аналітичним методом. Розрахунок проведений для семи положень механізму. Необхідна сила гідроциліндру визначалась для кожного положення за допомогою методу малих робіт. Графічна частина аналізу проведена в середовищі AutoCAD. До неї входять плани положень механізму, необхідні графіки залежностей, зокрема, графіки передаточної функції, гідроциліндр – вихідна ланка, кутів тиску в шарнірах гідроциліндра та інші, плани сил для кожної групи.

Силовий аналіз показав такі результати:

1. Кінематичні пари механізму навантажені під час роботи вкрай нерівномірно. Відношення навантаження між найбільш та найменш навантаженими парами становить 112 разів.

2. Навантаження в кінематичних парах змінюється від висоти піднімання. Найбільш небезпечним є положення, що відповідає початку піднімання платформи. При збільшенні кута нахилу важелів зменшуються сили в кінематичних парах.

Відношення навантажень в критичних кінематичних парах на початку і наприкінці піднімання становить 9 разів.

Висновки. Проведений силовий аналіз ножичного механізму показав характер зміни навантажень у кінематичних парах ланок механізму і дав можливість виявити найнавантаженіші ланки. Знайдені вектори критичних сил, які дають можливість розрахувати на міцність шарніри і ланки механізму. На нашу думку, доцільною є конструкція механізму, де б ідентичні за структурою ланки в різних групах механізму були б виконані різними за поперечним перерізом. Це б дозволило зменшити вагу конструкції.

М. Березовський
Науковий керівник – д-р техн. наук, проф. Р.Б. Дунець

ДИНАМІЧНО РЕКОНФІГУРОВАНЕ ПРОЦЕСОРНЕ ЯДРО

Мета – створення динамічно реконфігурованого процесорного ядра, як центрального процесора для персональних та високопродуктивних спеціалізованих комп'ютерів. Запропоновано нове процесорне ядро, яке за своєю структурою нагадує дещо спрощену ПЛІС, і складається з набору різних за функціональним призначенням комірок та набору зв'язків між ними. Таку архітектуру можна вважати підвидом архітектури VLIW.

У кожній комірці перебувають пристрої, наприклад, суматори, помножувачі, лічильники, мультиплексори, шифратори, дешифратори, регістри, інтерфейси тощо. Набір цих комірок повинен бути доволі великим, з тим щоби забезпечити високу швидкодію завдяки паралелізму обчислень. Вибираючи набір комірок, потрібно враховувати операції, які виконуються доволі часто (математичні операції з дійсними числами, операції сортування та пошуку даних тощо).

З'єднання між комірками будуть реалізовувати можливість з'єднувати комірки між собою подібно до з'єднань у ПЛІС. З'єднання між комірками у конкретний момент часу задає поточна машинна команда.

Виконання арифметичних та логічних операцій забезпечується пересилкою операндів у регістри вхідних даних комірки, і подальшим зчитуванням результату з виходу комірки. До того ж складні багатотактові команди, як, наприклад множення, під час виконання можна виконувати під час виконання наступних команд, при цьому у програмі повинна забезпечуватися необхідна затримка між записуванням операндів та зчитуванням даних.

Доступ до пам'яті відбувається також через комірку, яка має вхід адреси, вхід та вихід даних. Умовні та безумовні переходи виконуються відповідною коміркою, яка має входи адреси та дозволу переходу. При безумовному переході на вхід дозволу переходу подається константа – логічна одиниця, при умовному – сигнал з виходу іншої комірки, який визначає чи перехід повинен здійснюватися.

Крім того, в ядрі присутній сигнал дозволу наступної команди, який дає змогу виконуватися одній і тій самій машинній команді декілька тактів. Цей сигнал дозволяє організувати програмний цикл на одній машинній команді з швидкодією один крок циклу за машинний такт при простих циклах, таких як пошук, сортування тощо. Під час виконання такого циклу, конфігурація процесора залишається незмінною – наприклад, у конфігурації для пошуку даних вихід лічильника (попередньо проініціалізований початковою адресою масиву, в якому шукається значення), який здійснює операцію "+1" за тактовим імпульсом, під'єднаний до входу адреси ОЗП, а вихід даних ОЗП і регістр, із записаним попередньою командою значенням — числом, пошук якого здійснюється, підключені до входів компаратора, вихід якого під'єднаний до сигналу дозволу наступної команди.

Ця конфігурація забезпечує виконання циклу пошуку заданого значення (числа) за кількість машинних тактів, що дорівнює зміщенню

шуканого значення відносно першого елемента масиву плюс такт на завантаження початкової адреси у лічильник та шуканого значення у регістр.

Однією з ключових проблем під час розроблення такого ядра є вибір оптимального набору комірок та оптимізації матриці зв'язків між ними, оскільки велика кількість входів та виходів, поєднана з гнучкістю конфігурації, призводить до дуже великої розрядності машинної команди. Для цього необхідно скоротити варіанти з'єднань комірок, об'єднати входи і виходи у байти та слова, та зменшити кількість самих комірок.

Запропонована структура ядра, залежно від конкретного набору комірок, могла би емулювати нескладні ядра процесорів інших архітектур для забезпечення сумісності з наявним програмним забезпеченням під сучасні комп'ютери.

Отже, запропоноване ядро дає змогу отримати високу швидкодію, особливо під час циклів, наприклад під час сортування чи пошуку даних, порівняно із сучасними процесорами.

Ю. Мицко
Науковий керівник – канд. техн. наук, доц. А.О. Мельник

ПАРАЛЕЛЬНЕ ВИКОНАННЯ АЛГОРИТМІВ, ПОДАНИХ СТРУКТУРНОЮ МАТРИЦЕЮ

Сьогодні стрімко розвиваються комп’ютерні технології. Збільшується продуктивність процесорів, кількість ядер в кожному з них та їхня кількість у комп’ютерній системі. Також є можливість використання графічних процесорів для розв’язання задач загального призначення. Однак багато алгоритмів та програм є послідовними, виконуються на одному ядрі, тим самим не раціонально використовують ресурси обчислювальної системи. Саме тому нині є актуальним питання пошуку методів їх розпаралелення.

Відомо багато різних вирішень проблеми, однак вони не є універсальними і для кожного алгоритму потрібен свій підхід з його розпаралелення. Для прикладу в OpenMP потрібно явно вказувати в коді програми, яка її частина буде виконуватися паралельно. Якщо програма вже створена і виконується послідовно, то за такого підходу для