DOI: https://doi.org/10.36910/6775-2524-0560-2026-64-07
УДК 004.312.46
Булатецький Віталій Вікторович, к. фіз.-мат.н., доцент
https://orcid.org/0000-0002-9883-4550
Собчук Оксана Миколаївна, канд. пед. наук, доцент
https://orcid.org/0000-0003-2185-9896
Волинський національний університет імені Лесі Українки, м. Луцьк, Україна

ЕВОЛЮЦІЯ АРХІТЕКТУР ЦИФРОВИХ ПРИСТРОЇВ МНОЖЕННЯ У БАЗИСІ АЛГЕБРИ ЛОГІКИ

Булатецький В. В., Собчук О. М. Еволюція архітектур цифрових пристроїв множення у базисі алгебри логіки. У сучасній архітектурі обчислювачів операція множення є значно складнішою за додавання. Якщо суматори визначають базову швидкість АЛП, то ефективність пристроїв множення (або Помножувачів) лімітує продуктивність систем у задачах цифрової обробки сигналів, криптографії та штучного інтелекту. Фундаментом для побудови будь-якого цифрового помножувача є апарат алгебри логіки. На рівні логічних вентилів процес множення двійкових чисел традиційно розпадається на два ключові етапи: формування масиву часткових добутків та їхнє подальше каскадне підсумовування. Попри те, що математична суть операції залишається незмінною, існує велика кількість апаратних реалізацій, кожна з яких пропонує власний компроміс між швидкістю роботи, площею на кристалі та споживаною потужністю. У цій роботі розглядається еволюція логічних схем множення: від класичних матричних структур, що мають регулярну, але повільну топологію, до високоефективних деревних структур Воллеса та Дадда. Окрему увагу в дослідженні приділено методам подолання фізичних меж швидкодії комбінаційної логіки. Зокрема, аналізується впровадження бар'єрних регістрів для розбиття критичного шляху проходження сигналу. Метод конвеєризації перетворює статичну схему процедури множення (або Блоку множення) на динамічну «складальну лінію», що дозволяє обробляти нові набори даних у кожному такті, значно підвищуючи пропускну здатність пристрою. Також у роботі аналізуються методи скорочення кількості операцій за допомогою алгоритму Бута та особливості впровадження цих архітектур у сучасні напівпровідникові системи типу FPGA та ASIC. Розглядається сучасний модульний підхід до проектування, який дозволяє мінімізувати енерговитрати в системах на кристалі (SoC). Дослідження цих підходів дозволяє глибше зрозуміти, як за допомогою комбінації логічних елементів та стратегій керування часовими затримками досягається експоненціальне зростання обчислювальної потужності сучасних мікропроцесорних систем.
Ключові слова: пристрої множення, алгебра логіки, алгоритм Бута, дерево Дада, дерево Воллеса, конвеєризація, рекурсивна декомпозиція.

Bulatetskyi V., Sobchuk O. Evolution of Digital Multiplier Architectures in the Basis of Boolean Algebra. In modern computer architecture, the multiplication operation is significantly more complex than addition. While adders determine the baseline speed of the ALU, the efficiency of multiplier units (or multipliers) limits system performance in digital signal processing, cryptography, and artificial intelligence tasks. The foundation for constructing any digital multiplier is the apparatus of Boolean algebra. At the logic gate level, the process of binary multiplication traditionally breaks down into two key stages: the generation of a partial product array and their subsequent cascade summation. Although the mathematical essence of the operation remains unchanged, there is a large number of hardware implementations, each offering its own trade-off between operational speed, silicon area, and power consumption. This paper examines the evolution of multiplication logic circuits: from classical array structures, which feature a regular but slow topology, to highly efficient Wallace and Dadda tree structures. Special attention in the study is paid to methods for overcoming the physical speed limits of combinational logic. In particular, the introduction of pipeline registers to break the critical signal propagation path is analyzed. The pipelining method transforms the static circuit of the multiplication procedure (or multiplier unit) into a dynamic "assembly line," which allows processing new data sets in each clock cycle, significantly increasing the device throughput. Furthermore, the paper analyzes methods for reducing the number of operations using Booth's algorithm and the specific aspects of implementing these architectures in modern semiconductor systems such as FPGA and ASIC. The modern modular design approach, which allows minimizing energy consumption in systems-on-chip (SoC), is also considered. The study of these approaches provides a deeper understanding of how an exponential increase in the computational power of modern microprocessor systems is achieved through a combination of logic elements and time delay management strategies.
Keywords: multipliers, Boolean algebra, Booth's algorithm, Dadda tree, Wallace tree, pipelining, recursive decomposition.

Постановка проблеми. У сучасну епоху цифрової трансформації ефективність арифметико-логічних пристроїв стала визначальним фактором розвитку обчислювальних систем. Оскільки операція множення є однією з найбільш ресурсомістких і часто повторюваних у мікропроцесорній техніці, її оптимізація на рівні алгебри логіки безпосередньо впливає на продуктивність сучасних комп'ютерів. Особливої гостроти це питання набуває у сфері штучного інтелекту та глибокого машинного навчання, де алгоритми базуються на мільярдах операцій множення із накопиченням (MAC). Розуміння різних логічних реалізацій процедури множення двійкових чисел дозволяє розробникам знаходити критичний баланс між швидкістю обчислень, площею кристала та енергоефективністю, що є вирішальним для мобільних пристроїв та

автономних систем. Таким чином, дослідження логічних основ множення залишається фундаментом для проектування енергоефективних та високопродуктивних чипів.

Множення як послідовність логічних операцій. Найпоширенішою та найбільш інтуїтивно зрозумілою реалізацією множення є матричний помножувач. Його архітектура безпосередньо відображає ручний метод множення «у стовпчик», де кожна цифра нижнього числа множиться на кожну цифру верхнього, утворюючи сітку (матрицю) значень.

Як показано на рис. 1, *а*, процес починається з генерації часткових добутків. Оскільки в двійковій системі результат множення двох бітів ($x_i \cdot y_j$) може бути лише 0 або 1, ця операція повністю еквівалентна логічній кон'юнкції. У схемотехніці це реалізується масивом елементів «І» (AND), які працюють паралельно, створюючи всі необхідні добутки для подальшого додавання. [1, 2].

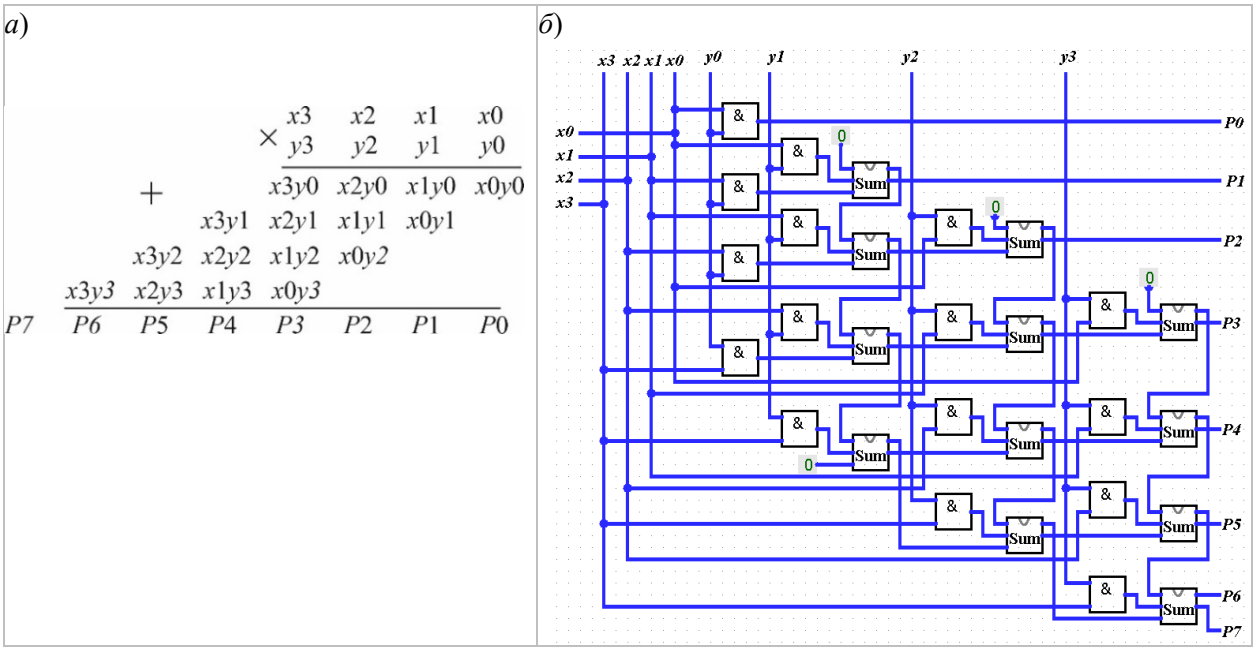

Рис. 1. Алгоритм (*а*) та схема (*б*) множення двох трирозрядних двійкових чисел

Головна складність полягає в ефективному сумуванні цих добутків із урахуванням їхньої ваги (зсуву). На рис. 1,*б* продемонстровано схему множення двох чотирирозрядних чисел (4×4). Вона складається з ієрархії однорозрядних суматорів. Перший біт результату P_0 отримується безпосередньо з елемента «І». Для P_1 використовується суматор, що додає два часткових добутки. При обчисленні середніх розрядів (зокрема P_2, P_3 та P_4) кількість доданків досягає свого максимуму (до 4-х у центральному стовпці). На цьому етапі критично важливо враховувати не лише поточні часткові добутки, а й декілька сигналів переносу (Carry), що надходять від суматорів молодших розрядів. Для коректної обробки такого «нашарування» сигналів у центральній частині матриці використовується каскадне включення повних суматорів, що дозволяє перетворювати декілька рівнів часткових добутків у два фінальні вектори.

Завершальні розряди результату (P_5, P_6 та P_7) формуються шляхом послідовного додавання залишків та фінальних переносів. Останній біт P_7 фактично є результуючим виходом переносу останнього каскаду суматорів, що завершує формування повного 8-розрядного добутку.

Матрична структура має чітко виражені особливості. Схема складається з ідентичних типів комірок, що робить її ідеальною для проектування на кристалах (VLSI). Вона легко масштабується для 8, 16 або 32 біт. Головним недоліком є «критичний шлях» сигналу. Перенос має пройти крізь довгий ланцюг суматорів. Це робить такий помножувач або пристрій множення відносно повільним при великій розрядності ($O(n)$ затримка).

Дерево Воллеса (Wallace Tree) та Дерево Дада (Dadda Tree). Для мінімізації затримок, характерних для матричних множників, застосовують архітектури на основі паралельного стиснення масиву часткових добутків (reduction trees), найбільш ефективними серед яких є дерева Воллеса та Дада [3,4,5]. Головна ідея полягає в тому, щоб не додавати добутки один за одним, а групувати їх і підсумовувати паралельно. У цій архітектурі повні суматори (FA) та напівсуматори (HA) розглядаються як компресори. Компресор 3:2 (стандартний повний суматор) приймає 3 вхідні

біти однакової ваги та «стискає» їх у 2 вихідні біти (сума та перенос). Компресор 4:2 більш складна логічна конструкція, що дозволяє обробляти 4 вхідні біти одночасно, суттєво зменшуючи кількість рівнів у дереві [6].

Процес побудови дерева Воллеса складається з трьох етапів.

1. Створення масиву всіх часткових добутків за допомогою елементів «І».
2. Групування бітів однакової ваги по 3 і проходження їх крізь компресори 3:2. Біти, що залишилися без пари, просто переходять на наступний рівень. Цей процес повторюється ітеративно, доки для кожної ваги розряду не залишиться лише 2 біти (рис. 2*а)*.
3. Два отримані фінальні вектори («вектор сум» – s_{ii} та «вектор переносів» – c_{ij}) додаються один до одного за допомогою швидкого суматора (наприклад, CLA – Carry Look-ahead Adder) (рис 2*а*, 3-ій каскад,) [3].

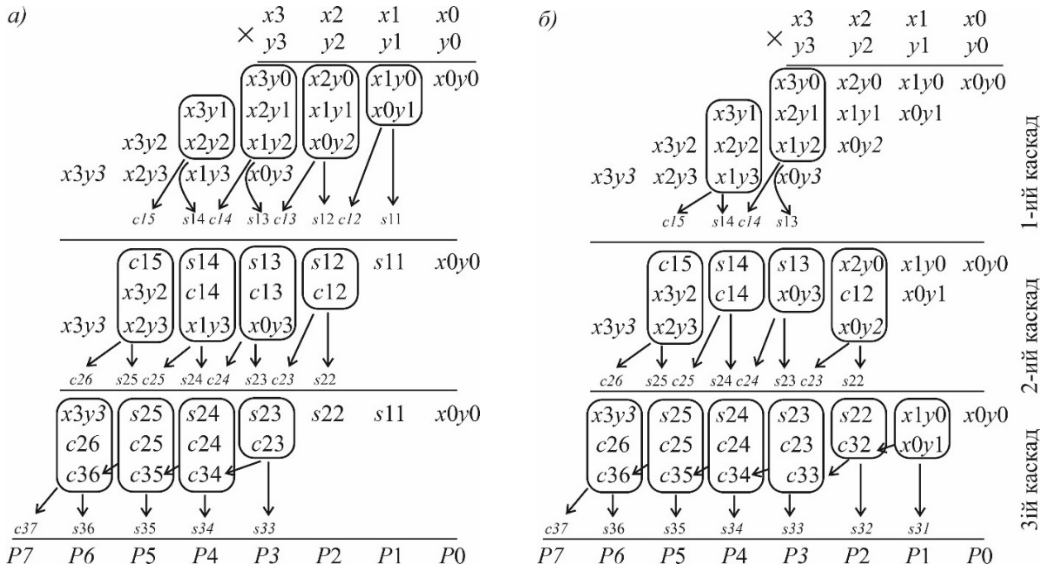

Рис. 2. Дерево Воллеса (*а*) та Дада (*б*) для 4-бітних операндів: обведені два доданки – напівсуматори; обведені три доданки – повні суматори.

Алгоритм дерева Дада базується на ідентичних принципах ієрархічного стиснення, проте використовує іншу стратегію оптимізації. На відміну від дерева Воллеса, де стиснення масиву часткових добутків відбувається максимально інтенсивно на кожному рівні, метод Дада передбачає виконання операцій стиснення лише для досягнення цільової висоти стовпців, визначеної критичною послідовністю (близькою до чисел Фібоначчі) (рис. 2, *б*). Такий підхід забезпечує ідентичну логічну затримку обчислень при суттєвому скороченні кількості апаратних суматорів, що сприяє зменшенню площі кристала при реалізації схеми [4].

Довгий час вважалося, що через однакову кількість рівнів стиснення обидва типи архітектур помножувачів мають ідентичні часові характеристики. Проте детальний аналіз затримок, проведений авторами роботи [7], спростовує це припущення. Результати їхнього дослідження вказують на те, що попри використання довшого фінального суматора, архітектура Дада забезпечує дещо вищу підсумкову швидкість обчислень порівняно з деревом Воллеса. Це пояснюється специфікою розподілу внутрішніх затримок на етапах стиснення, що робить архітектуру Дада ефективнішою у контексті часових витрат [7].

Загалом, перехід до будь-якої з цих деревоподібних структур кардинально змінює часову складність операції порівняно з традиційними матричними помножувачами. Якщо при матричному множенні затримка має лінійну залежність O(n), де n – розрядність чисел, то для дерев Воллеса та Дада вона є логарифмічною – O(log n) [3, 4].

Алгоритм Бута. Одним із найефективніших способів підвищення продуктивності пристроїв множення є скорочення кількості часткових добутків ще на етапі їх генерації. Основним інструментом для досягнення цієї мети є алгоритм Бута та його сучасні модифікації [8, 9]. Фундаментальна ідея алгоритму Бута базується на властивості двійкових чисел, згідно з якою групу з n послідовних одиниць можна представити як різницю двох степенів двійки. Наприклад, число

$01110_{(2)}$ (що дорівнює 14) можна записати як $2^4 - 2^1$ (16 - 2). З погляду алгебри логіки та арифметики, це дозволяє замінити декілька операцій додавання лише двома операціями: одним відніманням на початку послідовності одиниць та одним додаванням після її завершення.

Алгоритм аналізує пари сусідніх бітів множника (y_{i+1}, y_i) у напрямку від молодших до старших:

- 10: Початок послідовності одиниць – виконується віднімання.
- 01: Кінець послідовності – виконується додавання.
- 00 або 11: Послідовність триває або відсутня – операції додавання/віднімання пропускаються, виконується лише зсув.

Для старту алгоритму на самому початку (при i=0) вводять уявний додатковий біт y_{-1}.

Це дозволяє суттєво прискорити обчислення, якщо множники містять довгі блоки неперервних одиниць.

Подальший розвиток ідеї призвів до створення модифікованого алгоритму Бута (Modified Booth's Algorithm), який використовує кодування першого операнда дії множення за вищими основами (Radix). В алгоритмі Radix-4 аналізуються три біти одночасно (із перекриттям в один біт). Це дозволяє закодувати множене цифрами із множини {-2, -1, 0, 1, 2}. Головна перевага Radix-4 полягає у дворазовому скороченні кількості часткових добутків порівняно зі стандартним множенням. Алгоритм Radix-8 використовує вікно у чотири біти, що дозволяє скоротити кількість часткових добутків утричі (до n/3, де n – розрядність). Однак цей метод потребує попереднього обчислення «важких» кратних (наприклад, 3 × множене), що дещо ускладнює апаратну частину. Вибір між Radix-4 та Radix-8 є класичним компромісом (trade-off) між складністю генератора часткових добутків та швидкістю їх подальшого додавання в деревах Воллеса чи Дада. [10]

Застосування алгоритму Бута безпосередньо впливає на фізичні характеристики цифрових систем. Менша кількість часткових добутків означає потребу у меншій кількості суматорів у дереві стиснення. Оскільки динамічна потужність цифрової схеми прямо залежить від активності перемикання вентилів (switching activity), зменшення кількості арифметичних операцій призводить до значного енергозбереження. Скорочення висоти масиву часткових добутків дозволяє використовувати меншу кількість рівнів (каскадів) суматорів, що критично важливо для високочастотних обчислювальних систем. Таким чином, комбінація кодування Бута для генерації добутків та дерева Дада для їх стиснення є одним із найбільш енергоефективних архітектурних рішень у сучасній мікроелектроніці.

Ще більше скорочення кількості додавань може дати модифікація, запропонована Леманом [11]. Суть модифікації полягає в тому, що якщо дві послідовності нулів розділені одиницею, що стоїть у k-й позиції, то замість віднімання в k-й позиції та додавання в (k+1)-й позиції достатньо виконати лише додавання в k-й позиції. Якщо дві групи одиниць розділені нулем, що стоїть у k-й позиції, замість додавання в k-й позиції і віднімання в (k+1)-й позиції достатньо виконати тільки віднімання в k-й позиції.

Рекурсивна декомпозиція структур паралельного множення.

Як правило, апаратні пристрої множення мають обмеження щодо кількості розрядів вхідних чисел. Проте помножувач підвищеної розрядності можна отримати з модулів меншої розрядності, вибудовуючи ієрархічну структуру. Наприклад, для побудови помножувача розрядністю 8 × 8 доцільно використовувати чотири модулі типу 4×4. У такому випадку процес обчислення реалізується за таким алгоритмом.

1. Множене A розбивається на старшу (A_h) та молодшу (A_l) частини. Аналогічним чином множник B поділяється на B_h та B_l.
2. Чотири модулі 4×4 одночасно обчислюють часткові добутки: $A_h \times B_h$, $A_h \times B_l$, $A_l \times B_h$ та $A_l \times B_l$. На виходах цих модулів отримуються результати, що відповідають положенню частин у загальній розрядній сітці.
3. Отримані 8-розрядні результати комбінуються для формування остаточного 16-розрядного добутку. Частина $A_l \times B_l$ відповідає молодшим розрядам (7–0), перехресні добутки зміщуються ліворуч на 4 розряди, а старший добуток $A_h \times B_h$ – на 8 розрядів..

Схематично процес формування результату можна представити як багаторівневе сумування, де результати середніх рівнів додаються з урахуванням їхньої ваги в розрядній сітці.

Фундаментальним підходом до підвищення продуктивності сучасних обчислювальних вузлів є рекурсивна декомпозиція операції множення. Математичне обґрунтування цього методу, вперше запропоноване А. Карацубою, довело, що складність множення n-розрядних чисел може

бути суттєво знижена шляхом рекурсивного розбиття операндів на блоки меншої розрядності. На відміну від класичного підходу, алгоритм Карацуби базується на заміні одного з чотирьох часткових добутків операціями додавання та віднімання, що дозволяє знизити обчислювальну складність з $O(n^2)$ до $O(n^{1.58})$. При застосуванні логіки Карацуби кількість операцій множення може бути скорочена до трьох за рахунок обчислення добутку сум $(A_h+A_l)\times(B_h+B_l)$ та подальшої алгебраїчної корекції. Це дозволяє перейти від ітеративних матричних структур з регулярною топологією до ієрархічних архітектур, у яких затримка поширення сигналу суттєво зменшується відносно розрядності операндів [10, 12].

Така організація логічних каскадів забезпечує високу масштабованість пристрою, що підтверджується сучасними дослідженнями модульних архітектур [13]. Рекурсивна декомпозиція дозволяє нарощувати обчислювальну потужність без експоненціального зростання часових затримок, що є критично важливим для систем на кристалі (SoC), орієнтованих на швидкісну обробку сигналів та роботу нейронних мереж [14].

Розбиття схеми за допомогою регістрів. Навіть найбільш досконалі деревоподібні структури стиснення мають певну фізичну межу швидкодії, зумовлену затримкою на критичному шляху при проходженні сигналу крізь усі рівні комбінаційної логіки. Матричні та деревоподібні структури паралельних помножувачів мають значний потенціал для підвищення продуктивності через конвеєризацію. Цей метод передбачає розбиття обчислювального процесу на послідовність завершених кроків. Кожен етап множення виконується на окремому шарі конвеєра, які працюють паралельно. Результати з i-го шару передаються для подальшої обробки на $(i + 1)$-й шар через проміжну буферну пам'ять (регістри), встановлену між ними.

Конвеєризація – це техніка, що перетворює помножувач на «складальну лінію», максимізуючи використання апаратних ресурсів. Фундаментальні аспекти розбиття логічних схем бар'єрними регістрами для скорочення цього шляху детально висвітлені у роботі [15]. Дослідження [10] акцентують увагу на тому, що ключовим питанням є оптимальний вибір точок розрізу (cut sets) для встановлення регістрів, що дозволяє досягти балансу між латентністю (latency) та тактовою частотою. Кожен етап виконує лише частину обчислень (наприклад, один рівень стиснення дерева Воллеса), а регістри фіксують проміжний результат у кінці кожного такту. Це дозволяє наступному етапу продовжувати обробку першого набору даних, поки попередній етап уже приймає новий пакет інформації. Головна перевага конвеєрного пристрою множення – це здатність видавати готовий результат на кожному такті, незалежно від загальної кількості стадій конвеєра. Аналіз ефективності таких рішень у порівнянні з матричними структурами наведено в роботі [7], де підкреслюється перевага конвеєрних деревоподібних обчислювачів за показником пропускної здатності на одиницю площі кристала.

Сучасні дослідження в галузі інтеграції конвеєрних помножувачів у процесорні архітектури свідчать про суттєве підвищення загальної продуктивності систем. Зокрема, у роботі [17] досліджено конвеєризований пристрій множення у складі архітектури RISC-V, де поєднання алгоритмів Бута та дерева Воллеса дозволило досягти тактової частоти 1 ГГц. Такий підхід забезпечує оптимальний баланс між швидкістю та енергоспоживанням, що є критичним для вбудованих застосувань. Ще більш глибока конвеєризація представлена у дослідженні пристрою для операцій множення-додавання (Fused Multiply-Add, FMA) [18]. Ця архітектура підтримує до 10 рівнів конвеєра, що дозволяє функціонувати на частотах до 1,67 ГГц, орієнтуючись на високонавантажені обчислення з плаваючою комою.

Оцінка архітектурних рішень для FPGA-реалізацій процесорів на базі RV32I [19] також підтверджує, що навіть дворівнева конвеєризація блока множення дозволяє значно скоротити критичний шлях та підвищити стабільність роботи ядра. Питання енергоефективності при цьому залишається пріоритетним: використання модульного проектування дерева Воллеса [13] та оптимізація міжз'єднань у комбінації з конвеєризацією забезпечують кращу масштабованість обчислювальних вузлів у сучасних системах на кристалі (SoC), призначених для цифрової обробки сигналів (DSP) та апаратного прискорення нейронних мереж [16].

Висновки. Проведений аналіз еволюції архітектур логічних помножувачів дозволяє сформулювати цілісне бачення розвитку цього напряму. Еволюція від базових матричних структур до складних деревоподібних схем стиснення Воллеса та Дадазабезпечила стратегічний перехід від лінійної залежності затримки від розрядності до логарифмічної. Це стало визначальним фактором підвищення швидкодії сучасних обчислювальних систем. Водночас впровадження бар'єрних регістрів визначено як ключовий метод подолання фізичної межі швидкодії комбінаційної логіки.

Оптимальний вибір точок розрізу (cut sets) дозволяє стабілізувати роботу пристроїв на гігагерцових частотах, перетворюючи помножувач на високоефективну «складальну лінію» з максимальною пропускною здатністю. Сучасні пріоритети проєктування пристроїв вимагають суворого балансу між тактовою частотою, латентністю та енергоефективністю. Модульний підхід до побудови дерев стиснення забезпечує створення масштабованих архітектур, які адаптовані до жорстких вимог систем на кристалі (SoC), зокрема в галузях цифрової обробки сигналів (DSP) та апаратного прискорення нейронних мереж. Подальший розвиток у цьому напрямку пов'язаний із пошуком нових логічних базисів та методів мінімізації енергоспоживання при збереженні високої пропускної здатності.

Список використаних джерел

1. Harris, D., & Harris, S. (2021). *Digital Design and Computer Architecture.* Morgan Kaufmann. http://www.r-5.org/files/books/computers/hw-layers/hardware/digital-desigh/David_Harris_Sarah_Harris-Digital_Design_and_Computer_Architecture-EN.pdf
2. Vlăduțiu M. Computer Arithmetic. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. URL: https://doi.org/10.1007/978-3-642-18315-7.
3. Hsiao S.-F., Jiang M.-R., Yeh J.-S. Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers. *Electronics Letters*. 1998. Vol. 34, no. 4. P. 341. URL: https://doi.org/10.1049/el:19980306 (date of access: 01.05.2026).
4. Wallace C S. A suggestion for a fast multiplier. IEEE Transactions on electronic Computers, 1964 (1): 14-17.
5. Yu H. Systematic Analysis on Performance Optimization of Tree Multipliers. *Applied and Computational Engineering*. 2025. Vol. 127, №. 1. P. 7–15. URL: https://doi.org/10.54254/2755-2721/2025.20275.
6. Dadda L. Some schemes for parallel multipliers. IEEE Computer Society Press, 1990.
7. Townsend W. J., Swartzlander, Jr. E. E., Abraham J. A. A comparison of Dadda and Wallace multiplier delays. *Optical Science and Technology, SPIE's 48th Annual Meeting*, San Diego, California, USA / ed. by F. T. Luk. 2003. URL: https://doi.org/10.1117/12.507012 (date of access: 01.05.2026).
8. Booth A. D. A signed binary multiplication technique. *The Quarterly Journal of Mechanics and Applied Mathematics*. 1951. Vol. 4, Iss. 2. P. 236–240.
9. Bewick G. W. *Fast Multiplication: Algorithms and Implementation*. PhD Thesis. Stanford University, 1994. 155 p.
10. Parhami B. *Computer Arithmetic: Algorithms and Hardware Designs*. 2nd ed. Oxford University Press, 2010. 641 p.
11. Lehman M. Short-cut multiplication and division in automatic binary digital computers, with special reference to a new multiplication process. *Proceedings of the IEE - Part B: Radio and Electronic Engineering*. 1958. Vol. 105, no. 23. P. 496–504. URL: https://doi.org/10.1049/pi-b-1.1958.0332.
12. Karatsuba A., Ofman Yu. Multiplication of Multidigit Numbers on Automata. *Cybernetics and control theory*, 1963. Vol 7, №4. P.595-596
13. Solanki, V., Darji, A.D. & Singapuri, H. Design of Low-Power Wallace Tree Multiplier Architecture Using Modular Approach. *Circuits Syst Signal Process* 40, 4407–4427 (2021). https://doi.org/10.1007/s00034-021-01671-3.
14. Ercegovac M., Lang T. *Digital Arithmetic*. Morgan Kaufmann Publishers Inc., 2003. 709 p.
15. Patterson D. A., Hennessy J. L. *Computer Organization and Design: The Hardware/Software Interface*. 5th ed. Morgan Kaufmann Elsevier Inc. 2014. 575 p.
16. Модель, структура та метод синтезу нейронного елемента матричного типу / І. Г. Цмоць та ін. *Scientific Bulletin of UNFU*. 2024. Т. 34, № 4. С. 68–77. URL: https://doi.org/10.36930/40340409.
17. Design and Implementation of RISC-V Based Pipelined Multiplier / R. Sun et al. *Journal of Physics: Conference Series*. 2023. Vol. 2625, no. 1. P. 012006. URL: https://doi.org/10.1088/1742-6596/2625/1/012006.
18. A Deeply Pipelined FMA Unit for High Performance RISC-V Processor / Y. Qi et al. 2023 2nd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT), Xiamen, China, 4–7 August 2023. 2023. URL: https://doi.org/10.1109/ccpqt60491.2023.00015 (date of access: 05.05.2026).
19. Design of Adjacent Interconnect Processor Based on RISC-V / Y. Lu et al. 2021 IEEE 4th International Conference on Electronics Technology (ICET), Chengdu, China, 7–10 May 2021. 2021. URL: https://doi.org/10.1109/icet51757.2021.9451102 (date of access: 05.05.2026).

References

1. Harris, D., & Harris, S. (2021). *Digital Design and Computer Architecture.* Morgan Kaufmann. http://www.r-5.org/files/books/computers/hw-layers/hardware/digital-desigh/David_Harris_Sarah_Harris-Digital_Design_and_Computer_Architecture-EN.pdf
2. Vlăduțiu M. Computer Arithmetic. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. URL: https://doi.org/10.1007/978-3-642-18315-7.
3. Hsiao S.-F., Jiang M.-R., Yeh J.-S. Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers. *Electronics Letters*. 1998. Vol. 34, no. 4. P. 341. URL: https://doi.org/10.1049/el:19980306 (date of access: 01.05.2026).
4. Wallace C S. A suggestion for a fast multiplier. IEEE Transactions on electronic Computers, 1964 (1): 14-17.
5. Yu H. Systematic Analysis on Performance Optimization of Tree Multipliers. *Applied and Computational Engineering*. 2025. Vol. 127, №. 1. P. 7–15. URL: https://doi.org/10.54254/2755-2721/2025.20275.
6. Dadda L. Some schemes for parallel multipliers. IEEE Computer Society Press, 1990.

7. Townsend W. J., Swartzlander, Jr. E. E., Abraham J. A. A comparison of Dadda and Wallace multiplier delays. *Optical Science and Technology, SPIE's 48th Annual Meeting*, San Diego, California, USA / ed. by F. T. Luk. 2003. URL: https://doi.org/10.1117/12.507012 (date of access: 01.05.2026).
8. Booth A. D. A signed binary multiplication technique. *The Quarterly Journal of Mechanics and Applied Mathematics*. 1951. Vol. 4, Iss. 2. P. 236–240.
9. Bewick G. W. *Fast Multiplication: Algorithms and Implementation*. PhD Thesis. Stanford University, 1994. 155 p.
10. Parhami B. *Computer Arithmetic: Algorithms and Hardware Designs*. 2nd ed. Oxford University Press, 2010. 641 p.
11. Lehman M. Short-cut multiplication and division in automatic binary digital computers, with special reference to a new multiplication process. *Proceedings of the IEE - Part B: Radio and Electronic Engineering*. 1958. Vol. 105, no. 23. P. 496–504. URL: https://doi.org/10.1049/pi-b-1.1958.0332.
12. Karatsuba A., Ofman Yu. Multiplication of Multidigit Numbers on Automata. *Cybernetics and control theory*, 1963. Vol 7, №4. P.595-596
13. Solanki, V., Darji, A.D. & Singapuri, H. Design of Low-Power Wallace Tree Multiplier Architecture Using Modular Approach. *Circuits Syst Signal Process* 40, 4407–4427 (2021). https://doi.org/10.1007/s00034-021-01671-3.
14. Ercegovac M., Lang T. *Digital Arithmetic*. Morgan Kaufmann Publishers Inc., 2003. 709 p.
15. Patterson D. A., Hennessy J. L. *Computer Organization and Design: The Hardware/Software Interface*. 5th ed. Morgan Kaufmann Elsevier Inc. 2014. 575 p.
16. Модель, структура та метод синтезу нейронного елемента матричного типу / І. Г. Цмоць та ін. *Scientific Bulletin of UNFU*. 2024. Т. 34, № 4. С. 68–77. URL: https://doi.org/10.36930/40340409.
17. Design and Implementation of RISC-V Based Pipelined Multiplier / R. Sun et al. *Journal of Physics: Conference Series*. 2023. Vol. 2625, no. 1. P. 012006. URL: https://doi.org/10.1088/1742-6596/2625/1/012006.
18. A Deeply Pipelined FMA Unit for High Performance RISC-V Processor / Y. Qi et al. 2023 2nd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT), Xiamen, China, 4–7 August 2023. 2023. URL: https://doi.org/10.1109/ccpqt60491.2023.00015 (date of access: 05.05.2026).
19. Design of Adjacent Interconnect Processor Based on RISC-V / Y. Lu et al. 2021 IEEE 4th International Conference on Electronics Technology (ICET), Chengdu, China, 7–10 May 2021. 2021. URL: https://doi.org/10.1109/icet51757.2021.9451102 (date of access: 05.05.2026).

Історія статті:
Отримано: 31.05.2026 Доопрацьовано: 12.08.2026 Прийнято до друку: 23.09.2026 Опубліковано: 29.09.2026