УДК 004.27

О.П. КУРГАЄВ, І.В. САВЧЕНКО

ВИКОРИСТАННЯ СТРУКТУР СУЧАСНИХ КОМП'ЮТЕРНИХ СИСТЕМ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМ ОБРОБКИ ЗНАНЬ

Анотація. *У зв'язку зі стрімким розвитком технологій програмованих логічних інтегральних схем розробка нових, більш продуктивних архітектур комп'ютерів для обробки знань залишається актуальною задачею. В роботі виконано аналіз архітектур проблемно-орієнтованих систем, а також універсальних систем. Розглянуто переваги та шляхи використання комп'ютерів із універсальною архітектурою для реалізації систем обробки знань.*
Ключові слова: *система обробки знань, проблемно-орієнтована архітектура, універсальна архітектура, процесор.*

Аннотация. *В связи со стремительным ростом технологий программируемых логических интегральных схем разработка новых, более производительных архитектур компьютеров для обработки знаний остается актуальной задачей. В работе выполнен анализ архитектур проблемно-ориентированных и универсальных компьютерных систем. Рассмотрены преимущества и пути использования компьютеров с универсальной архитектурой для реализации систем обработки знаний.*
Ключевые слова: *система обработки знаний, проблемно-ориентированная архитектура, универсальная архитектура, процессор.*

Abstract. *Due to the rapid technologies growth of programmable logic integrated circuits, developing of a new computer's architectures, which effectively support knowledge processing systems, remains a relevant problem. The analysis of task-oriented architectures and universal computer systems was conducted. The advantages and ways of using computers with a universal architecture for knowledge-processing systems implementation were considered.*
Keywords: *knowledge-processing system, task-oriented architecture, universal architecture, processor.*

1. Вступ

Системи обробки знань (СОЗ) – це практична реалізація штучного інтелекту та застосувань до [1] розробки швидких прототипів прикладних програм; керування виробничими процесами; перекладу із однієї мови на іншу; створення природно-мовних інтерфейсів для існуючих систем; реалізації експертних систем і оболонок експертних систем; створення пакетів символьних обчислень для рішення рівнянь, диференціювання і інтегрування; створення систем прийняття рішень; доказу теорем; створення та використання наукових теорій рішення проблем з різних дисциплін; створення систем міждисциплінарних досліджень тощо. Задачі, що розв'язують із використанням СОЗ, зводяться до обробки символьних даних, для підтримки якої переважно використовують комп'ютери, засновані на принципах архітектури машини Тьюрінга-Фон Неймана. Ця архітектура реалізує імперативну модель обчислення, головним недоліком якої є великий семантичний розрив між моделями представлення знань і поняттями операцій та об'єктів, що визначаються архітектурою комп'ютера [2, 3].

Перші спроби розв'язання проблем семантичного розриву та ряду інших проблем були зроблені ще у 60-х роках минулого сторіччя. Суть даних спроб зводилась до розробки проблемно-орієнтованих архітектур комп'ютерних систем із «нефоннеймановською архітектурою», яка була б придатна для прямого виконання програм, написаних на спеціальній мові високого рівня або на проміжній мові [2–5]. З багатьох причин [6] ідея широкомасштабної заміни програмних засобів апаратними виявилась невиправданою, а розвиток комп'ютерної індустрії пішов протилежним шляхом, розвиваючи програмні засоби із застосуванням більш простих, але стандартних (універсальних) апаратних архітектур. Орга-

нізацію та архітектуру сучасних комп'ютерних систем із універсальною архітектурою широко висвітлено у [7].

Стрімкий прогрес технологій програмованих логічних інтегральних схем (ПЛІС) створив передумови до реалізації проблемно-орієнтованих систем на кристалі, які можна підключити до стандартних швидкісних шинних магістралей (USB, PCI, PCIe) комп'ютерних систем. ПЛІС-технології забезпечують можливість швидкої розробки системи на кристалі, при цьому вартість такої розробки значно нижча порівняно із системами на основі замовних інтегральних схем. Розвиток ПЛІС-технологій став основою для розробки сучасних проблемно-орієнтованих комп'ютерних систем. У [2, 8, 9] наведено широкий огляд апаратних реалізацій СОЗ, у тому числі і на ПЛІС, із використанням спеціалізованих апаратних рішень, реконфігурованих засобів, генетичних алгоритмів, нечіткої логіки, нейронних мереж та паралельних алгоритмів.

У зв'язку із стрімким розвитком ПЛІС-технологій актуальною задачею є аналіз архітектур апаратної підтримки обробки знань. Отже, метою статті є систематизація й аналіз архітектур підтримки систем обробки знань.

2. Проблемно-орієнтовані архітектури

Головними причинами, що сприяли перегляду фон-неймановської архітектури, були [2]: розуміння неминучої кризи програмного забезпечення; необхідність розширення функціональності програм, які потребували високої надійності при обробці даних; висока складність розв'язуваних задач та складність забезпечення їхнього паралельного виконання; труднощі, що виникають у процесі обміну даними між процесором та основною пам'яттю; стрімкий розвиток інтегральних технологій.

Це привело до появлення цілого ряду робіт по вдосконаленню архітектури комп'ютерних систем. Одним із рішень щодо розв'язку проблеми семантичного розриву було створення тегового комп'ютера, головною особливістю якого є врахування специфіки організації даних у пам'яті, при якій кожний тип даних містить свій окремий атрибут (тег).

Інший напрямок – це архітектури комп'ютерів із інтерпретацією програм, написаних на спеціальній мові високого рівня. До цього напряму відносяться MIP, Lisp, Prolog, Рефал-комп'ютери [2–5, 10]. У галузі розробки спеціалізованих комп'ютерів також слід зазначити такі дослідження [10]: процесор апаратної підтримки мови експертних систем OPS5; відмінні від Prolog мови логічного програмування та засоби їх апаратної підтримки; спеціалізовані паралельні процесори, орієнтовані на апаратну підтримку семантичних мереж; редукційні та потокові комп'ютери.

2.1. Lisp-архітектури

Головними особливостями Lisp-комп'ютерів є швидка обробка списків та швидке виконання функціональних обчислень. Головними особливостями архітектури Lisp-процесорів є [2]: 1) використання спеціальних команд обробки списків, наприклад, команди пошуку на основі посилання або команди керування пам'яттю; 2) для пошуку даних по бінарному дереву, обчислення рекурсивних функцій застосовується стекова пам'ять типу «останній увійшов, перший вийшов», яку реалізують спеціалізованими апаратними засобами; 3) з метою зменшення загального часу доступу до даних застосовуються механізми паралельного доступу до пам'яті; 4) для збільшення швидкості доступу до пам'яті на основі посилань застосовуються спеціальні регістри, які керуються програмними засобами; 5) збільшення швидкості обробки даних забезпечується за рахунок використання тегових структур даних. Серед Lisp-комп'ютерів можна виділити комп'ютери із складним (Dorado, Delphin, Xerox, Jerico, Cadr, Explorer, Symbolics 360, ALPHA) та спрощеним набором команд (μ3L, RIDGE, Pyramid 90X).

Розробка та промислова реалізація Lisp-комп'ютерів сприяла проведенню поглиблених досліджень щодо розробки нових архітектур у напрямках [10]: оптимізації Lisp-програм при компіляції; вибору найбільш ефективної форми для представлення програм; використання схем економного виконання програм; ефективної апаратної підтримки роботи стеку при виклику функції; паралельного виконання Lisp-програм; конвертування програм у програми на функціональній мові, які добре піддаються розпаралелюванню; конвеєрного виконання програм із використанням підвищеного обсягу апаратури; вибору ефективних схем доступу до значень змінних; підвищення компактності представлення даних в оперативній пам'яті із застосуванням хеш-адресації; застосування векторів при реалізації списків; забезпечення швидкого доступу до внутрішніх елементів списку; спеціальних алгоритмів для зменшення часу роботи із списками, розміщеними у віртуальній пам'яті; алгоритмів розділення пам'яті, які зменшують час на очікування повторного використання Lisp-комірок; ефективних алгоритмів управління пам'яттю, які забезпечують роботу програм у реальному часі.

2.2. Prolog-архітектури

У 1982 р. в Японії стартував проект із назвою «комп'ютер п'ятого покоління». Головними задачами даного проекту було розв'язати проблему семантичного розриву та реалізувати надпаралельну архітектуру комп'ютера для обробки знань, засновану на парадигмі логічного програмування [2].

Головні особливості архітектури комп'ютерів цього типу націлені на збільшення швидкості виконання функцій [2]: 1) символьної обробки даних; 2) обробки невизначеностей та перебору із поверненням за рахунок прискорення управління даним процесом.

Для досягнення поставлених цілей у Prolog-комп'ютері застосовуються стекова пам'ять для управління операцією перебору з поверненням та спеціальні апаратні засоби для операції уніфікації. Всього можна виділити 4 типи стекової пам'яті [2]: 1) управляючий стек для управління послідовністю виконання Prolog-програми; 2) локальний стек для керування областю комірок із змінними; 3) стек для управління значеннями змінних; 4) глобальний стек для генерації структури даних.

На початковому етапі проекту «комп'ютер п'ятого покоління» в Токійському інституті обчислювальної техніки було реалізовано Prolog-комп'ютер PCI із послідовною обробкою даних та комп'ютер Delta з реляційною базою даних. PCI було реалізовано за принципом тегового комп'ютера, який містить стек. Комп'ютер Delta було реалізовано за принципом комп'ютера баз даних, який має спеціальний механізм для поточної обробки бінарних операцій. Наступні стадії проекту не досягли поставленої мети [2].

2.3. Причини переходу до універсальних архітектур

На початку 1990-х років комп'ютери з універсальною архітектурою почали витісняти комп'ютери із проблемно-орієнтованою архітектурою. Слід виділити декілька причин цьому [6]: розвиток інтегральних технологій сприяв значному стрибку у продуктивності комп'ютерів з універсальною архітектурою (архітектурою фон-неймана), яка у деяких випадках була вищою за запропоновані паралельні архітектури; запропоновані технології розподіленого збереження та обробки даних виявились значно ефективнішими та надійнішими у порівнянні з централізованими системами; ідея широкомасштабної заміни програмних засобів апаратними виявилась невиправданою, оскільки призвела до суттєвого ускладнення апаратних засобів; труднощі при реалізації одночасної взаємодії великої кількості процесорів, пов'язаних із різким збільшенням витрат на комунікацію між ними; вартість реалізації комп'ютерів на основі універсальної архітектури була суттєво нижчою за рахунок їх масового випуску.

3. Універсальні архітектури

Розвиток інтегральних технологій наприкінці 1970-х та на початку 1980-х років зумовив виникнення надвеликих інтегральних схем та паралельних архітектур. Це в значній мірі вплинуло на архітектуру комп'ютерних систем обробки знань. Універсальні архітектури комп'ютерних систем широко висвітлено в літературних джерелах [7, 11].

Архітектури універсальних комп'ютерів можна поділити на комп'ютери із спрощеним набором команд та комп'ютери із складним набором команд. Комп'ютери першого класу виконують команди типів: «регістр←регістр, регістр», «регістр←пам'ять», «пам'ять ←регістр». Функціональні перетворення можуть бути виконані тільки над вмістом регістрів, а результат такого перетворення записується у регістр. Комп'ютери із складним набором команд виконують команди, які мають різний формат та для свого представлення потребують різної кількості комірок пам'яті. Ускладнений набір команд потребує значного ускладнення архітектури комп'ютера, однак це дозволяє підвищити швидкість виконання програм для окремих класів задач, у тому числі задач обробки знань [12].

3.1. Архітектури із спрощеним набором команд

Дослідження [13] ефективності мов високого рівня на комп'ютерах з універсальною архітектурою (VAX, PDP-11, Motorola 68000) показали, що під час виконання звичайної програми команди присвоєння, умовного переходу та безумовного переходу, виклику процедур та повернення із процедури займають більш, ніж 90% коду всієї програми, тоді як складні команди займають не більш, ніж 3% коду програми. При цьому більш, ніж 50% операцій виконуються над скалярними даними, 80% із яких знаходяться в середині процедури чи функції (локальні). В роботі [14] наведено дослідження комп'ютера DEC-10 та встановлено, що в середньому кожна команда використовує 0,5 операнда в пам'яті та 1,4 регістри. В роботі [15] продемонстровано, що близько 98% динамічно викликаних процедур використовують менш, ніж шість аргументів, а близько 92% з них використовують менш, ніж шість скалярних змінних. Ці дослідження свідчать про невелику кількість аргументів процедур і обґрунтовують доцільність створення комп'ютерів із мінімальною кількістю команд ефективного доступу до операндів та скалярних змінних.

Дана стратегія на початку 1970-х років сприяла виникненню універсальних комп'ютерів із спрощеним набором команд, головними принципами яких є [7] виконання команди за один машинний цикл, спрощений режим адресації регістрів, великий розмір регістрового файла, фіксована довжина команди, наявність конвеєра команд. Розробка таких комп'ютерів супроводжувалась дослідженнями щодо [7] принципів апаратної реалізації регістрового файла; оптимального використання та розміщення регістрів компілятором; підвищення ефективності виконання програми вибором оптимальної кількості регістрів; оптимізації прогнозування умовних переходів; підвищення ефективності виконання програм за рахунок планування виконання команд та розподілу регістрів.

3.2. Архітектури із складним набором команд

У напрямі розробки комп'ютерів із складним набором команд проводились такі дослідження: в [16] продемонстровано, що задача генерації ефективнішого коду для комп'ютерів із складним набором команд значно важча, ніж для комп'ютерів із простим набором команд, оскільки компілятору необхідно відшукати ті випадки, коли потрібно застосувати одну складну команду замість декількох простих; в [17] продемонстровано, що розмір коду програми для комп'ютерів із складним набором команд на 80–90% менший за розмір коду програми для комп'ютерів із простим набором команд; особливості реалізації кеш-пам'яті для таких комп'ютерів розглянуто у роботі [18].

Мікропроцесор сучасних комп'ютерів із складним набором команд містить більше десяти модулів, організованих у вигляді конвеєра. У випадку ефективної організації завантаження модулів конвеєра можливе виконання декількох операцій за один процесорний такт. Ефективне завантаження конвеєра може відбуватися або апаратним забезпеченням, що входить до складу процесора (суперскалярна архітектура), або компілятором, який виділяє паралелізм із послідовної програми (архітектура з довгим командним словом) [7].

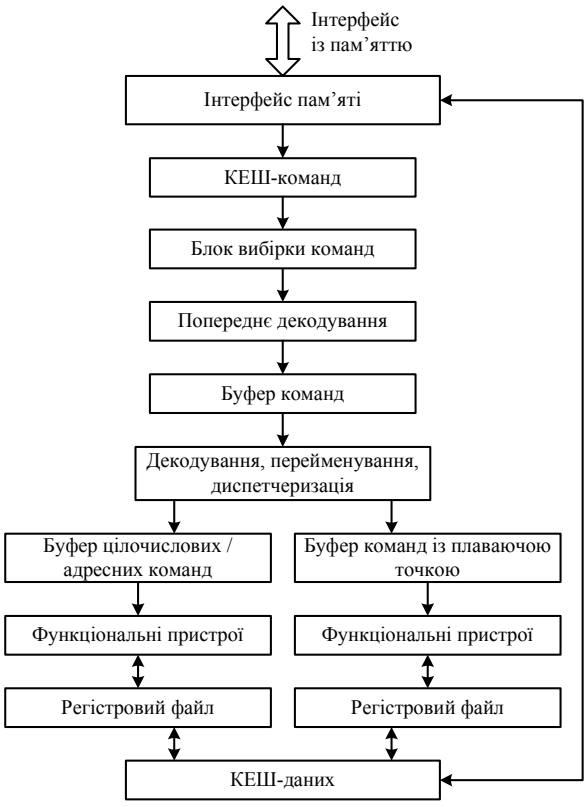

Рис. 1. Структура процесора із суперскалярною архітектурою

На рис. 1 зображено структуру типового процесора із суперскалярною архітектурою x86. До стандартних блоків процесора входять: інтерфейс пам'яті; КЕШ-команд; блок вибірки команд; блок попереднього декодування команд; буфер команд; блок декодування, перейменування та диспетчеризації; буфер цілочислових, адресних та команд із плаваючою точкою; функціональні пристрої; регістрові файли; КЕШ-даних.

Обробка команд, що містяться у пам'яті, відбувається у такій послідовності: вибірка команди, яка зчитує наступну команду у блок вибірки команд за допомогою інтерфейсу пам'яті та КЕШ-пам'яті даних; декодування вибраної команди, де визначаються коди операцій та необхідні операнди; обчислення операндів, де визначаються адреси операндів вибраної команди; вибірка із пам'яті необхідних операндів для виконання команди; виконання вибраної команди та збереження результату у регістровому файлі; збереження результату у пам'яті.

Сучасні процесори (наприклад, Intel Core i7, Intel Xeon) побудовано на основі архітектур Intel Nehalem та Intel Westmere, однак в їх основі лежить архітектура 8086 [19]. У процесорах з архітектурою Intel Nehalem головна увага зосереджується на двох принципах: зменшення енергоспоживання та підвищення швидкодії процесора за рахунок ефективнішого використання процесорних ресурсів. Такі процесори використовують 12-ступеневий конвеєр команд та виконують до чотирьох команд за один процесорний такт. Головним архітектурним рішенням щодо підвищення швидкодії стало застосування багатопотокового надходження команд до блока декодування команд.

3.3. Універсальні комп'ютери із паралельною архітектурою

На початку 1990-х років почалось широке впровадження паралельних обчислювальних комп'ютерів у науку, індустрію та бізнес. Основні принципи організації таких комп'ютерів: обчислення паралельно виконуються сукупністю взаємодіючих (до декількох тисяч), але автономних робочих станцій (комп'ютерів); у кожній автономній станції використовуються один або декілька універсальних процесорів; у кожному процесорі застосовуються конвеєризація виконання команд та паралельна обробка даних; широке використання мереж для взаємодії робочих станцій.

Розрізняють такі класи комп'ютерів із паралельною обробкою даних: масивно-паралельні системи, симетричні мультипроцесорні системи, системи із неоднорідним доступом до пам'яті, паралельні векторні системи та кластери [7, 11].

Масивно-паралельні системи (MPP) відносяться до класу систем «багато команд, багато даних». Масивно-паралельні системи складаються із центрального процесора, який формує завдання, та однорідних обчислювальних вузлів. Однорідні обчислювальні вузли містять у собі [20] один або декілька універсальних процесорів (зазвичай процесор із простим набором команд); локальну пам'ять (прямий доступ до пам'яті інших процесорних вузлів неможливий); комутаційний або мережевий адаптер із декількома каналами для організації ефективного сполучення між обчислювальними вузлами. Загальна кількість обчислювальних вузлів може складати декілька тисяч. Недоліком таких систем є те, що вона виходить з ладу при відмові центрального процесора. До таких систем належать [20]: IBM RS/6000 SP2, Intel PARAGON/ASCI Red, CRAY T3E, Hitachi SR8000, трансп'ютерні системи Parsytec.

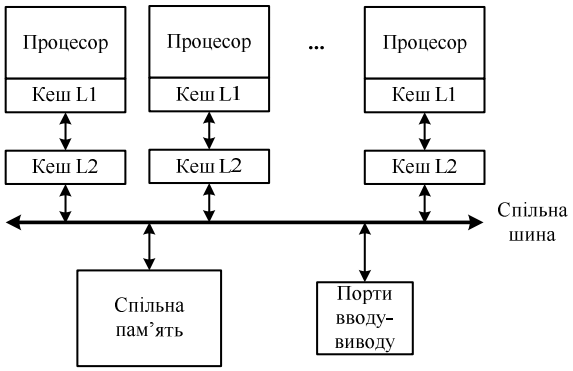

Рис. 2. Симетрична мультипроцесорна система

Симетричні мультипроцесорні системи (SMP) складаються з декількох однорідних процесорів із складним набором команд та масиву спільної пам'яті із декількох незалежних модулів пам'яті (рис. 2). Між кожним процесором та спільною пам'яттю присутня багаторівнева кеш-пам'ять (L1 та L2). Головні характеристики систем даного класу [20]: наявність одного або декількох універсальних процесорів; процесори мають доступ з однаковою швидкістю до всієї пам'яті та до пристроїв вводу-виводу, що є можливим завдяки використанню схеми комутації; процесори можуть отримати доступ до пристроїв вводу-виводу по різних каналах; процесори можуть виконувати одну і ту ж саму функцію; взаємодія процесорів контролюється операційною системою.

Перевагами таких систем є [7, 11]: надійність, оскільки відмова одного процесора не спричиняє до відмови усієї системи; можливість підвищити продуктивність системи за рахунок використання додаткових процесорів; у зв'язку з використанням спільної пам'яті значно спрощується міжпроцесорний інтерфейс, однак це спричиняє до виникнення проблеми когеренції кеш-пам'яті, яка унеможливлює одночасне поєднання більш, ніж 32 процесорів. Прикладами таких систем є [20]: Compaq AlphaServer GS140, HPN9000, HP 9000 V-class, N-class; SMP-сервери та робочі станції на базі універсальних процесорів Intel, IBM, HP, Compaq, Dell, ALR, Unisys, DG, Fujitsu тощо.

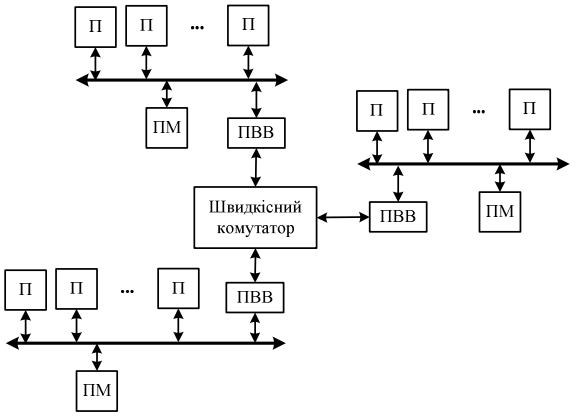

Рис. 3. Система із неоднорідним доступом до пам'яті

Системи із неоднорідним доступом до пам'яті (NUMA). В симетричних мультипроцесорних системах кількість процесорів є обмеженою (від 16 до 32) у зв'язку із використанням спільної пам'яті. Це спричинило до виникнення систем із неоднорідним доступом до пам'яті, в яких кількість процесорів може сягати декілька тисяч (рис. 3). Дані системи складаються із наборів модулів, які взаємодіють за допо-

могою швидкісного комутатора [7]. Кожний модуль містить декілька процесорів (П), які через кеш-пам'ять взаємодіють з блоком пам'яті (ПМ). Модулі взаємодіють між собою за допомогою портів. У системі підтримується єдиний адресний простір, причому доступ одного модуля до пам'яті іншого модуля забезпечується апаратно із дещо нижчою швидкістю. Недоліком даної системи є проблема когеренції кеш-пам'яті, яка усунута в системах із архітектурою cc-NUMA завдяки використанню протоколу MESI [7, 11]. Прикладами таких систем є [20]: HP 9000 V-class у SCA-конфігураціях, SGI Origin2000, Sun HPC 10000, IBM/Sequent NUMA-Q 2000, SNI RM600.

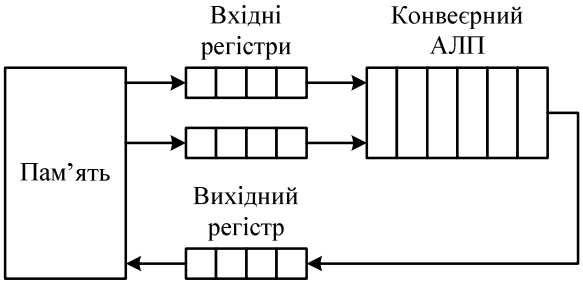

Рис. 4. Структура паралельного векторного процесора

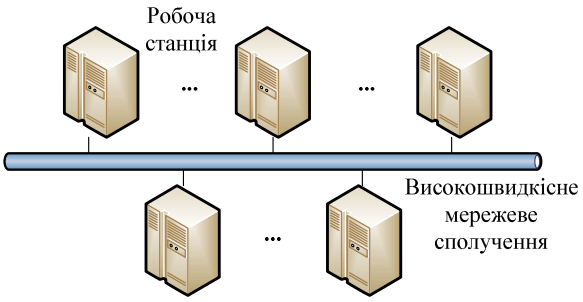

Рис. 5. Кластерна система

Паралельні векторні системи (PVP) характеризуються тим, що в них присутні спеціалізовані векторно-конвеєрні процесори, в яких передбачені команди однотипної обробки векторів незалежних даних, що ефективно виконуються на конвеєрних функціональних пристроях (рис. 4) [7]. Декілька таких процесорів (1–16) можуть працювати одночасно із спільною пам'яттю по аналогії із SMP-системами. Також ці процесори можуть бути сполучені за допомогою комутатора. Прикладами таких систем є: CRAY-1, CRAY J90/T90, CRAY SV1, CRAY X1 [20].

Кластерні системи є одним із найсучасніших різновидів комп'ютерних систем (рис. 5). Дані системи містять набір незалежних вузлів (робочих станцій або персональних комп'ютерів), на кожному із яких встановлена операційна система. Вузли кластера сполучені між собою за допомогою стандартних мережевих технологій (Fast Ethertnet, Gigabit Ethernet, Myrinet, cLAN, SCI, QsNet, MEMORY CHANNEL, InfiniBand), шинної архітектури або швидкісного комутатора. Управління вузлами здійснюється за допомогою спеціального програмного забезпечення (операційної системи), яке підтримує безперебійну роботу системи, здійснює розділення обчислювальних ресурсів при відмові одного із вузлів, підтримує єдину файлову систему. Такі системи мають переваги [7, 11]: дозволяють побудувати системи, значно потужніші за сучасні суперкомп'ютери; забезпечують можливість розширення системи за рахунок додавання нових вузлів; відмова одного вузла кластера не спричиняє до відмови усієї системи; вартість таких систем значно нижча за вартість суперкомп'ютера. Прикладами таких систем є [20]: NT-кластер у NCSA, Beowulf-кластери, IBM RS/6000 SP, KLAT2, МВС-1000М.

4. Шляхи підвищення продуктивності роботи комп'ютерних систем

Принциповими задачами підвищення продуктивності комп'ютерних систем залишаються [7]: підвищення продуктивності роботи процесора, підвищення швидкодії пам'яті та портів вводу-виводу, забезпечення швидкої комутації між обчислювальними вузлами.

Задача підвищення швидкодії пам'яті обумовлена тим, що швидкість роботи процесора на сьогодні в тисячі разів вища за швидкість роботи пам'яті. Обмін даними між процесором та пам'яттю і досі залишається вузьким місцем, що змушує розробників вводити додаткові засоби компенсації. Серед підходів до розв'язання даної задачі можна виділити [7]: підвищення кількості біт, які можна зчитувати за одне звернення до пам'яті; застосу-

вання буферних схем між процесором та пам'яттю (багаторівнева кеш-пам'ять, регістрова пам'ять); зменшення частоти доступу до пам'яті за рахунок використання ефективної кеш-пам'яті; використання шини із підвищеною швидкодією.

Задача підвищення швидкодії портів вводу-виводу обумовлена тим, що різниця між швидкістю процесора та швидкістю периферійних пристроїв є досить суттєва. Серед підходів до розв'язання даної задачі можна виділити [7]: використання буферів, схем кешування та шин обміну даними.

Задачу підвищення продуктивності процесора можна поділити на такі задачі [7]: зменшення розмірів логічних елементів та щільності їх розташування; збільшення розміру, швидкості кеш-пам'яті; внесення змін в архітектуру та організацію процесора (використання різних форм паралелізму, конвеєризації команд, додаткові набори спеціалізованих пристроїв та команд); застосування проблемно-орієнтованих архітектур. Розглянемо кожну із цих задач окремо.

4.1. Зменшення розмірів логічних елементів на кристалі та підвищення частоти тактового сигналу

Даний підхід є ефективним і успішно застосовувався впродовж багатьох років, однак на сьогодні поступово вичерпує себе.

Частота тактового сигналу процесора залежно від режиму роботи може сягати: Intel Pentium (1993р.) – 75–200 МГц, Intel Pentium 4 (2000р.) – 3,06–3,8 ГГц, Intel Core i7-900 Desktop Processor Extreme Edition (2008р.) – 2,66–3,2 ГГц, 2nd Generation Intel Core i7-2600 Processor (2011р.) – 3,4–3,8 ГГц. Це свідчить про суттєве зменшення впливу даного підходу на підвищення продуктивності процесора. Втрата актуальності даного підходу є наслідком [7] збільшення тепла, яке виділяється на одиницю площі; труднощів у реалізації процесора через наявність паразитних ємностей між провідниками; надто великого розриву щодо швидкості процесора та оперативної пам'яті.

Сучасні процесори використовують технологію виготовлення 32 нм, однак приблизно у 2020 р. розміри логічних елементів досягнуть атомарного розміру – 2 нм. На сьогодні дослідження щодо вдосконалення технології ведуться у таких напрямах: організація передачі сигналу на рівні елементарних частинок; багатократне використання електронів; вуглеводні та кременеві нанотрубки; збільшення розмірів кристалів [21].

4.2. Підвищення ефективності роботи кеш-пам'яті

На сьогоднішній день у процесорах застосовується дворівнева та трирівнева кеш-пам'ять. Наприклад, процесор корпорації Intel Pentium 4 має дворівневу кеш-пам'ять L2 – 512 КБ, Intel Core i7 -трирівневу кеш-пам'ять L3 – 12 МБ. Серед технологій, які використовуються корпорацією Intel для підвищення ефективності роботи кеш-пам'яті, є Intel Advanced Smart Cache (Intel Core2 Quad, Intel Core2 Duo, Intel Xeon 5000, Intel Xeon 3000, ін.), Intel Smart Memory Access (Intel Core2 Quad, Intel Core2 Duo, Intel Xeon 7000, Intel Xeon 3000 та ін.).

Технологія Intel Advanced Smart Cache застосовує оптимізовану багатоядерну кеш-пам'ять, що значно зменшує затримку при роботі з даними, а також підвищує продуктивність та енергозбереження за рахунок підвищення вірогідності доступу кожного з ядер процесора до даних. Оптимізація стає можливою завдяки тому, що кожне з ядер використовує спільну кеш-пам'ять другого рівня L2, при цьому дані зберігаються в одному місці і доступні для кожного з ядер у повному обсязі. Intel Advanced Smart Cache дозволяє спільне використання даних між ядрами, при цьому розмір кеш-пам'яті другого рівня L2 складає до 4 МБ, а кеш-пам'яті першого рівня L1 – до 34 КБ; для поліпшення зв'язку між L1 та L2 застосовується 256-розрядна шина, яка забезпечує пікову передачу даних 96 ГБ/с на частоті 3 ГГц [19].

Технологія Intel Smart Memory Access підвищує продуктивність системи за рахунок ефективнішого використання системної шини (до 10,7 Гб/с) та скорочення часу доступу до пам'яті. Містить нову технологію вирішення протиріч в пам'яті Memory Disambiguation, що підвищує ефективність позачергової обробки команд за рахунок вбудованих інтелектуальних ресурсів для попереднього завантаження даних для команд [19].

4.3. Внесення змін в архітектуру та організацію процесора

Даний підхід заснований на введенні та використанні нових складних наборів команд; ускладненні апаратного забезпечення; опрацюванні декількох потоків команд одночасно; використанні систем, в яких міститься багато процесорних ядер; використанні швидких каналів для обміну даними між процесорними ядрами.

Ці підходи мають певне відображення в нових технологіях: Intel Advanced Digital Media Boost, Intel Wide Dynamic Execution, Intel Hyper-Threading, Intel Quickpath, Intel Intelligent Power Capability, Intel Intelligent Power Capability ін., застосованих у процесорах (Intel Core2 Quad, Intel Core2 Duo, Intel Xeon 5000, Intel Xeon 3000 та ін.).

Технологія Intel Advanced Digital Media Boost забезпечує значне підвищення продуктивності в роботі процесора при виконанні команд типу SSE/SSE2/SSE3. Дані команди прискорюють роботу програм, які працюють з відео, телефонією, графікою, фотографіями, шифруванням, а також фінансовими, інженерними та науковими задачами. Технологія підтримує повне виконання цих 128-розрядних команд по одній за тактовий цикл, забезпечує виконання до восьми команд над числами з плаваючою точкою [19].

Технологія Intel Wide Dynamic Execution містить інноваційну технологію макрозлиття (macro-fusion), що дозволяє об'єднувати деякі розповсюджені команди x86 в одну команду [19].

Технологія Intel Hyper-Threading дає можливість одночасно опрацьовувати до двох потоків команд одному процесорному ядру. Дана технологія зменшує тривалість обчислень та забезпечує оптимальне використання кожного такту тактового сигналу [19].

Технологія Intel Quickpath забезпечує високошвидкісні канали з'єднання типу «точка-точка» між процесорними ядрами та пристроєм вводу-виводу. Кожний процесор має свою окрему пам'ять, доступ до якої відбувається через вбудований контролер пам'яті DDR3 із швидкістю до 5,32 Гб/с [19].

Технологія Intel Intelligent Power Capability призначена для зниження енергоспоживання шляхом переведення непотрібних у даний момент апаратних ресурсів у режим низького споживання енергії. Дана технологія керує режимом енергоспоживання ядер процесора, шин передачі даних, регістрових файлів [19].

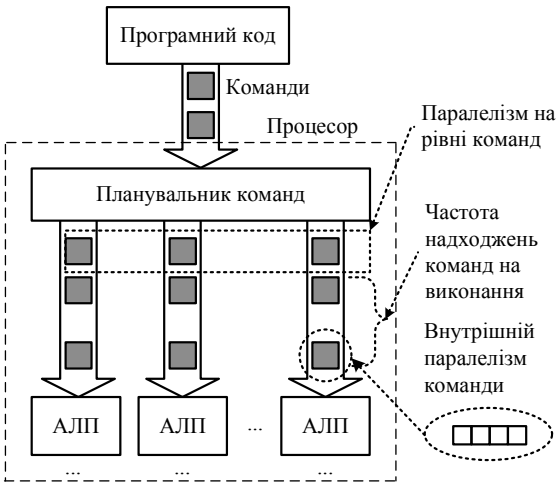

Рис. 6. Структура проблемно-орієнтованого процесора на базі кристалу ПЛІС

4.4. Використання проблемно-орієнтованих архітектур

Ще одним рішенням підвищення продуктивності комп'ютерних систем є застосування проблемно-орієнтованих архітектур [22] на базі кристалів програмованих логічних інтегральних схем (ПЛІС), загальний принцип роботи яких зображено на рис. 6. Широкий огляд архітектур, орієнтованих на апаратну підтримку СОЗ, висвітлено у [8].

Як платформа для реалізації пробле-

мно-орієнтованих архітектур сьогодні широко застосовуються кристали ПЛІС, які дозволяють швидко та відносно дешево реалізувати потрібну систему. Системи, які побудовані за таким принципом, називаються конфігурованими системами, оскільки мають змогу бути реконфігурованими у майбутньому. Головною особливістю конфігурованих систем є можливість змінювати архітектуру у відповідності з вимогами конкретної прикладної задачі. Принцип такої адаптації зводиться до масової заміни послідовної логіки на комбінаційну. Завдяки такій адаптації, конфігуровані системи здатні забезпечити значно продуктивніший розв'язок задач, які добре піддаються розпаралелюванню. Однак при цьому збільшується обсяг апаратури, яка бере участь у роботі системи, та як наслідок збільшуються витрати енергії на її роботу [23].

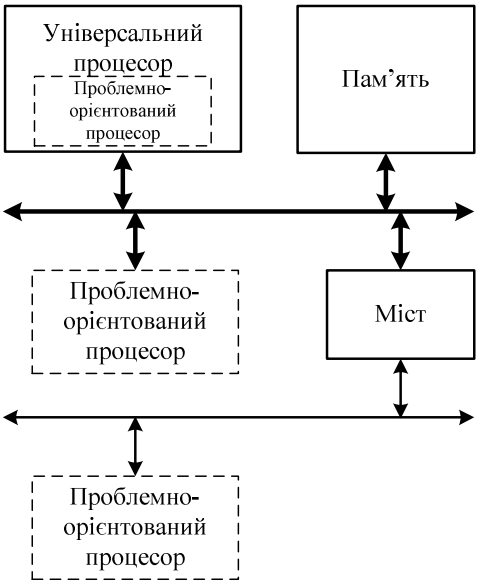

Рис. 7. Структура проблемно-орієнтованої системи

Проблемно-орієнтована система (рис. 7) містить такі ключові модулі: універсальний процесор (УП), пам'ять та проблемно-орієнтований процесор (ПОП). У залежності від розміщення ПОП відносно УП розрізняють два типи їх сполучення [23]: тісне сполучення – УП та ПОП розміщені в одному модулі (процесорі) або спільній процесорній шині та слабке сполучення – УП та ПОП розміщені на віддалених шинах. Вибір типу сполучення залежить від інтенсивності обміну даними між модулями.

Серед СОЗ такого класу можна виділити розробку [9], де запропоновано ПЛІС-реалізацію експертної системи, згідно з якою база знань трансформується в еквівалентну апаратну мережу, чиї вузли є фактами, а зв'язки між вузлами – відношеннями. Результати експериментів з даною системою свідчать про значну перевагу апаратної реалізації над програмною.

5. Висновки

На сьогодні реалізації СОЗ переважно використовують універсальні архітектури, зважаючи на їхню досконаліть, отриману завдяки зменшенню розмірів логічних елементів; зменшенню витрат споживаної енергії; підвищенню швидкості обміну даними між процесором та пам'яттю; використанню паралелізму на різних рівнях обробки інформації.

При розробці СОЗ доцільно застосовувати конфігуровані системи на базі ПЛІС, структура яких включає універсальний та спеціалізований процесори, який реалізує розширення набору команд універсального процесора. Спеціалізований процесор за рахунок заміни послідовної логіки на комбінаційну дозволяє суттєво підвищити загальну продуктивність системи.

Перспективність даного підходу обумовлена швидким розвитком ПЛІС-технологій: вдосконалення засобів проектування, розширення бібліотек компонентів, збільшення кількості логічних елементів на кристалі, збільшення розміру пам'яті на кристалі тощо.

СПИСОК ЛІТЕРАТУРИ

1. Knowledge-Based Intelligent Information and Engineering Systems (KES). Aim and Scope [Електронний ресурс]. – Режим доступу: http://www.kesinternational.org/aim.php.
2. Амамия М. Архитектура ЭВМ и искусственный интеллект / М. Амамия, Ю. Танака; пер. с японск. – М.: Мир, 1993. – 400 с.

3. Кургаев А.Ф. Проблемная ориентация архитектуры компьютерных систем / А.Ф. Кургаев. – Киев: Сталь, 2008. – 540 с.
4. Глушков В.М. Теория автоматов и вопросы проектирования структур цифровых машин / В.М. Глушков // Кибернетика. – 1965. – № 1. – С. 3 – 11.
5. Вычислительные машины с развитыми системами интерпретации / [В.М. Глушков, А.А. Барабанов, Л.А. Калиниченко и др.]. – Киев: Наукова думка, 1970. – 260 с.
6. Fifth generation computer: [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Fifth_generation_computer.
7. Stallings W. Computer Organization and Architecture. Designing for performance / W. Stallings, Eight edition. – Copyright © by Pearson Education, Inc., 2010. – 763 p.
8. Teodorescu H.N. Hardware Implementation of Intelligent systems / Teodorescu H.N., Jain L.C., Kandel A. – Heidelberg: Copyright © Physica-Verlag, 2001. – 282 p.
9. FPGA-Based hardware/software codesign of an expert system shell / A. Netin, D. Roman, O. Cret [et al.] // Proc. of the 13th International workshop, FPL 2003. – Lisbon, Portugal, 2003. – P. 1067 – 1070.
10. Захаров В.Н. Искусственный интеллект. Программные и аппаратные средства: справочник в 3 кн. / В.Н. Захаров, В.Ф. Хорошевский. – М.: Радио и связь, 1990. – Кн. 3. – 368 с.
11. Воеводин В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб.: БХВ-Петербург, 2004. – 608 с.
12. Корнеев В.В. Современные микропроцессоры / В.В. Корнеев, А.В. Киселев. – [3-е изд., перераб. и доп.]. – СПб.: БХВ-Петербург, 2003. – 448 с.
13. Patterson D. A VLSI RISC / D. Patterson, C. Sequin // IEEE Computer. – 1982. – September. – P. 8 – 21.
14. Lunde A. Empirical evaluation of some features of instruction set processor architectures / A. Lunde // Communications of the ACM. – 1977. – P. 143 – 153.
15. Tanenbaum A. Implications of structured programming for machine architecture / A. Tanenbaum // Communications of the ACM. – 1978. – P. 237 – 246.
16. Hennessy J. Hardware/Software tradeoffs for increased performance / J. Hennessy // Proc., Symposium on Architectural support for programming languages and operating systems. – 1982. – P. 2 – 11.
17. Heat J. Re-evaluation of RISC I / J. Heat // Computer architecture news. – 1984. – Vol. 12, N 1. – P. 3 – 10.
18. Smith A. Cache memories / A. Smith // ACM computing surveys. – 1982. – Vol. 14, N 3. – P. 473 – 530.
19. Intel 64 and IA-32 Architectures. Software Developer's Manual. – Vol. 1: Basic Architecture. Copyright © 1997 – 2010 Intel Corporation. – P. 512.
20 Основные классы современных параллельных компьютеров [Электронный ресурс]. – Режим доступа: http://www.parallel.ru/computers/classes.html.
21. Закону Мура – 40 лет! [Электронный ресурс]. – Режим доступа: http://www.ixbt.com/editorial/moorelaw40th.shtml.
22. Poznanovic D.P. The emergence of Non-von Neuman Processors / D.P. Poznanovic // Proc. of Second International Workshop, ARC 2006 Delft, (The Netherlands, March 2006). – 2006. – P. 243 – 254.
23. Hauck S. Reconfigurable computing: the theory and practice of FPGA-based computation / S. Hauck, A. Dehon. – Copyright © by Elsevier Inc., 2008. – P. 908.

Стаття надійшла до редакції 29.05.2012