційно застосовуються для збільшення швидкості доступу до пам'яті [40, 41]. В мультипроцесорних системах пам'ять з розшаровуванням дає можливість реалізувати безліч незалежних звернень до модулів пам'яті з боку різних процесорів.

В розподілених системах (рис. 2, *б*) кожний процесор має свою особисту локальну пам'ять, таким чином один процесор не може читати ніякі директорії в пам'яті іншого процесору. Обмін даними відбувається за застосування протоколів обміну повідомленнями. Системи з розподіленою пам'яттю більш масштабовані якщо тільки комунікаційні засоби розподілені між процесорами. В якості недоліків слід зазначити витрати додаткового часу на формування пакетів даних та послідовну передачу пакетів даних каналами зв'язку. Це критично для систем з великою кількістю операцій обміну даними, наприклад, системи реального часу, на реалізаціє яких часто орієнтовані убудовані системи-на-кристалі.

Системи з розподіленою пам'яттю потребують спеціальних механізмів для підтримання прямих комунікацій між процесорами. В архітектурі із розподіленою пам'яттю комунікаційна інфраструктура призначена для з'єднання елементів обробки інформації з їх пам'яттю, та дозволу операцій обміну. Зазвичай виробники пропонують бібліотеки примітивів, що дозволяють реалізацію комунікаційних каналів. Найбільш популярний стандарт, що може бути застосований – інтерфейс обміну повідомленнями *MPI* (*Message Passing Interface*) [30, 42 – 44].

Наступна найбільш актуальна класифікація *FPGA-MPSoC* стосується адаптації системи до класів вирішуваних задач, в цьому контексті розглядають статичні і динамічні (*Run-time*) реконфігуровані системи. В основі реконфігу-

рації лежить така особливість технології ПЛІС, а як можливість багаторазового перепрограмування. Традиційно це статична реконфігурація, під якою розуміють зміну конфігурації пристрою або його фрагменту поза процесом функціонування. Така реконфігурація потребує залучання певних стадій потоку проектування ПЛІС, таких як компіляція, завантаження всієї конфігурації або її змінюваної частини, налагоджування. Динамічні реконфігуровані системи на сьогодні представляють собою найбільш передовий та актуальний клас мультипроцесорних систем на базі ПЛІС [10, 30, 44, 45 – 47]. Вони застосовують функцію динамічної реконфігурації ПЛІС для адаптації апаратної структури пристрою під специфіку виконуваної задачі. При цьому реконфігурація відбувається без залучання розробників та потоку проектування, у готовому пристрою, «на льоту». Загалом для цього реалізують декілька динамічних модулів які завантажуються через спеціальний інтерфейс в залежності від цілей додатку [26, 45]. В літературі такий тип реконфігурації називають апаратною динамічною реконфігурацією, або частковою реконфігурацією [45]. На рис. 3 зображена узагальнена схема динамічно реконфігурованої системи. Принципи динамічної реконфігурації в обчислювальних системах, в тому числі на базі мультикристальних *FPGA-MPSoC* описані в роботі [47].

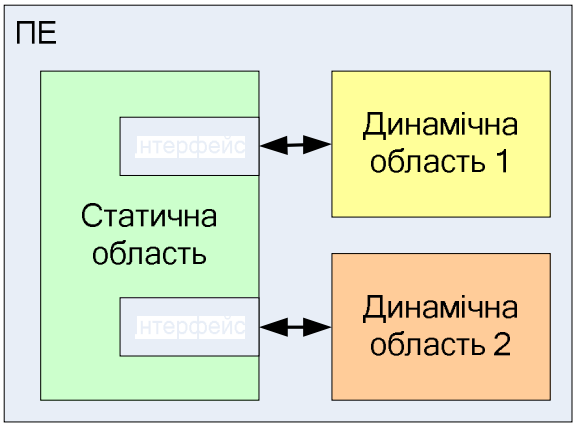


Рис. 3. Динамічна реконфігурована система

Огляд відомих гетерогенних архітектур FPGA-MPSoC

Більшість *FPGA-MPoPC* є специфічно застосовувані і залежать від додатків, що вони виконують, таким чином архітектура розробляється такою, щоб досягти найкращої продуктивності для специфічних задач. Як вже було зазначено такі системи належать до класу гетерогенних систем. До області вирішуваних задач в даному випадку можна віднести: мультимедіа, мережі, управління, біоінформатика, тощо.

Далі представимо огляд залежних від специфіки вирішуваних задач *FPGA-MPoPC*. Мережеві задачі реалізовані в роботі [8]. Запропоновані ряд рішень для передачі пакетів *IPV*4 в конвеєрній «*Master-Slave*» архітектурі. Комунікації між програмованими ядрами *MicroBlaze* реалізовані способом «крапка-крапка» за застосування портів *Fast Simplex Link* (*FSL*) ядра *MicroBlaze* компанії *Xilinx*. Різні стадії конвеєра реплікуються в просторі процесорів для збільшення пропускної здатності. Для демонстрації життєздатності цього рішення автори порівнюють запропоновану *FPGA-MPoPC* і *ASIC-MPSoC* з аналогічною функціональністю. *FPGA-MPoPC* отримують зменшення фактору 2.6X в продуктивності (нормованої за площиною кристалу). Що за результатами останніх досліджень є достатнім показником ефективності [8].

Пристрій для кодування *MPEG*-4 формату реалізовано в роботі [48]. Система має «*Master-Slave*» архітектуру на базі процесорних ядер *NIOS* II з підтримкою обміну повідомленнями і з загальною пам'яттю *SDRAM*. Система гетерогенна, для підключення загальної пам'яті команд застосовується загальна шина *Avalon* і для підключення загальної пам'яті даних методом «*plug-and-play*» («підключайся та працюй» – спосіб підключення пристроїв до системної шини без необхідності їх фізичної конфігурації та будь-якого втручання користувача для рішення конфліктних ситуацій) застосовується Гетерогенний *IP*-блок взаємозв'язку (*HIBI*, *Heterogeneous IP Block Interconnection*). Це досить просто масштабуєма обчислювальна система, що досягається за застосування спеціального паралелізму: кожне відображення ділиться на горизонтальні зрізи, кожний зріз обробляється чотирма програмними процесорами «*Master-Slave*» конфігурації.

Потокова мультипроцесорна система-на кристалі (*Stream CMP*, *Stream Chip Multiprocessor*) [8] розроблена за застосування комбінації конвеєрної архітектури і архітектури класу «*Master-Slave*» та ієрархічної структури пам'яті. Система розробляється для рішення цільової задачі, що стосується прискорення обчислення біоінформаційних додатків. Система легко масштабується з точки зору додавання процесорних ядер, обмеженням є тільки логічна ємність кристалу ПЛІС. В роботі порівнюється рішення потокових задач за

застосування універсального *CPU* (C*entral Pprocessing Unit* – центральний обробляючий пристрій), *GPU* (*Graphics Processing Unit* – графічний процесор) та реалізації мультипроцесорної системи на ПЛІС. Широко відомо, що *GPU* досягає збільшення швидкості у порядки у порівнянні з універсальним *CPU* під час рішення потокових задач, зокрема задач обробки комп'ютерної графіки. Апаратна реалізація на ПЛІС тестового потокового алгоритму також дала можливість виконати розпаралелювання задачі та отримати прискорення обчислювального процесу. Запропонована версія мікропроцесорної системи на *ПЛІС Xilinx Virtex*-5 з шістдесятьма програмованими процесорними ядрами *MicroBlaze*, що працюють на частоті 1 ГГц, з шиною класу *PCI*, зі швидкістю доступу до головної пам'яті до 2,7 Гбайт/с. За реалізації тестового алгоритму отримані збільшення продуктивності в 41,7 рази швидше, ніж процесор 1.2 ГГц *Sun Niagara*, в 16.5 разів швидше ніж процесор 2.4 ГГц *AMD Opteron* 250 і в 12.3 разів швидше ніж процесор 2.4 ГГц *Intel Xeon.*

Ще одне рішення [49] представлене для автомобільної промисловості – система асистент водія призначена для виявлення і попередження зіткнень. Система працює в режимі реального часу, при цьому критично важлива швидкість передачі та обробки інформації. Мультипроцесорна система запропонована на основі процесорних ядер *Nios* II. Архітектура побудована за конвеєрним принципом і складається з ланцюга з п'ятнадцяти процесорів, в якості способу з'єднання реалізований зв'язок «кратка-крапка». В системі реалізована розподілена пам'ять та кожний процесор виконує свою задачу, замість того, щоб розпаралелювати загальні задачі між процесорами. Виявилось що в системі для рішення задач реального часу немає сенсу застосовувати загальну шину з тієї причини, що час виконання задачі значно менше ніж час доступу до загальної шини для обміну даними, таким чином загальна шина стає так званим «вузьким горлом». Застосування локальної пам'яті для кожного процесора сприяло значному зменшенню затримок в системі. Задача управління розкладається на незалежні паралельні процедури з ціллю реалізації на реконфігурованій платформі. Таким чином визначені незалежні задачі відображаються на мультипроцесорну архітектуру і обробляються незалежними процесорами. Система застосовує 50% логічних ресурсів ПЛІС *Stratix* II 2*s*60, що містить 60 000 логічних елементів.

Огляд відомих гомогенних архітектур FPGA-MPSoC

Хоча більшість *FPGA MPSoC* мають гетерогенну структуру, для досягнення вимог певного ряду задач гомогенні мультипроцесорні системи можуть стати реальною альтернативою, забезпечуючи таки переваги, як динамічне балансування навантаження та адаптація до різних класів задач [30]. Гомогенні архітектури в основному застосовуються в системах з паралелізмом за даними. Бездротові базові станції, в яких один і той самий алгоритм застосовується до декількох незалежних потоків даних; або обробка зображень, коли різні частини зображення можуть бути оброблені окремо. В роботі [30] автори застосовують кристали ПЛІС *Spartan*3 *Xilinx* для реалізації гомогенної системи з розподіленою пам'яттю.

Мультипроцесорна убудована система (*MPLEM*, *MES*, *Multiprocessor embedded system*) [50] є гомогенною системою, що складається із восьмидесяти тридцятидвохбітних процесорів та реалізована на базі сучасних ПЛІС. Автори презентують ефективність застосування запропонованого підходу для ресурсномістких задач вирішуваних молекулярними біологами, таких як розшифрування послідовності ДНК та пошук у базах даних великого об'єму.

Автори роботи [31] презентують систему із загальною шиною та загальною пам'яттю розроблену на кристалах *Cyclone Altera*. Кожний процесор має у складі локальну кеш-пам'ять команд, для синхронізації роботи процесорів застосований механізм *Mutex* від компанії *Altera*.

Гомогенна мультипроцесорна система класу *NoC* представлена в роботі [34]. Система складається з двадцяти чотирьох процесорів, розроблена на кристалах *Virtex*-4 від компанії *Xilinx* та має чотири зовнішніх банка *DDR*2 пам'яті (*DDR*2 – *double-data-rate two synchronous dynamic memory* – синхронна динамічна пам'ять з подвоєною швидкістю передачі даних). Всі процесори та пам'ять підключені до мережі за допомогою *IP Open Core* протоколу (*OCP-IP, IP Open Core Protocol*). Мережевий інтерфейс (*NI*, *Network Interface*) переводить протокол *OCP* в протокол *NoC*, а у склад кожного процесорного ядра *MicroBlaze* входить *OCP* адаптер. Мультипроцесорна сис-

тема реалізована на *Xilinx Virtex*-4 *FPGA FX*140 та використовує 90% площини кристалу.

Симетричні мультипроцесори (SMP, Symmetric Multiprocessors) більш відомі мультипроцесорні системи загального призначення. В роботі [32] досліджують можливості ПЛІС для побудови симетричних мультипроцесорів. Автори розробили функціонально закінчену *SMP* систему на ПЛІС за застосування програмованих процесорних ядер. Система має централізовану загальну пам'ять. Процесори не мають локальних кешів для запобігання проблем когеренції.

Автори роботи [33] пропонують симетричну мультипроцесорну систему на базі ПЛІС із наявністю кеш-пам'яті у складі кожного процесора, вони запропонували ефективне *HW/SW* рішення для проблем когеренції кеш-пам'яті. Система виконана на базі тридцятидвохбітних процесорів *Nios* поєднаних класичною загальною шиною *Avalon* компанії *Altera*.

Гомогенна мультипроцесорна система на ПЛІС із комутуємими з'єднаннями типу крапка-крапка та обміном даними пакетами запропонована в роботі [26]. Запропонований підхід збільшення пропускної здатності загальної пам'яті, за рахунок розділення глобальної системної пам'яті на модулі з реалізацією в кожному модулі багаторівневого кешу. Таким чином відбувається одночасний доступ до декількох модулів пам'яті з боку декількох процесорів. В роботі також досліджені різні топології комутаційних мереж з точки зору їх використання у мультипроцесорних системах на базі ПЛІС та запропонована ефективна топологія повнозв'язаного дерева, що дозволила реалізувати комутаційне з'єднання з низькими часом очікування та високою пропускною здатністю та можливістю масштабування без непродуктивного розростання комунікаційної мережі.

Операційна система із функціональністю симетричної мультипроцесорної системи презентована в роботі [51]. Автори вважають, що недоліком симетричних мультипроцесорів є відсутність операційних систем які дозволяють програмувати багатопотокові задачі, та пропонують операційну систему, призначену для застосування в системах *SMP* на базі програмованих процесорів *MicroBlaze*.

Проблеми та обмеження реалізації високопродуктивних FPGA-MPSoC

1. *Обмеження логічних ресурсів мікросхеми ПЛІС.*

Принциповими обмеженнями побудови мультипроцесорних систем на ПЛІС є кількість логічних ресурсів, зокрема об'єму убудованої пам'яті, внутрішні канали передачі даних та виводи мікросхеми. Для зберігання убудованої пам'яті широко застосовують архітектуру загальної пам'яті. В цьому випадку є можливість розподіляти пам'ять даних та пам'ять програмам між процесорами. В роботі [52] досліджується ефективність застосування загальної пам'яті команд. Але застосування загальної пам'яті в першу чергу пов'язано з проблемою продуктивності системи за збільшення кількості процесорних елементів. В роботах [23, 32, 33] розглянуті проблеми ефективності функціонування обчислювальних систем із загальною пам'яттю та запропоновані методи підвищення продуктивності за застосування локальної та комунікаційної пам'яті. Це звичайно сприятиме збільшенню загального об'єму використаних убудованих блоків пам'яті за великої кількості процесорних ядер. Для рішення проблеми збільшення загального об'єму пам'яті часто застосовують зовнішню відносно мікросхеми ПЛІС пам'ять. Але в цьому випадку об'єм зовнішньої пам'яті обмежений кількістю виводів мікросхеми ПЛІС [34].

Під час обміну даними через загальну пам'ять, як за реалізації архітектури із загальною шиною, так і за реалізації різноманітних комутаторів відбувається передавання багаторозрядних слів даних внутрішніми каналами зв'язку, що обумовлює ще одну значну проблему реалізації *MPSoC* на базі ПЛІС, яка потребує вирішення. Одне з найбільш актуальних рішень – застосування пакетної передачі [26, 30, 34, 48]. Окрім того вчені пропонують різноманітні модифікації обчислювальних засобів, які реалізують методи формування на виході пристрою послідовного коду результату [53, 54], здебільшого засновані на відомих класичних методах формування послідовних кодів [55]. Для рішення проблеми подолання обмеженої кількості контактів в багатокристальних *FPGA MPSoC* була запропонована технологія мультиплексування виводів мікросхеми *Virtual Wires* [56], яка на сьогодні є дуже розповсюдженою серед розробників цифрових систем на ПЛІС.

2. *Синхронізація роботи процесорів.*

Важливим питанням в мультипроцесорних системах із загальною пам'яттю є синхронізація роботи процесорів. Процесорні ядра основних виробників ПЛІС зазвичай реалізують

сучасні механізми синхронізації. Наприклад, механізм для розподілу пам'яті *Mutex Core* (*Mutex* – взаємновиключення, спеціальний об'єкт синхронізації для здійснення взаємодії потоків даних) реалізований компанією *Altera. Altera* презентує *Mutex Core* як модуль для застосування в мультипроцесорних середовищах на бізі процесорних ядер *Nios* II з інтерфейсом *Avalon* для координації доступу до загальних системних ресурсів. Автори роботи [31] дослідили застосування даних продуктів для побудови мультипроцесорної системи та показали, що засоби *Mutex Core* достатньо ефективно та без помилок виконують своє призначення, надаючи процесорам доступ до загального системного ресурсу, але виявились значні витрати часу протягом кожного зв'язку, що критично в системах з великою кількістю процесорних ядер. Означені часові затримки вже значно впливають на продуктивність системи, яка містить тільки чотири процесорі ядра. Засоби *Altera Mutex Core* і *Xilinx Mutex* для продуктів обох виробників ПЛІС досліджені в роботах [32, 33, 57] із загальним висновком, що до необхідності їх вдосконалення.

На фоні недоліків механізмів синхронізації провідних виробників ПЛІС вчені пропонують так звані методи *ad-hoc* (спеціальні методи) для рішення проблем синхронізації. Так відомий підхід глобальної асинхронних та локально синхронних систем (*GALS*, *Global Asynchronous, Local Synchronous*) був запозичений для реалізації синхронізації в мережі-на-кристалі в роботі [34]. Традиційно [58] GALS розділяє систему на невеликі локальні синхронні домени, які інтегруються у загальну систему взаємодіють один з одним асинхронно через загальну комунікаційну мережу. Автори роботи показали, що підхід *FPGA-NoC* надає широкі можливості для застосування технології *GALS* з метою здійснення синхронізації в системах даного класу.

Автори роботи [32] презентують *IP* апаратне рішення *HW Mutex (Hardware Mutex)* для симетричних мультипроцесорних систем на базі продуктів компанії *Xilinx* –програмованих процесорів *MicroBlaze* та загальної системної шини *On-Chip Peripheral Bus* (*OPB*). Для синхронізації процесів була застосоване операційна система *Xilkernel Xilinx,* що призначена для убудованих процесорів, в тому числі процесорів реального часу, на базі процесорних ядер компанії *Xilinx*. Операційна система (ОС) реалізує механізм *POSIX threads* для планування та синхронізації. Тестування системи показало, що механізми синхронізації, що пропонує ОС *Xilkernel* ефективні для процесів запущених на одному процесорі і не підходять для синхронізації процесів запущених на різних процесорах. Для вирішення проблеми синхронізації автори запропонували апаратне ядро *Hardware Mutex* , яке підключається до загальної шини і є загальним для всіх процесорів системи.

Апаратні ядра для синхронізації, що здійснюють блокування та бар'єрні функції представлені в [57]. Автори розробили та дослідили *FPGA-MPSoC* на базі компонентів компанії *Xilinx.* Були виявлені проблеми застосування IP ядра *XilinxMutex*, пов'язані з повною відсутністю будь-якого механізму когерентності пам'яті та реалізацією синхронізації тільки в межах монопроцесорних систем. Запропоновані два альтернативних рішення проблем синхронізації, це мультишинна архітектура з синхронізацією на базі IP ядер *XilinxMutex*, та архітектура з загальною шиною та запропонованими власними апаратними ядрами, що реалізуюсь декілька різних варіантів синхронізації та механізм когерентності пам'яті.

3. *Когерентність кеша.*

Когерентність кеша це ще один критичний момент в розробці мультипроцесорних системах на ПЛІС. Зазвичай програмовані процесорні ядра, в тому числі ядра провідних виробників ПЛІС, що застосовуються в таких системах, не підтримують ніяких механізмів гарантованої когерентності кеша [32, 33, 57]. Автори роботи [33] запропонували *ad-hoc HW*/*SW* (*Hardware/Software)* рішення проблеми когеренції кеш-пам'яті в симетричній мультипроцесорній системі на базі ПЛІС. Автори роботи [57] запропонували апаратні рішення для реалізації механізму когерентності кеш-пам'яті.

4. *Продуктивність, пропускна здатність та масштабованість комунікаційних з'єднань.*

В мультипроцесорних системах на базі ПЛІС є необхідність в розробці ефективних та масштабованих внутрикристальних комунікацій між процесорами в умовах розміщення все збільшуваної кількості процесорних ядер в *FPGA-MPSoC.* Гетерогенні *IP* блоки взаємозв'язку (*HIBI*, *Heterogeneous IP Block Interconnection*) є ефективним *MPSoC* рішенням, яке представлене в [48, 59] для забезпе-

чення простоти масштабування та рішення конфліктних ситуацій в системі із загальною шиною. Але за проведеним оглядом існуючих розробок здається, що мережа на кристалі на базі ПЛІС (*FPGA NoC*) є кращим рішенням для взаємозв'язку процесорів [35, 60]. Такий підхід є ефективним у порівнянні з підходом загальної шини, коли система складається з великої кількості процесорних ядер і великого об'єму даних, які мають ще збільшуватись під вимоги вирішуваних задач [27, 61]. В роботі [10] представлені дослідження ефективності архітектури комунікаційного середовища. Складні *FPGA-MPNoC* представлені у роботі [34, 62]. Мультиступеневе рішення взаємозв'язку (*MIN*, *Multistage Interconnection Network*) представлене в [28]. В роботі [26] досліджені різні топології комутаційних мереж з точки зору їх використання у мультипроцесорних системах на базі ПЛІС та запропонована ефективна топологія повнозв'язаного дерева, з низькими часом очікування, високою пропускною здатністю та можливістю масштабування. В якості подолання даного недоліків САПР провідних виробників, відносно розробки мереж на кристалі автори роботи [27] презентують універсальну мережу на кристалі з перехресним комутатором (*Generic Cross-bar Network-on-Chip*) для *FPGA-MPSoC* і рекомендують що виробники ПЛІС чипів мають інтегрувати такі технології в свої стандартні потоки проектування. Автори роботи [46] презентують детальне дослідження різних реконфігурованих комунікаційних інфраструктур націлених на застосування в *FPGA-MPSoC*. Вони прийшли до висновку, що жодна з існуючих моделей не виконує поставлений цілей досягнення високої продуктивності функціонування *FPGA-MPSoC* та запропонували нову модель мережі на кристалі *Starwheels Network on Chip*, яка є поєднанням технологій пакетної комутації з комутаційним протоколом.

5. *Розробка паралельного програмного забезпечення*

Інша важлива область досліджень це розробка ефективного паралельного програмного забезпечення. Для максимального використання мультипроцесорів важливо розвивати паралельні обчислення. Для розпаралелювання цільової задачі, що має виконуватись на спеціалізованій архітектурі можна застосовувати відомі технології, такі як *MPI* [30, 38, 39], *OpenMP* [42, 43] та інші. Звичайно в цьому випадку необхідна участь професійних паралельних програмістів. Для ефективного використання широким колом користувачів була б доцільно розробка спеціальних інструментів для автоматичного розпаралелювання цільових задач [60, 46, 51, 63 – 65]. Авторами робіт [23 – 25] запропоноване спеціальне середовище розподілу задач в потоковій мультипроцесорній системі.

Методологія розробки

Два основні шляхи розробки убудованих мультипроцесорних систем на базі ПЛІС це ручне проектування (*Hand-Tuned Design* – вручну налагоджуване проектування), за застосування уніфікованого потоку проектування (*Design Flow*), який пропонують компанії виробники ПЛІС разом зі своїми САПР, та автоматичне проектування (*Automatic design*), за застосування автоматичного відображення алгоритму рішення задачі на архітектуру.

Вручну налагоджуване проектування це найбільш розповсюджений метод проектування за застосування стандартних потоків проектування компаній виробників ПЛІС. Дві найбільш відомі з них пропонують засоби *EDK* від компанії *Xilinx* та *SOPC Builder* від компанії *Altera*. Основні переваги застосування потоку проектування в наступному:

– простота спеціалізованих інструментів;

– наявність великої кількості бібліотек стандартизованих модулів, як постачальників програмного та апаратного забезпечення, так і сторонніх виробників;

– обізнаність розробників електронних пристроїв на ПЛІС різної степені кваліфікації зі спеціальними інструментами та потоком проектування.

Проблема застосування ручного проектування в тому, що складно дослідити та удосконалити весь простір проекту цілком для отримання найкращої можливої архітектури для конкретної задачі. Цей метод може бути гарним вибором для побудови невеликих систем, де відомо, як має бути розроблена архітектура.

Іншою важливою проблемою є те, що за застосування потоків проектування компаній виробників ПЛІС неможливо реалізувати всі потреби проектування мультипроцесорних систем стосовно збільшення їх продуктивності та адаптації до вирішуваних задач. Тобто стандартні потоки проектування обмежують в першу чергу таку важливу перевагу ПЛІС, як *гнучкість проектування*. Функціональність систем автоматизованої розробки, запропонованих бібліотечних модулів, програмованих процесорних ядер, інтерфейсів

зумовлена виробниками ПЛІС, окрім того інструменти для здійснення потоку проектування спочатку створені для розробки монопроцесорних систем, і мають певний ряд недоліків, коли мова йде про високопродуктивні мультипроцесорні системи. Ці проблеми були детально оговорені вище, але як підсумок наведемо найбільш важливі недоліки.

1. *Обмеження архітектури для реалізації міжпроцесорних комунікацій.* Основний проектний потік тільки дозволяє реалізувати загальну шину, та з'єднання типу «крапка-крапка». Та складні *MPSoC* можуть вимагати більш високої пропускної здатності ніж може запропонувати шина, або можуть вимагати більш ефективного використання ресурсів мікросхеми ніж за з'єднання «крапка-крапка» [26, 27, 34, 48 – 50, 52].

2. *Відсутність ефективних механізмів сумісного застосування ресурсів.* Як було вже зазначено засоби синхронізації від провідних виробників ПЛІС виявляються неефективним як метод синхронізації для високопродективних мультипроцесорних систем з великою кількістю процесорних ядер [31, 32, 34, 57].

3. *Обмежена кількість процесорних ядер.* Кількість ядер, що можуть бути убудовані в один дизайн обмежені виробниками бібліотек, зокрема відсутністю в них ефективних механізмів синхронізації та когерентності для великої кількості процесорних ядер. Дослідники намагаються вирішити ці обмеження шляхом створення *ad-hoc IP* блоків та різноманітних функціональних модулів, що діють як доповнювання для вирішення певних проблем та обмежень інструментів *EDK* [31 – 34, 57, 62].

Зазначимо, що серед оглянутих у даній роботі літературних джерел потік проектування компанії *Xilinx* за застосування САПР *EDK* застосовується в роботах [8, 9, 32, 34, 46, 50, 57] та ряд мультипроцесорних систем розроблені за застосування потоку проектування *SOPC Builder* компанії *Altera* [31, 33, 48, 49, 59]. Це найбільш широко застосовані засоби ручного проектування широким колом розробників та дослідників.

Основним напрямком досліджень в галузі проектування реконфігурованих високопродуктивних мультипроцесорних систем на базі ПЛІС методом налагоджування вручну на сьогодні є подолання зазначених вище недоліків та проблем шляхом втілення нових методів підвищення ефективності функціонування мультипроцесорних систем та створення на їх основі спеціалізованих засобів що дозволять отримати більш ефективні архітектури *FPGA-MPSoC.*

Автоматичне проектування – інша не менш актуальна сьогодні тенденція проектування електронних засобів на ПЛІС, що стосується застосування автоматичних засобів синтезу для відображення алгоритму задачі на архітектуру.

Застосування методології автоматичного проектування у порівнянні з ручним проектуванням має важливі переваги, пов'язані із зменшенням строків та трудовитрат проектування високопродуктивних систем на базі ПЛІС. Окрім того дана методологія в більшій степені відповідає концепції гнучкості проектування, адаптації системи під вимоги вирішуваної задачі та динамічної реконфігурації систем на базі ПЛІС. Слід зазначити, що хоча автор не ставив задачу детального дослідження засобів динамічної реконфігурації *FPGA MPSoC*, більша кількість оглянутих в даній роботі систем належать до класу динамічно реконфігурованих і розроблені за застосування автоматичного проектування [10, 30, 60, 46], які найкращим чином втілюють ідею реконфігурації готових пристроїв «на льоту».

Висновки

Виконані огляд та дослідження сучасних мультипроцесорних систем, реалізованих за технологією система-на-кристалі та представлених в закордонних та вітчизняних наукових публікаціях. В результаті огляду спостерігається значне підвищення кількості публікацій в даній галузі за останні роки, що свідчить про актуальність тематики досліджень. На підставі огляду літератури ми можемо зробити ряд висновків, що обумовлюють актуальність та необхідність подальших досліджень в цій галузі.

Одне з найбільш важливих питань, що належить до проблеми проектування високопродуктивних МПС *FPGA MPSoC* є обмеженість апаратних ресурсів ПЛІС, зокрема обмежена кількість убудованих блоків пам'яті та кількість виводів мікросхеми. Кількість убудованих блоків пам'яті фіксовано, що більш обмежує кількість процесорів, убудованих в один проект, ніж логічні ресурси мікросхеми взагалі. Під час розробки систем із загальною пам'яттю, узгодженість кеш-пам'яті та синхронізація є критичними аспектами, тому що програмовані монопроцесори, які поставляють

виробники ПЛІС не мають стандартних рішень цих питань.

Системи класу *NoC* є слабим місцем в потоці проектування виробників мікросхем ПЛІС, це пов'язано з тим, що вони не включають цей підхід в інструменти розробки. Але ми вважаємо, що це є лише питання часу, бо ефективність мережевого підходу доказана у великої кількості літературних джерел. Іншою актуальною та дуже перспективною проблемою в галузі розробки мультипроцесорних систем є створення паралельного програмного забезпечення та здійснювання на його основі автоматизації процесу проектування систем на кристалі ПЛІС. Є певна кількість досягнень в цій галузі, ці розробки все ще не стандартизовані і не достатньо випробувані.

Динамічна реконфігурація *FPGA MPSoC* використовує таку характерну особливість ПЛІС, як багаторазова перепрограмованість або реконфігурованість, для забезпечення нового рівня свободи проектування мільтипроцесорних систем, роблячи ці системи більш гнучкими до цілей вирішуваних задач на рівні апаратного забезпечення. Розробка динамічно реконфігурованих *FPGA MPSoC* на сьогодні є найбільш перспективним напрямком розвитку високопродуктивних обчислювальних систем, покликані вирішити багато існуючих проблем в даній галузі.

Список літератури

1. Wolf W. Multiprocessor system-on-chip (MPSoC) technology / W. Wolf, A.A. Jerraya, G. Martin // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems – USA, CA, Sonoma: 2008. – Volume 27, Issue 10 – P. 1701–1713.

2. Wolf W The future of multiprocessor systems-on-chips // Proceedings of the 41st annual Design Automation Conference (DAC '04) [ACM], (USA, San Diego, CA, 7-11 June, 2004). – USA: ACM, 2004. – P. 681–685.

3. Martin G. Overview of the MPSoC design challenge // Proceedings of the 43rd Design Automation Conference (DAC '06) [ACM/IEEE], (San Francisco, CA, USA, 24-28 July, 2006). – New York, NY, USA: ACM, 2006. – P. 274–279.

4. Клименко І.А. Тенденції застосування сучасної елементної бази для побудови високопродуктивних обчислювальних систем // Проблеми інформатизації та управління: Зб.наук.пр.– К. : НАУ-друк, 2010.– Вип.1(29). – С 90–103.

5. Dorta T. Reconfigurable Multiprocessor Systems: A Review / T. Dorta, J. Jiménez, J.L. Martín, U.Bidarte, A.Astarloa // International Journal of Reconfigurable Computing [Special issue on selected papers from ReconFig International conference on reconfigurable computing and FPGAs (ReconFig 2009)] – 2010. – Volume 2010. – 11 c.

6. Жуков І.А. Апаратна концепція розв'язку нечітких слар / І.А. Жуков, І.А. Клименко // Проблеми інформатизації та управління: Зб.наук.пр.– К. : НАУ, 2011. – Вип.2 (28). – С. 50–54.

7. Gohringer D.A Taxonomy of Reconfigurable Single-/Multiprocessor Systems-on-Chip / D. Gohringer, T. Perschke, M. Hubner, J. Becker // International Journal of Reconfigurable Computing. – 2009. – Volume 2009. – P. 11.

8. Ravindran K. An FPGAbased soft multiprocessor system for IPV4 packet forwarding / K. Ravindran, N. Satish, Y. Jin, K. Keutzer // Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '05), (Tampere Hall, Tampere, Finland, 24-26 August, 2005). – 2005. – P. 487–492.

9. Karanam R.K. A stream chipmultiprocessor for bioinformatics / R.K. Karanam, A. Ravindran, A. Mukherjee, // SIGARCH Computer Architecture News. – 2008. – Volume 36, № 2 – P. 2–9.

10. Wang X. Design and implementation of a resource-efficient communication architecture for multiprocessors on FPGAs / X. Wang and S. Thota // Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig'08), (Cancun, Mexico, 3-5 December, 2008). – 2008. – P. 25–30.

11. ISE Design Suite Manuals [Електронний ресурс]. – Xilinx Inc., 2011. – Режим доступу: http://www.xilinx.com/support/documentation/dt_ise13-3.htm.

12. SOPC Builder User Guide. Version 1.0. [Електронний ресурс]. – Altera Corporation, 2010. – Режим доступу: http://www.altera.com/literature/ug/ug_sopc_builder.pdf.

13. Embedded Design Handbook Version 2.9. [Електронний ресурс]. – Altera Corporation, 2011. – Режим доступу: http://www.altera.com/literature/hb/nios2/edh_ed_handbook.pdf.

14. Ключев А.О. Программное обеспечение встроенных вычислительные систем /

А.О. Ключев, П.В. , Д.Р. Ковязина, Е.В. Петров. – СПб. : СПбГУ ИТМО, 2009. – 212 с.

15. MicroBlaze Processor Reference Guide. Embedded Development Kit EDK 13.1 [Електронний ресурс]. – Xilinx Inc., 2011. – Режим доступу: http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/mb_ref_guide.pdf.

16. PicoBlaze 8-bit Embedded Microcontroller. User Guide for Spartan-3, Spartan-6, Virtex-5 and Virtex-6 FPGAs [Електронний ресурс]. – Xilinx Inc., 2011. – Режим доступу: http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf.

17. Nios II Processor Reference Handbook. Version 11.0.0. [Електронний ресурс]. – Altera Corporation, 2011. – Режим доступу: http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf.

18. Nios II Software Developer's Handbook. Version 11.0. [Електронний ресурс]. – Altera Corporation, 2011. – Режим доступу: http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf.

19. Embedded Peripherals IP User Guide [Електронний ресурс]. – Altera Corporation, 2011. – Режим доступу: http://www.altera.com/literature/ug/ug_embedded_ip.pdf.

20. Березин В.В. Исследование времени реакции на прерывание системы на кристалле с SOFT-процессором Nios II // В.В. Березин, А.В. Зинкевич // Вестник ТОГУ. Приборостроение, метрология и информационно-измерительные приборы. – 2010. – № 2(17). – С. 67–74.

21. OpenRISC 1200 IP Core Specification [Електронний ресурс]. – OpenCores, 2011. – Режим доступу: http://opencores.org/ openrisc,or1200.

22. GRLIB IP Core User's Manual. Version 1.1.0 - B4108 [Електронний ресурс]. – Aeroflex Gaisler, 2011. – Режим доступу: http://www.gaisler.com/products/grlib/grip.pdf.

23. Жабин В.И. Архитектура вычислительных систем реального времени / В.И. Жабин. – К. : ВЕК +, 2003. – 176 с.

24. Клименко І.А. Пат. № 59112 Україна, МПК G06F 15/16 (2006.01). Обчислювальний пристрій / І.А. Клименко, В.В. Жабина; Заявник і патентовласник: Національний авіаційний університет, Київ. – Заявлений 06.08.2010. – Опубл. 10.05.2011. – Бюл. №9.

25. Клименко И.А. Обеспечение отказоустойчивости потоковых систем на однотипных вычислительных модулях / И.А. Клименко, Жабина В.В. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: ВЄК +, 2009. – №51. – С. 166 – 171.

26. Balkan A.O. Mesh-of-trees and alternative interconnection networks for single-chip parallelism / A.O. Balkan, G. Qu, U. Vishkin // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – USA, NJ: IEEE Educational Activities Department Piscataway, 2009. – Volume 17, Issue 10 – 1419–1432.

27. Bafumba-Lokilo D. Generic crossbar network on chip for FPGA MPSoCs / D. Bafumba-Lokilo, Y. Savaria, J.-P. David // Proceedings of the Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA'08), (Montréal, Canada, 22-25 June 2008). – 2008. – P. 269–272.

28. Neji B. Multistage interconnection network forMPSoC: performances study and prototyping on FPGA / B. Neji, Y. Aydi, R. Benatitallah, S. Meftaly [et al.] // Proceedings of the 3rd International Design and Test Workshop (IDT '08), (December 2008). – 2008. – P. 11–16.

29. Cyclone II Device Handbook [Електронний ресурс]. – Altera Corporation, 2008. – Режим доступу: http://www.altera.com/devices/fpga/cyclone2/cy2-index.jsp.

30. Almeida G.M. An adaptive message passing mpsoc framework / G.M. Almeida, G. Sassatelli, P. Benoit // International Journal of Reconfigurable Computing. – Volume 2009. – 2009. – 20 p.

31. Tseng C.Y. Design and Implementation of Multiprocessor System on a Chip (MPSoC) Based on FPGA / C.Y.Tseng, Y.C. Chen, // Proceedings of the International Computer Symposium (ICS '09), (Tainan, Taiwan, 16-18 December, 2009). – 2009. – P. 173–178.

32. Huerta P. Exploring FPGA Capabilities for Building Symmetric Multiprocessor Systems / P. Huerta, J. Castillo, J. I.Martınez, C. Pedraza // Proceedings of the 3rd Southern Conference on Programmable Logic (SPL '07), (Mar del Plata, Argentina, 26-28 February, 2007) – 2008. – P. 113–118.

33. Hung A. Symmetric Multiprocessing on Programmable Chips Made Easy / A. Hung, W. Bishop, A. Kennings // Proceedings of the Conference on Design, Automation and Test in Europe (DATE '05), (Munich, Germany, 7-11

March, 2005). – Volume 1. – IEEE Computer Society, 2005. – P. 240–245.

34. Wang Z. External DDR2-constrained NOC-based 24-processors MPSoC Design and Implementation on Single FPGA / Z. Wang, O. Hammami // Proceedings of the 3rd International Design and Test Workshop (IDT '08) [IEEE], (Monastir, Tunisia, 20-22 December, 2008). – IEEE Computer Society, 2008. – P. 193–197.

35. Benini L. Networks on chips: a new SoC paradigm / L. Benini G.De Micheli // Computer. – 2002. – Volume 35, № 1. – P. 70–78.

36. B. Nichols Pthreads Programming / B. Nichols, D. Buttlar, J.P. Farrell. – USA, Sebastopol, Calif:, O'Reilly & Associates, 1996. – 269 p.

37. Senouci B.; Bouchhima A.; Rousseau F.; Petrot F.; Jerraya A. Fast Prototyping of POSIX Based Applications on a Multiprocessor SoC Architecture: "Hardware-Dependent Software Oriented Approach" // Proceedings of the 17th IEEE International Workshop on Rapid System Prototyping (Chania, Crete, Greece, 14-16 June. 2006). – IEEE Computer Society, 2006. – P. 69–75.

38. The openmp api specification for parallel programming: OpenMP 3.1 API C/C++ Syntax Quick Reference Card [Електронний ресурс], 2008. – Режим доступу: http://openmp.org /wp/openmp-specifications.

39. Sato M. OpenMP: parallel programming API for shared memory multiprocessors and on-chip multiprocessors // Proceedings of the 15th International Symposium on System Synthesis (ISSS 2002), (Kyoto, Japan, 2-4 October, 2002). – IEEE Computer Society, 2002. – P. 109–111.

40. Шнитман В., Современные высокопроизводительные компьютеры / В. Шнитман. – [Електронний ресурс]. – CIT Forum, 1996. – Режим доступу: http://citforum.ru /hardware/svk/contents.shtml

41. Цилькер Б.Я. Организация ЭВМ и систем/ Б.Я. Цилькер – М: ПИТЕР, 2004. – 668 с.

42. Shanyuan G. Hardware implementation of MPI_Barrier on an FPGA cluster Field Programmable Logic and Applications / G. Shanyuan, A.G. Schmidt, R. Sass // International Conference on 19th International Conference On Field Programmable Logic And Applications (FPL 2009), (Pragua, Czech Republic, 31 August – 2 September, 2009). – 2009. – P. 12–17.

43. Bolsens I. Programming customized parallel architectures in FPGA // Parallel & Distributed Processing IPDPSW: Proceeding of the IEEE International Symposium [Workshops and Phd Forum], (19-23 April 2010). – 2010. – P. 1.

44. Saldaña M. Using Partial Reconfiguration in an Embedded Message-Passing System / M. Saldaña, A. Patel, H.J. Liu; P. Chow // Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig'10), (Cancun, Quintana Roo, Mexico, 13-15 December, 2010) – 2010 – P. 418–423.

45. Hubner M. Parallel and Flexible Multiprocessor System-on-Chip for adaptive Automotive Applications based on Xilinx Microblaze Soft-Cores / M. Hubner, K. Paulsson, J. Becker // Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2005), (Denver, Colorado, 4-8 April 2005). – 2005. – P. 149 a.

46. Gohringer D. Star-wheels network-on-chip featuring a self-adaptive mixed topology and a synergy of a circuit- and a packet-switching communication protocol / D. Gohringer, B. Liu, M. Hubner, J. Becker // Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL '09), (Prague, Czech Republic, 31 August-2 September, 2009). – IEEE, 2009. – P. 320–325.

47. Палагин А.В. Реконфигурируемые вычислительные системы / А.В. Палагин, В.Н. Опанасенко. – К.: Просвіта, 2006. – 295 с.

48. Lehtoranta O. A parallel MPEG-4 encoder for FPGA based multiprocessor SoC / O. Lehtoranta, E. Salminen, A. Kulmala, M. Hannikainen, T. D. Hamalainen // Proceedings of the International Conference on Field Programmable Logic and Applications (FPL'05), (Tampere Hall, Tampere, Finland, 24-26 August 2005). – 2005. – P. 380–385.

49. Khan J. An MPSoC architecture for the multiple target tracking application in driver assistant system / S. Niar, A. Menhaj, Y. Elhillali, and J. L. Dekeyser, // Proceedings of the 19th International Conference on Application-Specific Systems, Architectures and Processors (ASAP'08) [IEEE], (Leuven, Belgium, 2-4 July 2008) – P. 126–131.

50. Mplemenos G.G. MPLEM: an 80-processor FPGA Based Multiprocessor System / G.G. Mplemenos, I. Papaefstathiou // Proceedings of the 16th Symposium on Field-Programmable Custom Computing Machines (FCCM'08) [IEEE], (Stanford, Palo Alto, California 14-15 April 2008). – USA : IEEE Computer Society, 2008. – P. 273–274.

51. Huerta P. Operating System for Symmetric Multiprocessors on FPGA // P. Huerta, J. Castillo, C. Sanchez, J.I. Martınez // Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '08), (Cancun, Mexico, 3-5 December 2008). – 2008. – P. 157–162.

52. Kulmala A. Instruction memory architecture evaluation on multiprocessor FPGA MPEG-4 encoder / A. Kulmala, E. Salminen, T.D. Hamalainen // Proceedings of the IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS '07), (Krakow, Poland, 11-13 April 2007). – 2007. – P. 1–6.

53. Жабин В.И. Выполнение последовательностей зависимых операций в режиме совмещения / В.И.Жабин // Вісник Національного технічного університету України "КПІ". „Інформатика, управління та обчислювальна техніка". –2007. – №46. – С. 226–233.

54. Дичка И.А. Совмещение зависимых операций на уровне обработки разрядов операндов / И.А.Дичка, В.В.Жабина // Искусственный интеллект. – 2008. – №3. – С. 649–654.

55. Самофалов К.Г. Основы теории многоуровневых конвейерных вычислительных систем // К.Г. Самофалов, Г.М. Луцкий. – М: Радио и связь,1989. – 271 с.

56. Babb J. Virtual wires: overcoming pin limitations in FPGA-based logic emulators // J. Babb, R. Tessier, A.Agarwal // Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, (Napa, CA , USA, 5-7 April, 1993). – 1993. – P. 142–151.

57. Tumeo A. HW/SW methodologies for synchronization in FPGA / A. Tumeo, C. Pilato, G. Palermo, F. Ferrandi, D. Sciuto // Proceedings of the 7th International Symposium on Field-Programmable Gate Arrays (FPGA '09), [ACM SIGDA], (Monterey, California, USA, 22-24 February 2009). – USA, NY, New York : ACM, 2009. – P. 265–268.

58. Royal A. Globally Asynchronous Locally Synchronous FPGA Architectures / A. Royal, P.Y.K. Cheung // Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL 2003), (Lisbon, Portugal, 1-3 September, 2003) [Field Programmable Logic and Applications: Lecture Notes in Computer Science (LNCS)]. – Volume 2778. – Springer-Verlag New York, Inc., 2003. - P. 355–364.

59. Salminen E. HIBI-based multiprocessor soc on FPGA / E. Salminen, A. Kulmala, T.D. Hamalainen // Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'05), (Kobe, Japan, 23-26 May, 2005). – Volume 4. – 2005. – P. 3351–3354.

60. Kumar S. A network on chip architecture and design methodology / S. Kumar, A. Jantsch, M. Millberg [et al.] // Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2002), (Pittsburgh, PA, USA, 25-26 April, 2002). – USA : IEEE Computer Society 2002. – P. 117.

61. Freitas H.C. Evaluating network-on-chip for homogeneous embedded multiprocessors in FPGAs / H.C. Freitas, D.M. Colombo, F.L. Kastensmidt, P.O. A. Navaux // Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '07), (New Orleans, USA, 27-30 May, 2007). – 2007. – P. 3776–3779.

62. Lukovic S. An automated design flow for NoCbasedMPSoCs on FPGA / S. Lukovic and L. Fiorin // Proceedings of the 19th IEEE/IFIP International Symposium on Rapid System Prototyping (RSP'08), (Monterey, California, 2-5 June, 2008). – IEEE Computer Society, 2008. – P. 58–64.

63. Ishebabi H. Answer set versus integer linear programming for automatic synthesis of multiprocessor systems from real-time parallel programs / H. Ishebabi, P. Mahr, C. Bobda, M. Gebser, T. Schaub // International Journal of Reconfigurable Computing. – 2009. – Volume 2009. – 11 p.

64. Nikolov H. Efficient automated synthesis, programing, and implementation of multiprocessor platforms on FPGA chips / Nikolov, T. Stefanov, and E. Deprettere // Proceedings of the International Conference on Field Programmable Logic and Applications (FPL'06), (Madrid, Spain, August 28-30, 2006). – IEEE, 2006. – P. 1–6.

65. Kumar A. Multiprocessor systems synthesis for multiple use-cases of multiple applications on FPGA / A. Kumar, S. Fernando, Y. Ha, B. Mesman, H. Corporaal // ACM Transactions on Design Automation of Electronic Systems. – 2008. – Volume 13, № 3. – P. 1–27.