4. АРХІТЕКТУРА ТА МЕТОДИ СИНТЕЗУ ПЛІС ПРИСТРОЇВ

4.1 Технології програмування апаратних засобів і області їх застосування

Інтегральні мікросхеми з логікою, що програмується (ПЛІС), або FPGA (field programmable gate arrays) є цифровими інтегральними мікросхемами (ІС), що складаються з програмованих логічних блоків і програмованих з'єднань між цими блоками. Можливість конфігурувати ці пристрої дозволяє інженерам-розробникам розв'язувати безліч різних задач.

Залежно від способу виготовлення, ПЛІС можуть програмуватися або один раз, або багатократно. Пристрої, які можуть програмуватися тільки один раз, називаються одноразово програмованими.

Словосполучення «field programmable», що міститься в розшифровці абревіатури FPGA, означає, що програмування FPGA-пристроїв виконується в конкретних («польових») умовах залежно від задачі, що розв'язується, (на відміну від пристроїв, внутрішня функціональність яких жорстко задана виробником). Воно може також означати, що FPGA-пристрої (ПЛІС) конфігуруються в лабораторних умовах, або свідчити про те, що йдеться про можливість модифікації функцій пристрою, вбудованого в електронну систему, яка вже використовується. Якщо пристрій може бути запрограмований, залишаючись у складі системи вищого рівня, він називається внутрішньосистемно програмованим [61].

Існує багата кількість різних типів цифрових мікросхем, у тому числі і такі, як «розсипна логіка» (невеликі компоненти, що містять декілька простих фіксованих логічних функцій), пристрої пам'яті і мікропроцесори. В даному випадку інтерес викликають програмовані логічні пристрої (ПЛП), спеціалізовані замовні інтегральні мікросхеми ASIC (application specific integrated circuit), спеціалізована інтегральна схема і ASSP (application specific standard parts) та ПЛІС. Причому термін ПЛП об'єднує два типи пристроїв: прості програмовані логічні пристрої (прості ПЛП) і складні програмовані логічні пристрої (складні ПЛП).

У російській технічній літературі ASIC прийнято називати «замовними інтегральними схемами (мікросхемами)», хоча сьогодні часто ASIC називають «спеціалізованими інтегральними схемами». Часто робиться уточнення, і мікросхеми поділяють на «замовні» і «напівзамовні».

Внутрішня архітектура ПЛП визначена виробником таким чином, що вони можуть бути конфігуровані (перепрограмовані) «на місці» для виконання самих різних функцій. На відміну від ПЛІС ці пристрої містять меншу кількість вентилів і використовуються для розв'язання невеликих і досить простих задач.

Разом з тим, існують замовні інтегральні схеми ASIC і ASSP, які містять сотні мільйонів логічних вентилів і можуть виконувати неймовірно великі і складні функцій. У основі ASIC і ASSP лежать ті самі конструкторські рішення, і у них та сама технологія виробництва. Обидва типи пристроїв розробляються для використання у складі спеціальних додатків, але при цьому ASIC розробляються і виробляються за замовленням спеціалізованих компаній, а ASSP призначаються масовому користувачу.

Не дивлячись на те що пропоновані користувачу замовні мікросхеми відрізняються високим ступенем інтеграції, рівнем складності задач, що розв'язуються, і продуктивністю, їх розробка і виробництво є досить тривалим і дорогим процесом. До того ж остаточний варіант схеми «заморожується в кремнії», і для її модифікації потрібне створення нової версії.

Таким чином, ПЛІС займають проміжне положення між ПЛП і замовними інтегральними схемами. З одного боку, їх функціональність може бути задана безпосередньо на місці відповідно до вимог замовника-користувача. З іншого боку, вони можуть містити мільйони логічних вентилів і, отже, реалізовувати надзвичайно великі і складні функції, які спочатку могли бути реалізовані тільки за допомогою замовних інтегральних схем.

Що стосується вартості ПЛІС, то вона набагато нижча за вартість замовних інтегральних схем (хоча остаточна версія замовної мікросхеми при масовому виробництві виявляється дешевшою). До того ж, у разі використання ПЛІС, внесення змін у цей пристрій не викликає особливих складнощів і навіть істотно скорочує терміни його виходу на ринок. Все це робить ПЛІС привабливими не тільки для крупних розробників, але і для невеликих новаторських конструкторських бюро. Іншими словами, «апаратні» або «програмні» ідеї окремих інженерів або невеликих груп інженерів можуть бути реалізовані у вигляді випробувальних стендів на основі ПЛІС без великих одноразових витрат на проектування або закупівлю дорогого оснащення, необхідного для розробки замовних мікросхем.

В даний час ПЛІС заповнюють п'ять крупних сегментів ринку [61]:

- замовні інтегральні схеми. Сучасні ПЛІС використовуються для створення пристроїв такого рівня, який до цього могли забезпечити тільки замовні мікросхеми;

- системи цифрової обробки сигналів. Високошвидкісна цифрова обробка сигналів традиційно здійснювалася за допомогою спеціально розроблених мікропроцесорів, так званих цифрових сигнальних процесорів (ЦСП) або digital signal processors (DSP). Проте сучасні ПЛІС містять вбудовані помножувачі, схеми арифметичного перенесення і великий обсяг оперативної пам'яті усередині кристала. Все це в поєднанні з високим ступенем паралелізму ПЛІС забезпечує їх перевагу над найшвидшими сигнальними процесорами в 500 і більше разів;

- системи на основі вбудованих мікроконтролерів. Нескладні задачі

управління звичайно виконуються вбудованими процесорами спеціального призначення, які називаються мікроконтролерами. Ці недорогі пристрої містять вбудовану програму, пам'ять команд, таймери, інтерфейси вводу/виводу, що розташовані поряд з ядром на одному кристалі. Найпростіші ПЛІС можна використовувати для реалізації програмного мікропроцесорного ядра з необхідними функціями вводу/виводу. В результаті ПЛІС стають все більш привабливими пристроями для реалізації функцій мікроконтролерів;

• мікросхеми, що забезпечують фізичний рівень передачі даних. ПЛІС вже давно використовуються як зв'язуюча логіка, що виконує функцію інтерфейсу між мікросхемами, які реалізують фізичний рівень передачі даних, і вищими рівнями мережних протоколів. Той факт, що сучасні ПЛІС можуть містити велику кількість високошвидкісних передавачів, означає, що мережні і комунікаційні функції можуть бути реалізовані в одному пристрої;

• системи з архітектурою, що перебудовується, або reconfigurable computing (RC). Можна застосовувати «апаратне прискорення» програмних алгоритмів, використовуючи такі властивості ПЛІС, як паралелізм та можливість перенастроювання. В даний час різні компанії зайняті створенням величезних перенастроюваних обчислювальних машин на основі ПЛІС. Такі системи можна використовувати для виконання широкого спектру задач — від моделювання апаратури до криптографічного аналізу або створення нових ліків.

Для програмування необхідний механізм, що дозволяє конфігурувати (програмувати) підготовлений кремнієвий кристал.

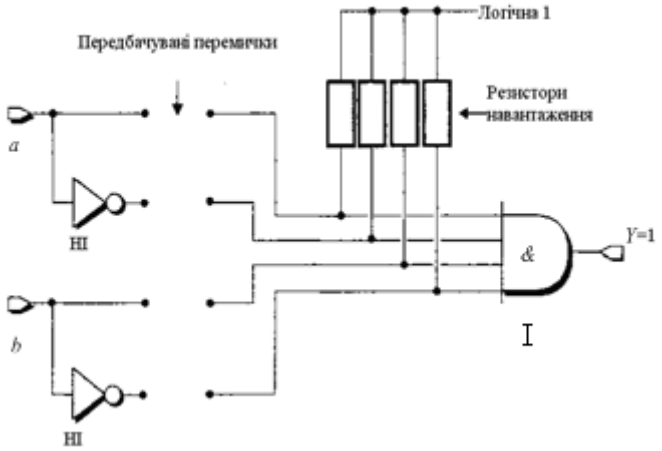


Рис. 4.1. Проста функція, що програмується

На рис. 4.1 наведено приклад простої функції, що програмується, з двома входами (*a* та *b*) та одним виходом. На входах присутні інвертори (операція НІ), а отже, кожен вхід може бути поданий або в прямій, або в інвертованій формі.

Особливої уваги заслуговує розташування передбачуваних перемичок. За відсутності перемичок на всі входи логічного елемента І через резистори навантаження буде подана логічна 1. Це означає, що вихід у даного пристрою завжди знаходитиметься в стані логічної 1. Щоб зробити просту програмовану функцію, необхідний якийсь механізм, що дозволив би встановлювати одну або декілька перемичок.

Одним з перших методів, який надав можливість користувачам програмувати власні пристрої, був і дотепер існує так званий метод плавких перемичок. При використанні цього методу пристрій виготовляється зі всіма можливими з'єднаннями, кожне з яких є плавкою перемичкою (рис. 4.2).

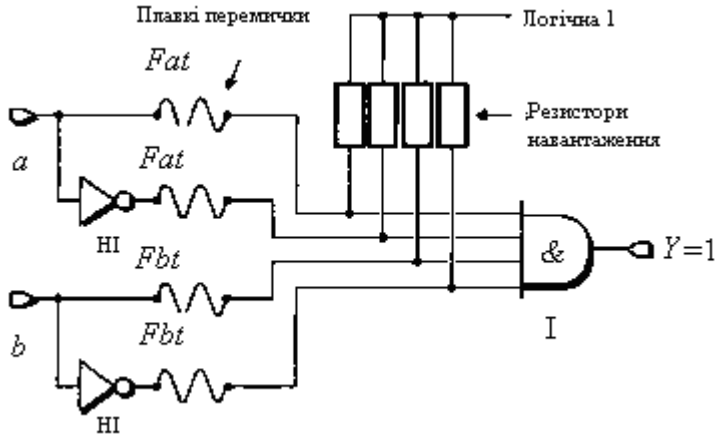


Рис. 4.2. Заповнення пристрою плавкими перемичками

Принцип дії цих перемичок аналогічний принципу дії запобіжників, які використовуються в побутовій техніці, наприклад в телевізорах. Якщо через непередбачені обставини телевізор починає споживати дуже велику потужність, це призведе до перегорання запобіжника. У результаті відбудеться розрив ланцюга (розрив провідника), який захистить пристрій від пошкодження.

Плавкі перемички мають мікроскопічні розміри, оскільки при їх виготовленні на кремнієвому кристалі використовуються ті ж технології, що і при виготовленні транзисторів і провідників.

Спочатку у програмованому пристрої на основі плавких перемичок всі перемички цілі. Це означає, що в незапрограмованому стані на виході функції, яка розглядається тут як приклад, завжди знаходитиметься рівень логічного 0. Наявність рівня логічного 0 хоча б на одному з входів вентиля І приведе до установлення 0 на виході. Так, якщо на вході *a* встановлений 0, та на виході вентиля І також буде 0. Аналогічно, якщо на вході *a* встановлена логічна 1, то вихід вентиля, який позначатимемо як *a*, буде встановлений в 0, і, отже, на виході вентиля І буде 0. Така ж ситуація має місце при використанні входу *b*.

Вся суть полягає у тому, що розробники можуть вибірково видаляти непотрібні плавкі перемички, подаючи на входи пристрою імпульси відносно високої напруги і великого струму. Розглянемо, наприклад, що відбудеться, якщо видалити плавкі перемички *Fat* і *Fbt* (рис. 4.3).

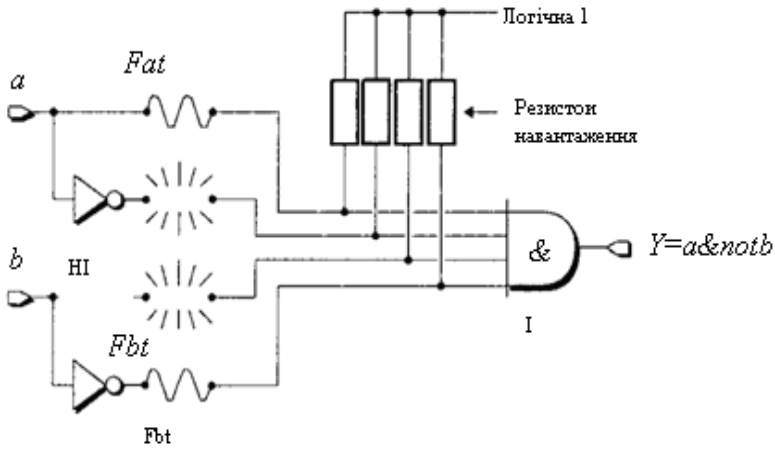


Рис. 4.3. Плавкі перемички після програмування

Видалення цих перемичок приведе до від'єднання інвертованого входу *a* і прямого входу *b* від логічного елемента І (через резистори навантаження на цих входах встановлюється значення логічної 1). Ця обставина вимушує пристрій формувати нову функцію $y = a \,\&\, \text{not}\, b$.

Такий процес видалення плавких перемичок зазвичай називається процесом програмування пристрою, але може також називатися перепалом перемичок або пропалюванням пристрою.

Пристрої, конфігурацію яких засновано на методі плавких перемичок, є одноразово програмованими пристроями (OTP — one-time programmable),

оскільки після перепалу плавка перемичка не може бути замінена або повернена в первинний стан.

Для різних ПЛІС використовуються різні методи програмування, але метод плавких перемичок до сучасних ПЛІС не застосовують. Він наведений як елементарний приклад.

Метод нарощуваних перемичок протилежний методу плавких перемичок. У цьому випадку кожне з'єднання, що конфігурується, має лінію зв'язку, що називають нарощуваною перемичкою. У незапрограмованому стані нарощувана перемичка має такий високий опір, що її можна розглядати як розімкнений ланцюг (розрив провідника, рис. 4. 4).

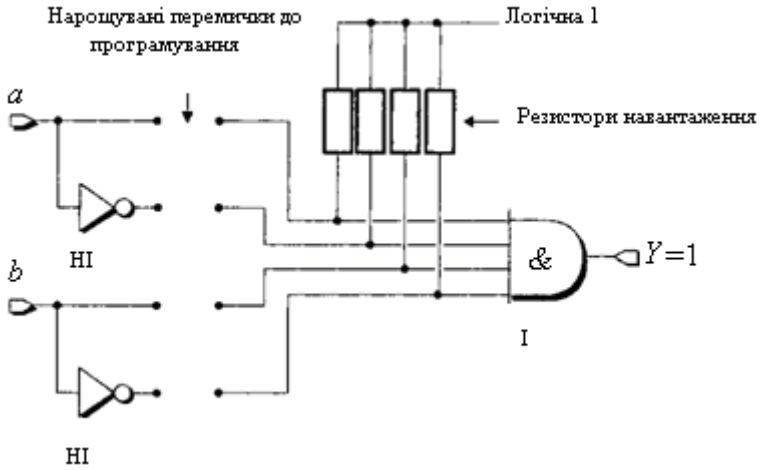


Рис. 4.4. Нарощувані перемички до програмування

Проте перемички можуть вибірково «нарощуватися» (програмуватися) за допомогою імпульсів відносно високої напруги і великого струму, що подаються на входи пристрою. Наприклад, якщо додати нарощувані перемички в ланцюг інвертованого входу *a* і прямого входу *b*, пристрій реалізує функцію *y* = not *a* & *b* (рис. 4.5).

Спочатку нарощувана перемичка являє собою мікроскопічний стовпчик аморфного (некристалічного) кремнію, що зв'язує два металеві провідники. У такому «незапрограмованому» стані аморфний кремній поводиться як діелектрик з опором, що досягає мільярда Ом (рис. 4.6, а).

Процес програмування окремого елемента полягає у вирощуванні зв'язку, що називається з'єднанням, і здійснюється шляхом перетворення аморфного кремнію-ізолятора в полікристалічний кремній, що проводить електричний струм. (рис. 4.6, *б*).

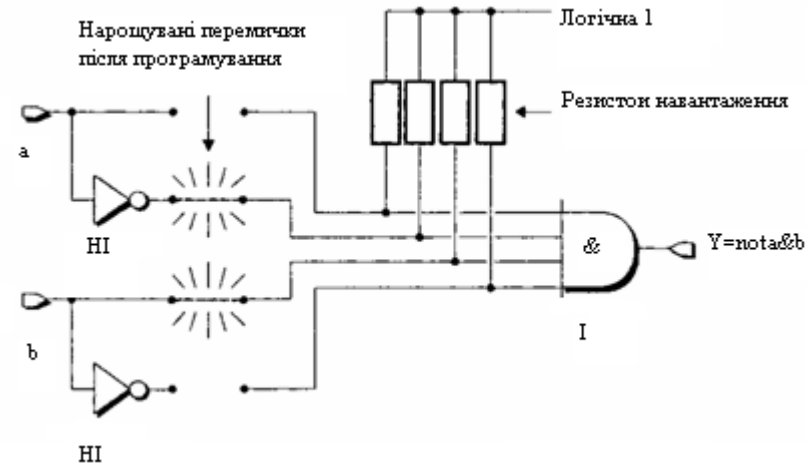


Рис. 4.5. Нарощувані перемички після програмування

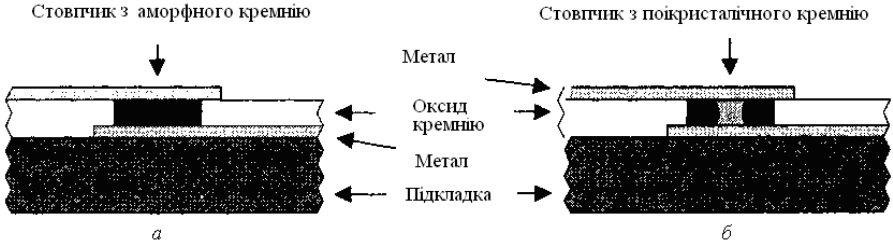


Рис. 4.6. Вирощування перемичок:
а – до програмування; б – після програмування

Пристрої, конфігурація яких здійснена методом нарощуваних перемичок, є однократно програмованими тому, що перемичка не може бути зруйнована або повернена до початкового стану і «перепрограмована».

Електронні системи загалом і комп'ютери зокрема використовують два типи запам'ятовуючих пристроїв: постійний запам'ятовуючий пристрій (ПЗП) і оперативний запам'ятовуючий пристрій (ОЗП).

ПЗП називають незалежними пристроями, оскільки інформація, що зберігається в них, не руйнується при відключенні живлення системи. Інші компоненти системи можуть зчитувати інформацію з пристроїв ПЗП, але не можуть записувати в них нові дані. На відміну від ПЗП, в ОЗП дані можуть

як записуватися, так і зчитуватися. Такі пристрої є енергозалежними, оскільки при відключенні живлення вся інформація, що зберігається в ОЗП, буде втрачена.

Типові мікросхеми ПЗП також називаються програмованими фотошаблоном, або масочно-програмованими, оскільки будь-які дані, що містяться в них, жорстко прошиваються у процесі виробництва за допомогою фотошаблона. Фотошаблони використовуються для створення транзисторів і металевих провідників, що з'єднують їх, які також називаються шарами металізації.

Розглянемо, наприклад, транзисторну комірку ПЗП, яка може зберігати один біт даних **(**рис. 4. 7**).** Стандартний пристрій ПЗП складається з деякої кількості рядків (адреси) і стовпців (дані), які разом утворюють масив даних. До кожного стовпця підключений один резистор навантаження, який дозволяє підтримувати на виході стовпця рівень логічної 1**,** а в кожному перетині рядка і стовпця присутній транзистор і, при необхідності, перемичка. Наявність/відсутність перемички задається фотошаблоном.

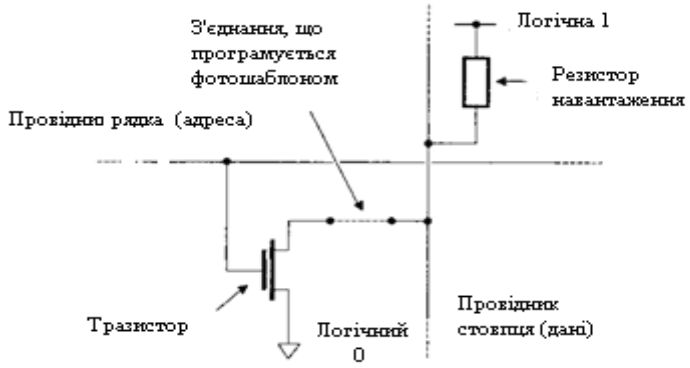


Рис. 4.7. Транзисторна комірка ПЗП, що програмується фотошаблоном

Більшість мікросхем ПЗП спочатку конфігурують для масового користувача. Коли необхідно конфігурувати пристрій відповідно до особливих вимог використовується індивідуально виготовлений фотошаблон, за допомогою якого визначаються комірки з перемичками і комірки без перемичок.

При подачі активного сигналу на провідник рядків цей рядок намагається перевести всі підключені до нього транзистори у відкритий стан. У разі, коли комірка містить запрограмоване за фотошаблоном з'єднання, транзистор цієї комірки, що активується, з'єднує провідник стовпця з рівнем

логічного 0, встановлюючи таким чином значення виходу в 0. І навпаки, якщо в комірці немає запрограмованого з'єднання, транзистор не виконуватиме будь-якої дії і через резистор навантаження, приєднаний до стовпця, підтримуватиме на виході мікросхеми рівень логічної 1.

Недолік масочно-програмованих пристроїв полягає у тому, що їх виробництво є досить дорогим задоволенням, якщо тільки не передбачається їх виробництво в дуже великих кількостях. Крім того, вони досить рідко використовуються при розробці апаратури, коли виникає необхідність часто змінювати склад компонентів.

Перші програмовані постійні запам'ятовуючи пристрої (ППЗП) були розроблені компанією Harris Semiconductor. Для створення цих пристроїв використовувався метод плавких перемичок з ніхрому.

Спрощену схему комірки ППЗП на основі транзистора з плавкою перемичкою наведено на рис. 4.8.

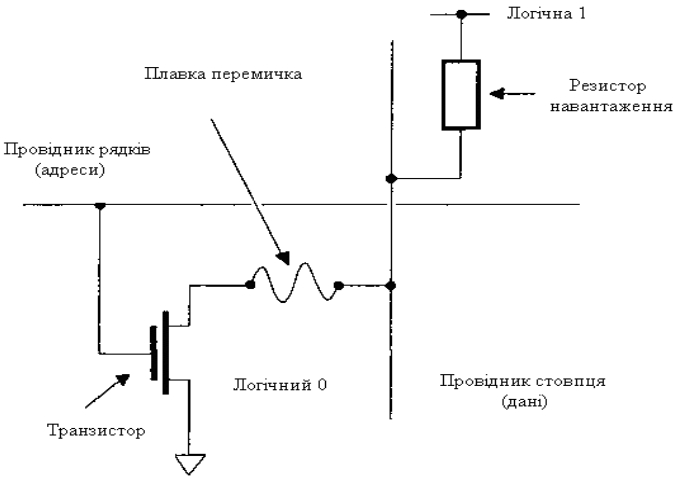


Рис. 4.8. Комірка ППЗП на основі транзистора з плавкою перемичкою

Мікросхема, що надходить від виробника, перебуває в незапрграмованому стані, але при цьому вона містить всі передбачені конструкцією плавкі перемички. В цьому випадку при переключенні рядка в активний стан включатимуться всі транзистори, під'єднані до цього рядка, вимушуючи всі стовпці знижувати рівень вихідної напруги до логічного 0 через відповідні їм транзистори. Інженер-розробник може на свій розсуд вибірково видаляти непотрібні плавкі перемички, подаючи на вхід пристрою імпульси відносно високої напруги і великого струму. При видаленні плавкої перемички на виході комірки формуватиметься рівень логічної 1.

Ці мікросхеми спочатку призначалися для використання як пристрої пам'яті, тобто для зберігання комп'ютерних програм і значень постійних величин (звідси абревіатура ПЗП). Проте розробники знайшли їм корисне застосування при реалізації простих логічних функцій, таких як, таблиці відповідності і кінцеві автомати.

З часом з'явилися різні ПЛП загального призначення, засновані на методах пропалюваних і нарощуваних перемичок.

Пристрої на основі пропалюваних або нарощуваних перемичок можуть бути запрограмовані тільки одноразово. Тому внесення в систему яких-небудь змін після перепалу або нарощування перемички вимагає великих витрат часу. В деяких випадках можна модифікувати пристрій шляхом пропалення або нарощування ще не модифікованих початкових перемичок, але можливість скористатися цією властивістю випадає вкрай рідко. У зв'язку з цим виникла ідея створення таких пристроїв, які можна було б програмувати, стирати і знов програмувати, тобто перепрограмувати.

Одним з варіантів реалізації цієї ідеї став ППЗП із стиранням (СППЗП). Перший такий пристрій був розроблений в 1971 році компанією Intel, яка привласнила йому назву 1702.

СППЗП-транзистор має таку ж структуру, як стандартний МОП-транзистор але з додатковим (другим) плаваючим затвором з полікристалічного кремнію, ізольованого шарами оксиду кремнію (рис. 4.9).

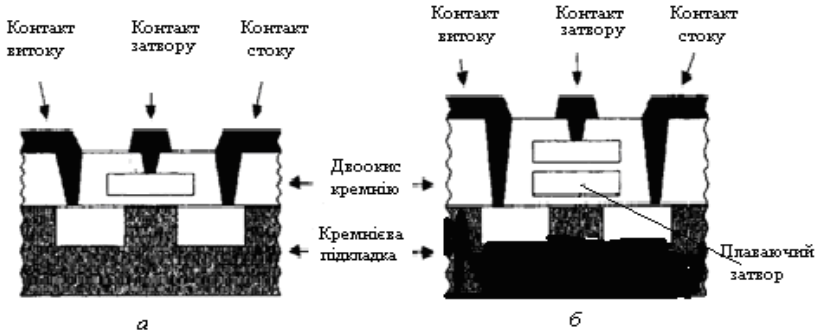


Рис. 4.9. Порівняння МОП та СППЗП транзисторів:
а – МОП-транзистор; б – СППЗП-транзистор

У незапрограмованому стані плаваючий затвор не заряджений і не впливає на роботу звичайного затвора. Щоб запрограмувати транзистор, необхідно прикласти до контактів затвора і стоку польового транзистора відносна високу напругу, близько 12 вольт. При цьому транзистор різко включається, і швидкі електрони долають шар оксиду кремнію, прямуючи в плаваючий затвор. Цей процес відомий як інжекція гарячих,

або високо енергетичних, електронів. Після зняття сигналу програмування негативно заряджені частинки залишаються в плаваючому затворі. Їх заряд стабільний і при дотриманні правил експлуатації вони не розсіюються впродовж більше 10 років. Накопичені на плаваючому затворі заряди блокують нормальну роботу звичайного затвора, і таким чином дозволяють розрізняти запрограмовані комірки від незапрограмованих. Завдяки цій властивості такі транзистори можна використовувати для формування елементів пам'яті (рис. 4.10).

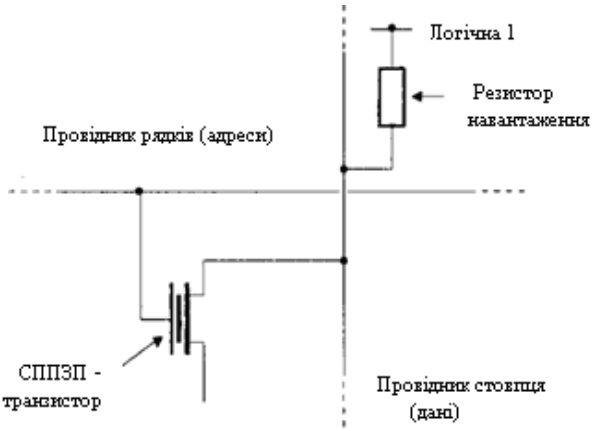


Рис. 4.10. Комірка пам'яті на основі СППЗП-транзистора

Такий елемент пам'яті більше не потребує плавких перемичок, нарощуваних перемичок або програмованих фотошаблоном з'єднань. У незапрограмованому стані, який забезпечується виробником, всі плаваючі затвори у СППЗП-транзисторах будуть не заряджені. В цьому випадку при переведенні провідника рядків в активний стан включатимуться всі транзистори, приєднані до цього рядка, примушуючи всі провідники стовпців скидати сигнал на виході до рівня логічного 0 через відповідні транзистори. Для програмування можуть використовуватися входи пристрою для заряду плаваючих затворів обраних транзисторів, тим самим блокуючи їх роботу. У цих випадках в елементах пам'яті зберігатимуться логічні 1.

З погляду розташування на кремнієвому кристалі елемента пам'яті СППЗП більш ефективні, оскільки їх розміри істотно менші, ніж розміри їх аналогів на плавких перемичках. Однак основною перевагою таких елементів пам'яті є можливість стирання та перепрограмування. Стирання елементів пам'яті є не що інше, як «витікання» електронів з плаваючого затвора. Енергія, необхідна для витікання електронів, забезпечується за допомогою

джерела ультрафіолетового (УФ) випромінювання. Мікросхеми СППЗП поставляються в керамічному або пластиковому корпусі, зверху якого є невелике кварцове вікно. Це вікно звичайно закрите шматочком непрозорої клейкої стрічки. Щоб стерти мікросхему, спочатку треба витягнути її з монтажної плати, відкрити кварцове вікно і потім помістити в контейнер, що закривається, з потужним джерелом УФ-випромінювання.

Головними недоліками СППЗП є їх дорогий корпус з кварцовим вікном і час, необхідний для їх очищення, який складає приблизно 20 хвилин. Проблема, яку доведеться вирішувати найближчим часом, пов'язана з удосконаленням технологічного процесу, який дозволить виконувати транзистори меншого розміру. Через те, що напівпровідникові структури стають меншими і їх густина, тобто кількість транзисторів і з'єднань, збільшується, великий відсоток поверхні кристала виявляється покритим металом. Це утрудняє поглинання УФ-випромінювання комірками СППЗП і збільшує час експозиції, що необхідний для стирання.

Дані мікросхеми спочатку призначалися для використання як програмовані постійні запам'ятовуючи пристрої, що знайшло віддзеркалення в їх назві — ППЗП. Пізніше цю технологію стали застосовувати в більш універсальних ПЛП, які одержали назву ПЛП із стиранням.

Наступний ступень технології являє собою ППЗП з електричним стиранням (ЕСППЗП). Комірка пам'яті ЕСППЗП наведена на рис. 4.11.

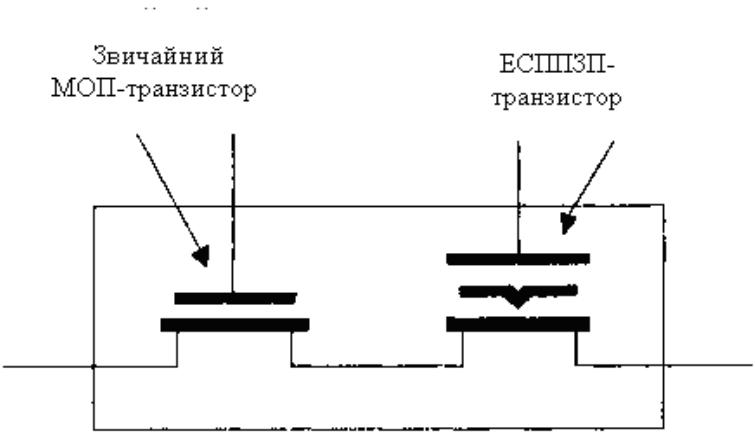


Рис. 4.11. Комірка пам'яті ЕСППЗП

Комірка ЕСППЗП приблизно в 2.5 рази більша, ніж еквівалентна їй комірка СППЗП, оскільки вона складається з двох віддалених один від одного транзисторів.

ЕСППЗП-транзистор має плаваючий затвор, але цей затвор оточений дуже тонким ізолюючим шаром оксиду кремнію. Другий транзистор може використовуватися для стирання елемента пам'яті електричним способом.

На початку свого існування ЕСППЗП-мікросхеми використовувалися як комп'ютерна пам'ять. Ця ж технологія була згодом застосована до пристроїв ПЛП, які стали називатися ПЛП з електричним стиранням (ЕСПЛП) або програмованими логічними інтегральними схемами (ПЛІС).

Технологія, відома як Flash (спалах, мить, миттєвість), починалась від технології виготовлення як СППЗП, так і ЕСППЗП. Спочатку назва Flash була присвоєна пристроям, виготовленим за цією технологією, щоб відобразити характерний для них надзвичайно малий час стирання в порівнянні з пристроями СППЗП. Компоненти, виконані за Flash-технологією, можуть бути реалізовані в різних типах архітектури. Одні пристрої можуть мати елементи пам'яті, виконані на одному транзисторі з плаваючим затвором такої ж площі, як в комірках СППЗП, але з набагато тоншими ізолюючими шарами оксиду кремнію. Такі пристрої можуть стиратися електричним способом, але при цьому тільки шляхом очищення всього пристрою або більшої його частини. Інші типи пристроїв побудовані на елементах пам'яті з двома транзисторами, подібно до мікросхемам ЕСППЗП, що дозволяє стирати або перепрограмувати інформацію послівно.

Перші версії Flash-пристроїв могли зберігати тільки один біт інформації на комірку. Але вже до початку 2002 року технологи виконали ряд експериментів по збільшенню місткості комірок. Виявилося, що, згідно з одним методом, запам'ятовування двох бітів в одній комірці можливе завдяки зберіганню різних рівнів зарядів в плаваючих затворах Flash-транзисторів, а згідно з іншим методом – завдяки створенню двох дискретних запам'ятовуючих вузлів в шарі під затвором.

Існують два основні типи напівпровідникових пристроїв оперативної пам'яті: динамічний ОЗП і статичний ОЗП. У разі динамічного ОЗП кожен елемент пам'яті формується за допомогою пари транзистор-конденсатор, яка займає мало місця на поверхні кремнієвого кристала. Визначення «динамічне» відображає той факт, що з часом конденсатор втрачає свій заряд, тобто для збереження даних кожна комірка повинна періодично перезаряджатися. Ця операція називається регенерацією. Вона є достатньо складною і вимагає значної кількості додаткових схемних рішень. Застосування цих мікросхем виявляється виправданим, якщо «вартість» ланцюгів регенерації покривається десятками мільйонів біт в одній мікросхемі динамічного ОЗП. Проте з погляду програмованої логіки технологія динамічного ОЗП не викликає великого інтересу.

Визначення «статичне» у назві статичного ОЗП відображує те, що значення, однократно записане до комірки пам'яті, буде залишатися

незмінним доти, поки не буде спеціально змінено, або поки система не буде відключена від електроживлення. Позначення комірки пам'яті, що програмується на основі статичного ОЗП, наведено на рис. 4.12.

Рис. 4.12. Програмована комірка пам'яті на основі статичного ОЗП

Комірка містить мультитранзисторний елемент статичного ОЗП, вихід якого підключений до додаткового управляючого транзистора. Залежно від вмісту (логічний 0 або логічна 1) елементу пам'яті управляючий транзистор буде закритий (тобто відключений) або відритий (тобто включений). Один з недоліків програмованих пристроїв на основі елементів пам'яті статичного ОЗП полягає у тому, що кожна комірка займає значну площу на поверхні кремнієвого кристала, оскільки складається з чотирьох або шести транзисторів, конфігурованих у вигляді регістра-клямки. Іншим недоліком є те, що дані про конфігурацію пристрою будуть втрачені при відключенні живлення системи.

Програмовані пристрої і пов'язані з ними технології програмування наведено у табл.. 4.1 [61].

Таблиця 4.1 – Технології програмування

Технології	Умовне позначення	Галузь використання
Плавкі перемички		Прості ПЛП
Нарощувані перемички		ПЛІС
СППЗП		Прості та складні ПЛП
ЕСППЗП та Flash		Прості та складні ПЛП (деякі ПЛІС)
Статичний ОЗП	SRAM	ПЛІС (деякі складні ПЛП)

Треба мати на увазі, що постійно з’являються нові технології. Одна з таких технологій називається магнітним ОЗП (MRAM — magnetic RAM). Ця

технологія почалася з того, що фірма IBM розробила так званий магнітний тунельний перехід. Йшлося про тришарову структуру, що складається з двох феромагнітних шарів, розділених тонким ізолюючим шаром. Комірки магнітного ОЗП могли створюватися на перетині двох провідників, провідника рядків і провідника стовпців, з магнітним тунельним переходом між ними.

Комірки магнітного ОЗП — це висока швидкість роботи статичного ОЗП, великий обсяг динамічного ОЗП, незалежність Flash-технології і мінімальна кількість енергії, що споживається.

Класифікація архітектурних особливостей ПЛП наведена на рис. 4.13.

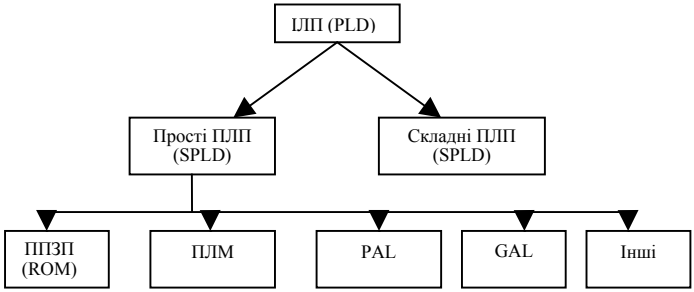


Рис.4.13. Класифікація архітектури побудови ПЛП

Першими представниками простих ПЛП були ППЗП-мікросхеми. Для розуміння роботи цих пристроїв краще всього уявити їх такими, що складаються з фіксованого масиву логічних функцій І, приєднаного до програмованого масиву логічних функцій АБО.

Для прикладу розглянемо роботу ППЗП з трьома входами і трьома виходами (рис. 4.14).

Як програмовані зв'язки в масиві елементів АБО можуть застосовуватися плавкі перемички або СППЗП-транзистори і комірки ЕСППЗП відповідно для мікросхем СППЗП і ЕСППЗП. Рис. 4.14 дає лише загальне уявлення про принцип дії даного пристрою і не відображає реальну діаграму з'єднань. В дійсності кожна функція І визначеного масиву має три входи, які приєднані до істинних або інвертованих входів a, b або c пристрою. Аналогічно кожна функція АБО програмованого масиву має вісім входів, які приєднані до виходів масиву функцій І.

Мікросхеми ППЗП спочатку призначалися для зберігання програмних інструкцій і значень констант, тобто для виконання функцій комп'ютерної пам'яті. Проте розробники використовують їх також для реалізації простих логічних функцій, таких як таблиці відповідності або кінцеві автомати. У

дійсності мікросхеми ППЗП можуть використовуватися для реалізації будь-якого блока комбінаційної логіки, або комбінаційного пристрою, за умови, що він має невелику кількість входів і виходів. Наприклад, простий ППЗП з трьома входами і трьома виходами, що показаний на рис. 4.14**,** може використовуватися для синтезу будь-якої комбінаційної функції з не більш ніж трьома вхідними і трьома вихідними параметрами. Щоб зрозуміти, як він працює, розглянемо невеликий логічний блок, наведений на рис. 4.15**.** Слід мати на увазі, що ця схема має сенс тільки для даного прикладу.

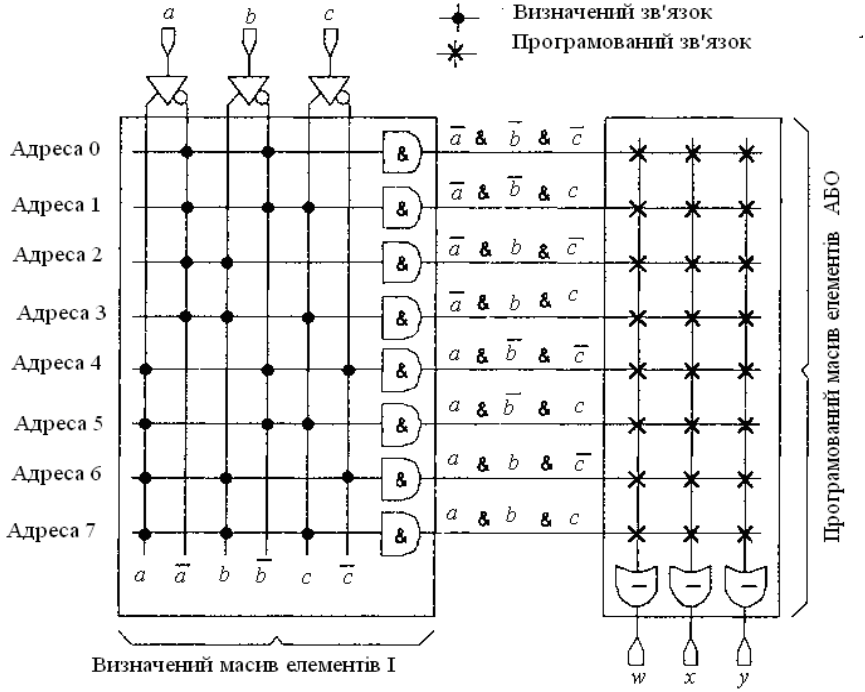
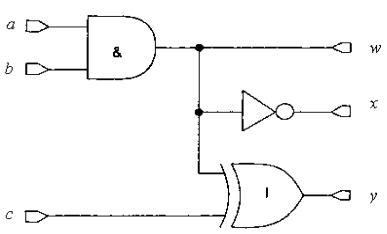


Рис. 4.14. Незапрограмована мікросхема ППЗП



a	b	c	w	x	y
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Рис. 4.15. Блок комбінаційної логіки

Логічний блок (рис. 4.15) може бути замінений мікросхемою ППЗП з трьома входами і трьома виходами. Для цього треба всього лише запрограмувати відповідні зв'язки в масиві логічних елементів АБО (рис. 4.16).

У виразах на цьому рисунку використані такі позначення: «&» — операція І (AND), «|» — операція АБО (OR), «^» — операція ВИКЛЮЧНОГО АБО (XOR), «¯» — операція НІ (NOT).

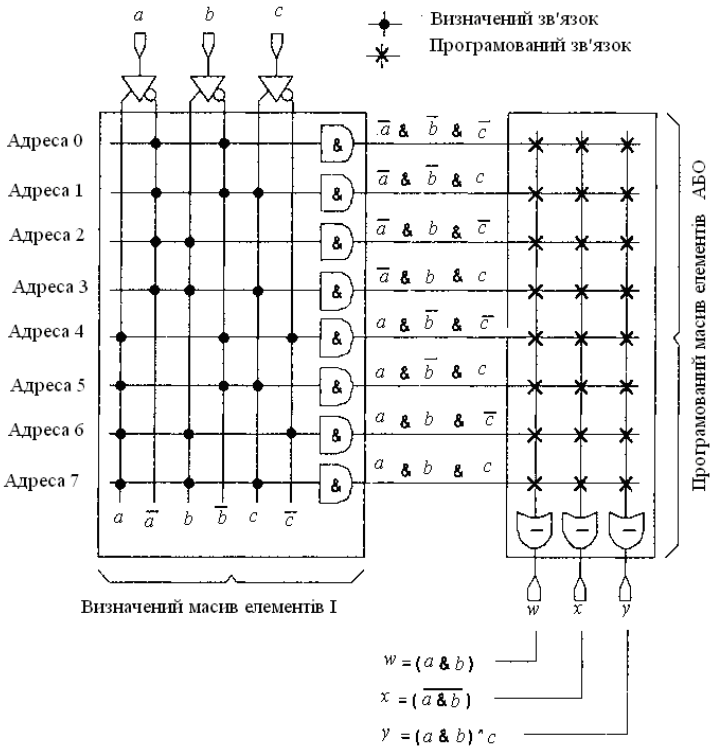


Рис. 4.16. Запрограмована мікросхема ППЗП

Розглянута вище мікросхема ППЗП є дуже простою. Насправді мікросхеми ППЗП мають значно більше входів і виходів і можуть використовуватися для реалізації великих блоків комбінаційної логіки. Комбінаційна логіка в загальному випадку реалізовувалася за допомогою мікросхем таких як, наприклад, 74-та серія компанії Texas Instruments [61].

Той факт, що досить велике число таких мікросхем можна замінити однією мікросхемою ППЗП, означає, що монтажну плату можна зробити меншою, простішою, дешевшою і менш схильною до збоїв (кожне паяне з'єднання на монтажній платі — потенційно збійний вузол). Крім того, якщо в цій частині системи виявиться логічна помилка, наприклад якщо розробник помилково використовував функцію І замість І-НІ**,** то вона може бути легко виправлена шляхом прошивання нової ППЗП-мікросхеми або за допомогою очищення і перепрограмування СППЗП або ЕСППЗП-мікросхем. Цей спосіб більш прийнятний, ніж спосіб виявлення помилок на друкарській платі, що виконана на основі окремих мікросхем.

У логічних виразах оператор І («&») називається логічним множенням або добутком, а оператор АБО («|») — логічним складанням або сумою. Проте, коли йдеться про логічний вираз вигляду

$$y = (a \& b \& c) | (a \& b \& c) | (a \& b \& c) | (a \& d \& c),$$

поняття «літерал» означає або істинну, або інвертовану змінну (a, b і т. д.), а група літералів, що зв'язана оператором «&», називається добутком. Таким чином, добуток (*a* & *b* & *c*) складається з трьох літералів, а саме з *a*, *b* і *c*, і наведене вище рівняння буде сумою добутків.

Суть полягає у тому, що коли ці вирази використовуються для реалізації комбінаційної логіки (рис. 4.15 і 4.16), ППЗП зручно використовувати для виразів з великою кількістю добутків, але невеликою кількістю входів, оскільки всі вхідні комбінації жорстко зашиті в матриці І та завжди декодуються.

4.2. Програмовані схеми ПЛМ, PAL, GAL — призначення, архітектура, технології

Наступний ступінь розвитку ПЛП — вирішення проблем, пов'язаних з обмеженнями, властивими архітектурі ППЗП [61].

Перші програмовані логічні матриці (ПЛМ) з'явилися приблизно в 1975 році. Більшість користувачів застосовувала їх як прості ПЛП, оскільки обидва масиви, тобто і масив функцій І, і масив функцій АБО, були програмованими.

Якщо узяти просту ПЛМ з трьома входами і трьома виходами в незапрограмованому стані (рис. 4.17), то**,** на відміну від ППЗП, можна побачити, що кількість функцій І в однойменному масиві не залежить від кількості входів матриці. При цьому додаткові функції І можуть бути сформовані шляхом простого додавання в масиві рядків.

Аналогічна кількість функцій АБО в однойменному масиві не залежить від кількості входів матриці і від розміру масиву функцій І.

Додаткові функції АБО можуть бути сформовані шляхом простого додавання в цьому масиві стовпців.

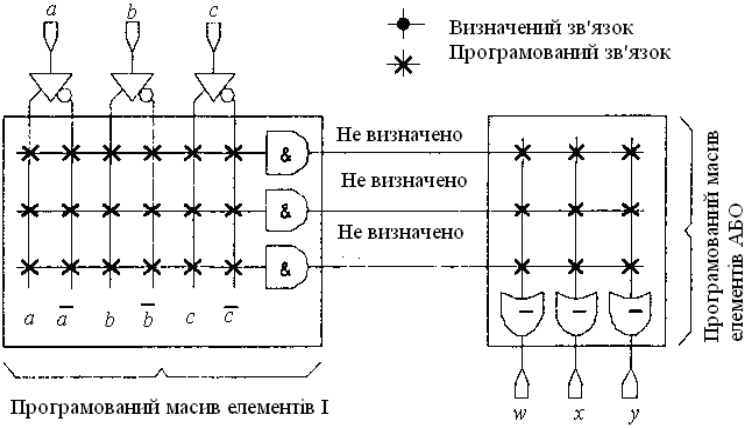


Рис.4.17. Незапрограмована ПЛМ

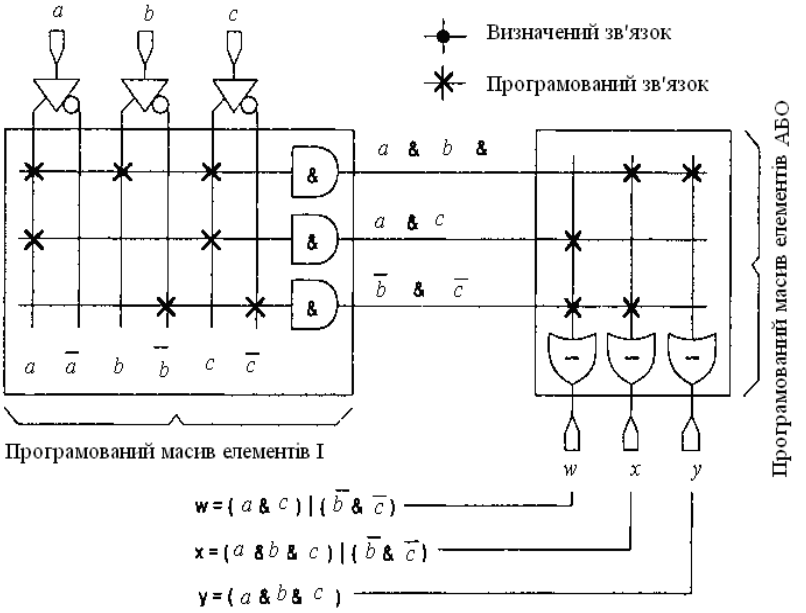


Рис. 4.18. Запрограмована ПЛМ

Припустимо, що ПЛМ повинна реалізувати наступні три рівняння:

w= (a & c) | (b & c), x = (a & b & c) | (b & c) та y = (a & b & c).

Розв’язати цю задачу можна шляхом програмування відповідних зв'язків, як показано на рис. 4.18.

У ПЛМ немає жорсткої необхідності підключати масив функцій АБО до виходу масиву функцій І. Тому деякі альтернативні конфігурації, наприклад масив функцій І, підключений до входу масиву АБО-НІ, іноді показував непогані результати. Проте, не дивлячись на теоретичну можливість реалізації на практиці, функції АБО-І, І-НІ-АБО, І-НІ-АБО-НІ відносно рідко зустрічалися або просто не існували. Одною з причин, з якої ПЛМ продовжують працювати з архітектурою І-АБО чи І-АБО-НІ, стало те, що вираз «сума добутків» найчастіше використовується в логічних рівняннях. Інші типи рівнянь, такі, як добуток сум, зводяться до них за допомогою стандартних методів алгебри. Таке перетворення звичайно виконується за допомогою програм.

Мікросхеми ПЛМ рекламувалися як дуже корисні пристрої для великих схем, логічні рівняння яких характеризувалися великою кількістю однакових добутків. Наприклад (рис. 4.18), добуток вигляду (*b* & *c*) використовується двічі: для обчислення виходів *x* та *y*. Ця властивість називається розподілом добутків.

Разом з тим для проходження через програмовані зв'язки сигналам потрібно набагато більше часу, ніж при проходженні через їх визначені аналоги. Тому ПЛМ працює повільніше ніж ППЗП, оскільки обидва масиви (і функції І, і функцій АБО) є програмованими.

Для того щоб розв'язати проблему швидкодії, властивої програмованим логічним матрицям, з'явився новий клас пристроїв, що має назву «програмований масив логіки» (PAL — Programmable Array Logic). На понятійному рівні пристрої PAL майже повністю протилежні мікросхемам ППЗП, оскільки вони складаються з програмованого масиву логічних функцій І і визначеного масиву функцій АБО. Пристрої GAL (Generic Array Logic — змінний масив логіки), розроблені компанією Lattice Semiconductor Corporation, є більш складними КМОП-різновидами ідеології PAL з електричним стиранням [61].

Як приклад розглянемо простий PAL-пристрій з трьома входами і трьома виходами (рис. 4.19). Перевагою мікросхем PAL (у порівнянні з програмованими логічними матрицями) є вища швидкість, оскільки з двох масивів у них тільки один програмований.

Разом з тим програмований масив логіки більш обмежений щодо функціональності, оскільки він дозволяє складати тільки обмежену кількість добутків.

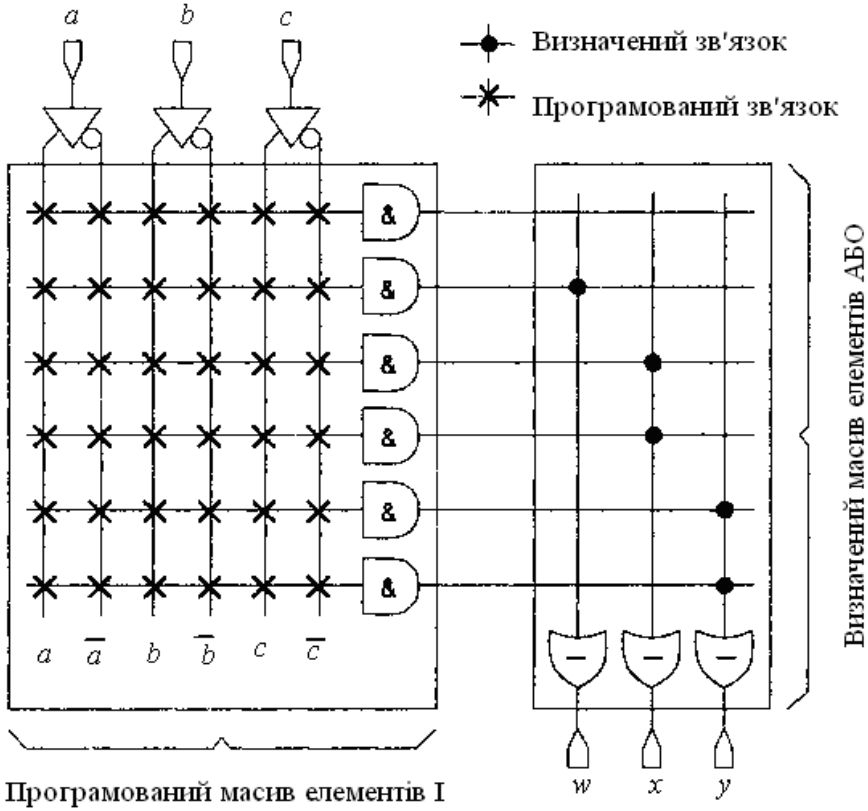


Рис. 4.19. Запрограмований пристій PAL

Розглянуті вище як приклади ПЛМ і PAL насправді дуже великі, з безліччю входів, виходів і з внутрішніми сигналами. У них можуть бути передбачені різні додаткові програмовані опції, такі як можливість інвертувати виходи, або мати виходи з трьома станами, або і те, і інше. Крім того, деякі з них підтримують регістрові виходи, тобто виходи з клямкою, аналогічно програмованим мультиплексорам, які дозволяють користувачу встановлювати по черзі для кожного виводу, яку версію виходу використовувати — регістрову або нерегістрову. Деякі пристрої дозволяють конфігурувати певні виводи як або виходи, або додаткові входи.

Проблема полягає у тому, що різні пристрої можуть забезпечувати різні набори опцій, що ускладнює вибір оптимального пристрою для конкретного додатку. У подібних випадках інженери або обмежують себе у виборі пристроїв і підганяють свою конструкцію під ці пристрої, або

використовують комп'ютерні програми, за допомогою яких вибирають пристрій максимально задовольняюче їх вимогам.

Типове задача для електроніки полягає в тому, щоб розробити пристрій з максимальними функціональними можливостями та мінімальними розмірами, ціною та потужністю, що споживається.

У зв'язку з цим в кінці 70-х — початку 80-х з'явилися складніші програмовані логічні пристрої, так звані складні ПЛП (CPLD — complex PLD) [61].

Лідерами були винахідники оригінальних пристроїв PAL — розробники компанії Monolithic Memories (MMI), які представили компонент під назвою Мега-PAL (Mega-PAL). Цей пристрій з 84 виходами по суті включав 4 PAL-пристрої і з'єднання між ними. Але Мега-PAL споживала непропорційне багато енергії, і давала невелику перевагу в порівнянні з використовуванням чотирьох окремих пристроїв [61].

Істотний прорив припав на 1984 рік, коли компанія Altera представила складний ПЛП, заснований на поєднанні КМОП- і СППЗП-технологій. Використовування КМОП дозволило компанії Altera досягти неймовірної функціональної густини і складності при порівняно невеликому споживанні енергії. Основою для програмування цих пристроїв були комірки СППЗП, і саме це зробило їх ідеальними для використовування при розробці і створенні прототипів устаткування.

Компанія Altera заслужила свою славу не тільки завдяки комбінації технологій КМОП і СППЗП. При нарощуванні архітектури простих ПЛП до рівня великих пристроїв, подібних Мега-PAL, інженери вважали, що центральний комутаційний масив, що з'єднує індивідуальні блоки простих ПЛП, відомий також як програмована комутаційна матриця, повинен на 100% з'єднуватися зі всім входам і виходам блоків. В результаті це приводило до істотного зниження швидкості, підвищення розсіюваної потужності і збільшення вартості компонентів.

Altera вдалося зробити технологічний прорив, використовуючи центральний комутаційний масив з кількістю з'єднань з входами/виходами блоків менше 100%**.** Це ускладнило програмне забезпечення для проектування ПЛП, але швидкість, споживана потужність і вартість цих пристроїв були цілком прийнятними [61].

Не дивлячись на те, що кожен виробник складних ПЛП реалізовував свої унікальні технології, в загальному випадку пристрій складався з декількох блоків простих ПЛП, зазвичай PAL, об'єднаних загальною програмованою комутаційною матрицею (рис. 4.20). Крім окремих блоків простих ПЛП можна було також запрограмувати з'єднання між ними за допомогою програмованої комутаційної матриці.

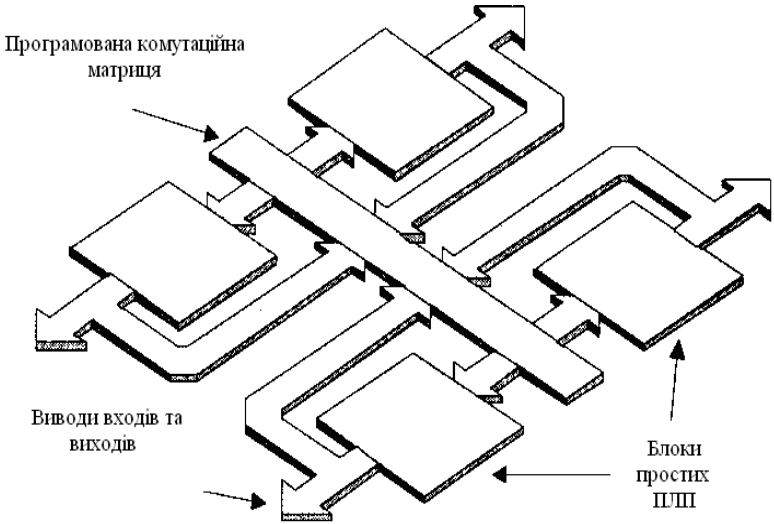


Рис. 4.20. Загальна структура складного ПЛП

На цьому рисунку не показані різні додаткові компоненти, він дає лише поверхневе уявлення про роботу складного ПЛП. Насправді всі структури пристрою сформовані на одному шматку кремнію. Наприклад, програмована комутаційна матриця може містити велику кількість провідників, наприклад 100. Але це більше, ніж може бути підключене до блока простого ПЛП, який здатний працювати тільки з обмеженою кількістю сигналів, наприклад 30. Блоки простих ПЛП пов'язані з комутаційною матрицею свого роду програмованими мультиплексорами (рис. 4.21).

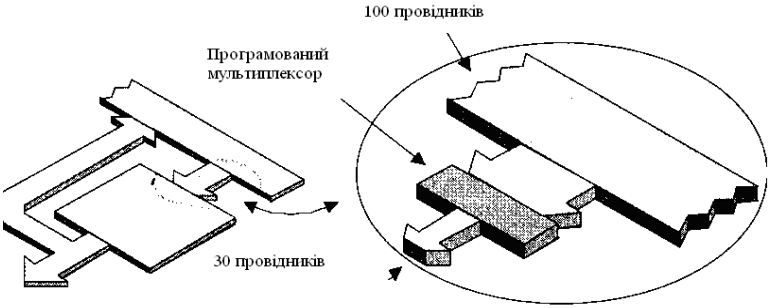


Рис. 4.21. Використання програмованого мультиплексора

Залежно від виробника і від типу пристрою, програмовані перемикачі складних ПЛП можуть бути виконані на елементах пам'яті типу СППЗП, ЕСППЗП, Flash або на статичному ОЗП. При використанні статичного ОЗП з'являється можливість збільшити універсальність цієї пам'яті, використовуючи її як програмовані перемикачі і як фактичну оперативну пам'ять.

У 1980 році комісія Об'єднаної інженерної ради з електронних пристроїв (JEDEC), підрозділ Асоціації електронної промисловості США, запропонувала стандарт форматів текстових файлів для програмування пристроїв ПЛП [61].

У цей же час Джон Беркнер (John Birkner), людина яка розробила перші пристрої PAL і яка стояла на чолі їх подальшого розвитку, створив PAL Асемблер (PALASM). PAL Асемблер поєднує в собі мову опису апаратних засобів (HDL — Hardware Description Language) і прикладне програмне забезпечення. Як мова опису апаратних засобів PAL Асемблер дозволяє інженерам-розробникам точно описати функції принципової схеми в текстовому файлі з початковими кодами. Файл складався з булевих рівнянь, записаних у форматі «сума добутків». PALASM читав текстовий файл з початковими кодами і автоматично генерував текстовий програмний файл, придатний для програмування пристрою.

Поява PALASM, поза сумнівом, була епохальною подією для того часу, але його первинні версії підтримували тільки пристрої компанії MMI (Monolithic Memories Inc) і не підтримували які-небудь види мінімізації або оптимізації. Для вирішення цих проблем в 1983 році компанія Data I/O представила мову ABEL (Advanced Boolean Expression Language — розширена мова логічних виразів). Приблизно в цей же час компанія Assisted Technology розробила пакет CU PL (Common Universal tool for Programmable Logic — Загальні універсальні засоби проектування для програмованої логіки). Обидва продукти поєднували в собі мову HDL і програмні додатки. Крім того, що вони підтримували конструкції кінцевих автоматів і алгоритми автоматичної мінімізації логіки, вони дозволяли працювати з численними типами ПЛП пристроїв різних виробників.

Окрім PALASM, ABEL і CUPL, які були, поза сумнівом, кращими з ранніх версій мови HDL, існувало багато інших версій, таких, як AMAZE (Automated Map and Zap of Equation — автоматичне перетворення і виправлення виразів) компанії Signetics. Ці прості мови і засоби проектування, що їх супроводжують, прокладали шлях для високорівневих версій мови HDL, таких, як Verilog або VHDL, і для засобів проектування, таких як логічний синтез, які в даний час використовуються для проектування сучасних замовних мікросхем і пристроїв з використанням ПЛІС [61].

4.3. Класифікація спеціалізованих замовних BIC (ASIC)

Замовні інтегральні мікросхеми (ASIC — application specific integrated circuit) представлені чотирма основними класами. У порядку збільшення складності такими пристроями є вентильні матриці (для вентильної матриці часто можна зустріти інший термін — базовий матричний кристал (БМК)), структуровані ASIC, схеми на стандартних елементах і повністю замовні мікросхеми (рис. 4.22) [61].

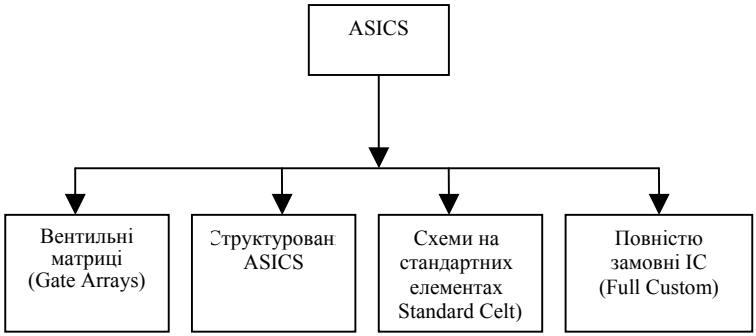


Рис. 4.22. Різні типи спеціалізованих замовних IC

Під час появи цифрових мікросхем існувало тільки два типи пристроїв, окрім, природно, мікросхем пам'яті. До першого типу належали відносно прості блоки, які вироблялися такими компаніями, як Texas Instruments або Fairchild, і які продавалися як стандартні компоненти. До другого типу належали замовні спеціалізовані мікросхеми, такі як, мікропроцесори, які розроблялися і вироблялися за замовленням.

При створенні замовних пристроїв розробка маски для кожного шару мікросхеми була прерогативою інженерів — розробників кінцевого пристрою. Постачальники спеціалізованих мікросхем заздалегідь не створювали ніяких компонентів на кремнієвому кристалі заводським способом і не поставляли бібліотек визначених логічних вентилів або функцій.

Використовуючи відповідні засоби проектування інженери, задавали розміри окремих транзисторів і на їх основі створювали функції більш високого рівня. Наприклад, при необхідності збільшити швидкість логічного вентиля вони могли змінити розміри транзисторів, що використовувалися для

побудови вентилів. Засоби проектування для розробки замовних пристроїв часто створювалися інженерами фірм-розробників.

Розробка замовних пристроїв є дуже складним і трудомістким процесом, але створені у такий спосіб мікросхеми містять необхідну кількість вентилів з мінімальними втратами вільного місця на кремнієвому кристалі.

4.4. Архітектура базисних модулів структурованих ASIC

У середині 60-х компанія Fairchild Semiconductor представила новий пристрій під назвою мікроматриця (Micromatrix). Цей пристрій складався з невеликої кількості, близько 100, ні до чого не підключених транзисторів. Для того щоб цей пристрій міг виконувати корисні функції, розробники уручну викреслювали на двох пластикових листах шари металізації для з'єднання транзисторів.

На першому листі зображалися провідники по осі У, тобто зверху вниз, за допомогою яких виготовлявся перший шар металізації. На другому листі зображалися провідники по осі Х, тобто зліва направо, для виготовлення другого шару металізації. Крім того, використовувалися додаткові листи для зображення перехідних отворів, що сполучають перший шар металізації з транзисторами, і для перехідних отворів, що сполучають перший і другий шари металізації.

Виготовлення пристрою у такий спосіб було вкрай трудомістким заняттям і сприяло виникненню помилок, але так чи інакше дорога і по-справжньому трудомістка робота із створення транзисторів виконувалася. Іншими словами, мікроматриця дозволяла розробникам створювати замовні пристрої, прийнятні за вартістю і термінами виготовлення .

Після декількох років, а точніше в 1967 році, компанія Fairchild представила новий пристрій, що одержав назву мікромозаїки (Micromosaic). Цього разу пристрій містив декілька сотень окремих транзисторів, які могли з'єднуватися разом, утворюючи близько 150 логічних елементів І, АБО і НІ. Унікальність мікромозаїки полягала у тому, що розробники могли задавати функції замовного пристрою, використовуючи логічні вирази у вигляді текстового файла, за допомогою якого комп'ютерна програма визначала необхідні міжтранзисторні з'єднання і виготовляла фотошаблон, за яким вже створювався пристрій. Такий підхід був справжньою революцією для того часу, і сьогодні мікромозаїка вважається передвісником сучасних вентильних матриць як різновиду спеціалізованих замовних мікросхем і першим справжнім додатком систем автоматизованого проектування (САПР).

Ідея створення матриць логічних елементів, або вентильних матриць, розглядалася ще в кінці 60-х в компаніях IBM, Fujitsu та ін. Проте перші

пристрої використовувалися тільки для внутрішнього споживання, і лише у середині 70-х вентильні матриці, виготовлені за технологією КМОП, стали доступні широкому кругу споживачів. Перші вентильні матриці називалися нескомутованими логічними матрицями (ULA — uncommitted logic array). З часом цей термін перестав вживатися [61].

Основу вентильних матриць складає базисна комірка, що складається з набору нез'єднаних транзисторів та резисторів. Кожен постачальник замовних мікросхем самостійно визначає оптимальний набір компонентів, що входять до складу окремої базисної комірки (рис. 4.23).

Постачальники замовних мікросхем починали свою роботу з виготовлення заводським способом кремнієвого кристала, що містить масив базисних комірок. У разі канальної матриці базисні комірки, як правило, розташовувалися у вигляді одностовпцевих або двостовпцевих масивів, причому між стовпцями були вільні області, так звані канали (рис. 4.24).

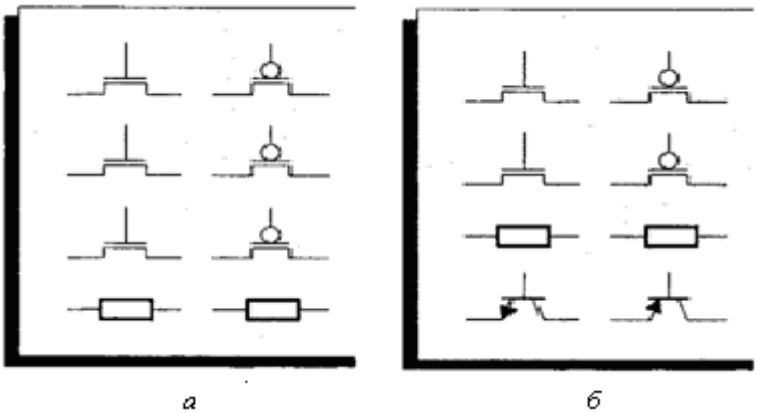


Рис. 4.23. Приклади простих базисних комірок вентильної матриці:
а – однорідна КМОП; б – змішана біполярна КМОП

За відсутності каналів на матриці комірки розташовувалися у вигляді одного великого масиву. Поверхня такого пристрою була покрита великою кількістю комірок і не мала виділених каналів для внутрішніх з'єднань. Такі матриці називалися безканальними**.**

Розробники можуть використовувати визначений постачальником мікросхем набір логічних функцій, таких, як прості логічні вентилі, мультиплексори і регістри. Кожна з цих функцій-блоків розглядається як окремий елемент або комірка (не плутати з базисною коміркою). Набір таких функцій, що підтримується постачальником, називається бібліотекою елементів.

Рис. 4.24. Архітектура канальних логічних матриць:
а – одностовпцеві масиви: б – двостовпцеві масиви

Розробники одержують таблицю з'єднань на рівні вентилів, яка описує, які вентилі використовуватимуться, і які між ними будуть з'єднання. Щоб встановити відповідність логічних вентилів і базисних комірок і визначити, як ці комірки зв'язуватимуться між собою, використовуються спеціальні програми визначення відповідностей, програми розставляння, трасування та інші.

На основі отриманих результатів виготовляються фотошаблони, за допомогою яких створюються шари металізації. Ці шари зв'язуватимуть компоненти усередині базисної комірки, а також комірки між собою, з входами та виходами пристрою.

Вентильні матриці мають розумну ціну, оскільки транзистори і інші компоненти виробляються заводським способом, а за замовленням виготовляються тільки шари металізації. Більшість таких пристроїв не використовує всіх внутрішніх ресурсів. Крім того, розташування вентилів жорстко визначене, а трасування внутрішніх з'єднань далеке від оптимального. Усі ці недоліки негативно позначилися на продуктивності і споживаній потужності цих пристроїв.

Вирішення проблем, властивих вентильним матрицям стало можливим з появою на початку 90-х схем на стандартних елементах. Ці компоненти мали багато спільного з вентильними матрицями.

Постачальник спеціалізованої замовної (ASIC) мікросхеми складає бібліотеку елементів, і цією бібліотекою можуть користуватися розробники. Постачальник також підтримує бібліотеки апаратних і програмних макровизначень, які включають такі елементи, як процесори, функції зв'язку, ОЗП або ПЗП. Розробники можуть також використати раніше створені функції або придбати блоки інтелектуальної власності.

Коли команда інженерів конструює складну мікросхему, вона може ухвалити рішення про покупку вже готового проекту одного або декількох функціональних блоків. Проект таких функціональних блоків називається інтелектуальною власністю або IP (intellectual property). Засоби інтелектуальної власності можуть охоплювати будь-які функціональні блоки, зокрема функції зв'язку і мікропроцесори. Найскладніші функції, такі, як мікропроцесори, можуть називатися ядрами.

Тим або іншим способом в наші дні за допомогою складних систем автоматизованого проектування розробники завершують свою роботу створенням таблиці з'єднань на рівні вентилів. Така таблиця описує, які вентилі використовуються і як вони з'єднані між собою.

На відміну від вентильних матриць, схеми на стандартних елементах не використовують концепцію базисних комірок, і компоненти на кремнієвому кристалі не виготовляються заводським способом. Для індивідуального розташування кожного вентиля в таблиці з'єднань і для оптимального розводення з'єднань між ними використовуються спеціальні засоби автоматизованого проектування. Отримані результати використовуються для створення замовних фотошаблонів для кожного шару мікросхеми.

Концепція схем на стандартних елементах дозволяє створювати логічні функції, використовуючи мінімальну кількість транзисторів без якої-небудь надмірності компонентів; самі функції можуть бути реалізовані так, щоб зробити менш трудомістким створення внутрішніх зв'язків між ними. Саме тому схеми на стандартних елементах забезпечують майже оптимальне використання поверхні кремнієвого кристала в порівнянні з вентильними матрицями.

Структуровані спеціалізовані мікросхеми, до речі спочатку вони називалися інакше, з'явилися на світ приблизно на початку 90-х. Через 10 років, приблизно в 2001-2002 році, деякі виробники приступили до вивчення способів зниження витрат на проектування спеціалізованих мікросхем і скорочення тривалості їх розробки. Приблизно в 2003 році з'явилася назва «структуровані спеціалізовані мікросхеми» (structured ASIC) [61].

У кожного постачальника власна архітектура пристрою. У зв'язку з цим будуть розглянуті тільки загальні риси цих пристроїв. Кожен пристрій починається з фундаментального елемента, який одні називають модулем, а інші – елементом мозаїки. Такий елемент може містити виготовлений заводським способом набір загальної логіки, виконаної у вигляді логічних вентилів, мультиплексорів або таблиць відповідності, одного або декількох регістрів, і, можливо, невеликого локального ОЗП (рис. 4.25).

Масив таких елементів виготовляється заводським способом по всій поверхні кристала. Деяка альтернативна архітектура починається або з

базисної комірки, або з базисного модуля, або з базисного елемента мозаїки, або з чогось іншого. До її складу входять тільки елементи загальної логіки у формі виготовлених заводським способом вентилів, мультиплексорів або таблиць відповідності. Масив таких базисних одиниць (говорять 4x4, 8x8 або 16x16) в поєднанні з деякими спеціальними модулями, що містять регістри, невеликі елементи пам'яті і іншу логіку, утворює базову комірку або базовий модуль, або базовий елемент мозаїки, або ін. Цей масив виготовляється заводським способом по всій поверхні кристала.

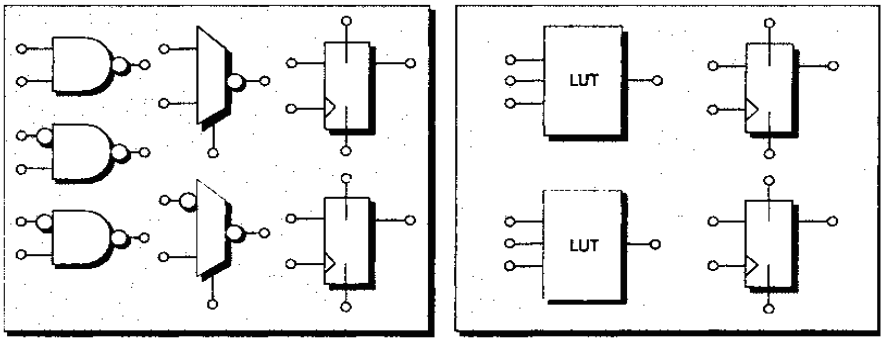


Рис. 4.25. Модулі спеціалізованих структурованих ІС:
а – модуль, що містить вентилі, мультиплексори та тригери;
б- модуль, що містить таблиці відповідності (LUT) та тригери

Заводським способом також виготовляються, зазвичай по контуру кристала, деякі додаткові функції, наприклад блоки ОЗП, тактові генератори, логіка периферійного сканування та інші функції (рис. 4.26).

При виконанні пристрою за замовленням виготовляються тільки шари металізації, як і у разі стандартних вентильних матриць. Відмінність криється у тому, що внаслідок більшої складності модулів структурованих спеціалізованих ІС більшість шарів металізації вже визначена. Таким чином, при виготовленні багатьох структурованих мікросхем за замовленням їх архітектура потребує всього лише двох або трьох шарів металізації, а в деяких випадках вимагається виготовити тільки один шар з перехідними отворами. У результаті значно знижується вартість і скорочується час виготовлення фотошаблонів, що залишилися, необхідних для створення пристрою.

Визначити точне значення параметрів досить складно, проте визначена і виготовлена заводським способом логіка структурованих мікросхем, в порівнянні з схемами на стандартних елементах, споживає велику потужність, має більшу продуктивність і займає більше місця на кристалі.

Для виконання тих самих функцій структуровані мікросхеми в середньому потребують в три рази більше місця на кристалі і споживають в два-три рази більше енергії, ніж схеми на стандартних елементах. На практиці ці параметри істотно залежать від типу пристрою і його архітектури.

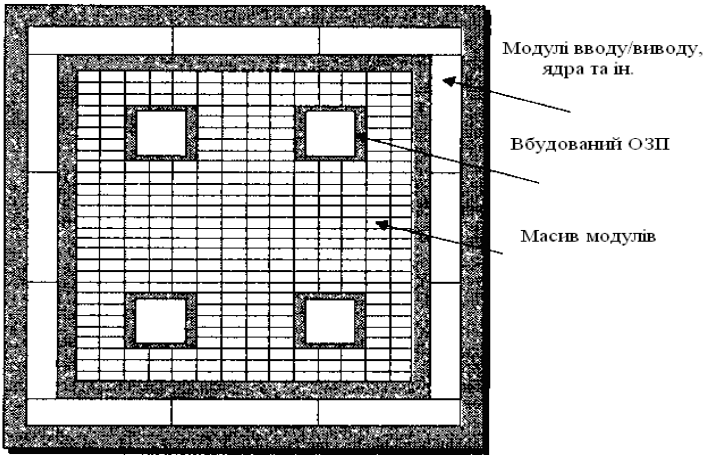


Рис. 4.26. Типова спеціалізована структурована ІС

4.5. Архітектура й технології проектування ПЛІС

На початку 80-х існували програмовані пристрої, подібні до простих і складних ПЛП, які мали відносно невелику вартість при малому часі виготовлення та модифікації. Але ці пристрої не були здатні підтримувати великі і складні функції.

З іншого боку, існували замовні спеціалізовані інтегральні мікросхеми. Вони підтримували надзвичайно великі і складні функції, але були надзвичайно дорогими, і для їх виготовлення був потрібен значний час. Крім того, остаточний варіант замовної мікросхеми «заморожувався в кремнії».

Для ліквідації недоліків означених пристроїв фірма Xilinx розробила новий клас мікросхем FPGA (Field programmable gate arrays), або ПЛІС (програмовані логічні інтегральні схеми) [62], які з'явилися у продажу в 1984 році. Перші ПЛІС виготовлялися за КМОП-технологією і для зберігання конфігурації використовували статичний ОЗП. Не дивлячись на те що перші версії цих пристроїв були порівняно простими і містили, за сьогоднішніми мірками, відносно мало вентилів або їх еквівалентів, багато уявлень, що лежать в основі їх архітектури, використовуються і в наші дні.

Основу перших ПЛІС складала концепція програмованих логічних блоків, які включали 3-входову таблицю відповідності (LUT — lookup table), регістр, що виконує функцію тригера або клямки, мультиплексор, а також деякі інші елементи. На рис. 4.27 як приклад показаний дуже простий програмований логічний блок (логічний блок сучасної ПЛІС може бути набагато складнішим).

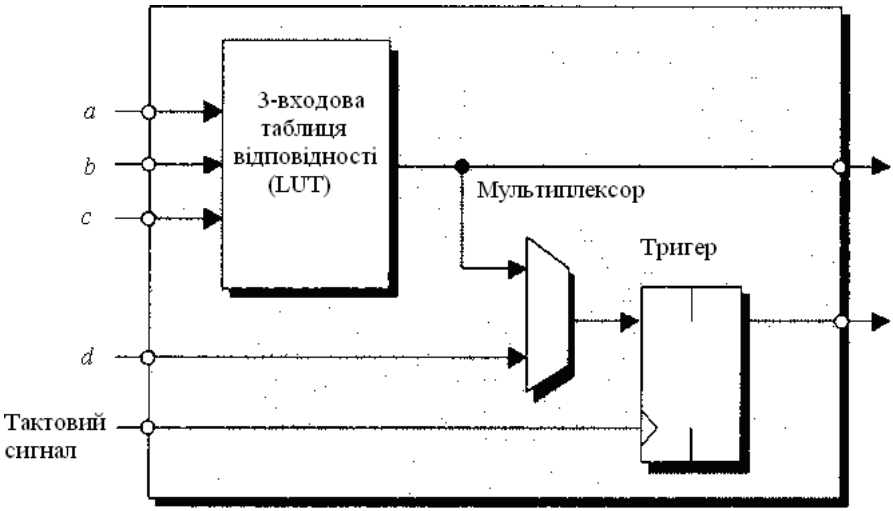


Рис. 4.27. Простий програмований логічний блок

Кожна ПЛІС містить велику кількість таких програмованих блоків. Шляхом програмування відповідних комірок статичного ОЗП кожен логічний блок пристрою може бути конфігурований для виконання різних функцій. У процесі конфігурації в кожен регістр записується його початкове значення у вигляді логічного 0 або логічної 1, а також визначається, чи регістр виконуватиме функцію тригера або клямки (рис. 4.27). У першому випадку також визначається, по якому фронту тактового сигналу, позитивному або негативному, вироблятиметься перемикання. Тактові сигнали є загальними для всіх логічних блоків. Мультиплексор, підключений до входу тригера, може конфігуруватися на передачу сигналів з виходу таблиці відповідності або з окремого входу логічного блока. Таблиця відповідності може програмуватися на роботу як будь-яка логічна функція з трьома входами і одним виходом.

Припустимо, що таблиці відповідності необхідно сформувати функцію

$$y = (a \,\&\, b) \mid \text{not } c.$$

Для цього в таблицю відповідності необхідно завантажити відповідні вихідні значення цієї функції (рис. 4.28).

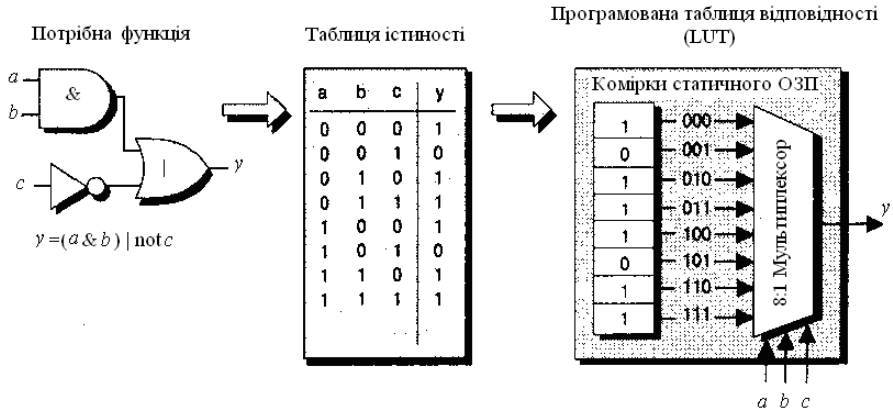


Рис. 4.28. Конфігурація таблиці відповідності

Таблиця відповідності з мультиплексором 8:1, наведена на рис. 4.28, є спрощеним варіантом існуючих таблиць відповідності. ПЛІС складається з великого числа програмованих логічних блоків з програмованими внутрішніми з'єднаннями (рис. 4.29).

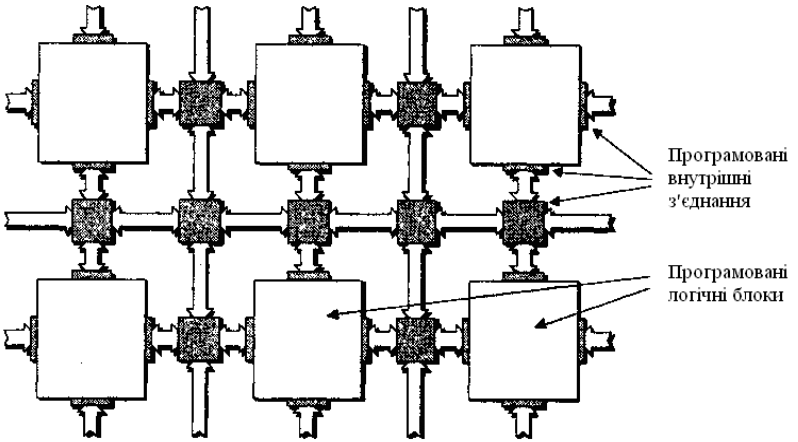


Рис. 4.29. Спрощена архітектура ПЛІС

Спрощена схема являє собою лише наближене уявлення про архітектуру ПЛІС. Насправді всі транзистори і внутрішні з'єднання

виконуються на одному кристалі кремнію за допомогою стандартних технологій виготовлення інтегральних мікросхем.

Окрім локальних внутрішніх зв'язків, показаних на рис. 4.29, існують глобальні, або високошвидкісні, з'єднання, які передають сигнали через кристал без участі численних локальних перемикаючих елементів.

Пристрій також містить контактні площадки і ніжки для вводу/виводу даних (на рис. 4.29 не показані). За допомогою елементів пам'яті статичного ОЗП внутрішні з'єднання програмуються так, щоб входи мікросхеми з'єднувалися з входами одного або декількох програмованих логічних блоків, а виходи цих блоків підключалися до входів інших блоків та/або подавалися на виходи мікросхеми.

ПЛІС успішно справилися з роллю моста між ПЛП і замовними спеціалізованими мікросхемами. З одного боку, вони мають високий ступінь можливості конфігурації з малим часом виготовлення і модифікації, як і ПЛП. З іншого боку, вони можуть використовуватися для реалізації великих і складних функцій, які раніше реалізовувалися тільки за допомогою замовних мікросхем. Спеціалізовані замовні мікросхеми призначалися для дійсно великих, складних і високотехнологічних систем. Проте на місце, відведене цим мікросхемам у сфері програмованої логіки, претендують ПЛІС, які у міру вдосконалення почали поступово їх витісняти.

Принцип розробки за зразком, або розробки платформи, тривалий час використовувався на рівні друкарської плати. Згідно з цим принципом на основі базової конфігурації може бути реалізована безліч похідних продуктів.

Сучасні високотехнологічні ПЛІС, крім величезної кількості програмованої логіки, містять вбудовані блоки ОЗП, вбудовані процесорні ядра, високошвидкісні блоки вводу/виводу і багато що інше. Крім того, розробники мають доступ до великого набору блоків інтелектуальної власності. Все це сприяло розвитку концепції ПЛІС-платформи. Суть її полягає у тому, що будь-яка компанія може використовувати вже спроектовану ПЛІС-платформу для багатьох пристроїв внутрішнього користування або запропонувати початкові проекти іншим компаніям для виготовлення замовних пристроїв.

Очевидно, що немає ніякого сенсу вбудовувати компоненти замовних інтегральних схем всередину ПЛІС, оскільки отримана інтегральна схема буде джерелом класичних проблем, які властиві технології створення замовних мікросхем: високі витрати на проектування і виробництво, тривалі терміни виходу на ринок і інші. Проте існує ряд випадків, коли один або декілька компонентів ПЛІС використовуються як частина замовних мікросхем на стандартних елементах.

Одна з причин вбудовування компонентів ПЛІС всередину замовних мікросхем — зробити якомога простішим принцип розробки платформи. В

цьому випадку платформа належатиме до класу замовних мікросхем, але вбудовані компоненти ПЛІС забезпечуватимуть один з механізмів, що використовується для адаптації і модифікації виробу.

Інша причинна пов'язана з тим, що останні декілька років спостерігається розширення сфери застосування ПЛІС, зокрема вони можуть використовуватися для надання додаткових властивостей виробам на замовних мікросхемах. В цьому випадку великі і складні спеціалізовані мікросхеми з'єднуються з ПЛІС, розташованими на одній друкарській платі в безпосередній близькості один від одного (рис. 4.30).

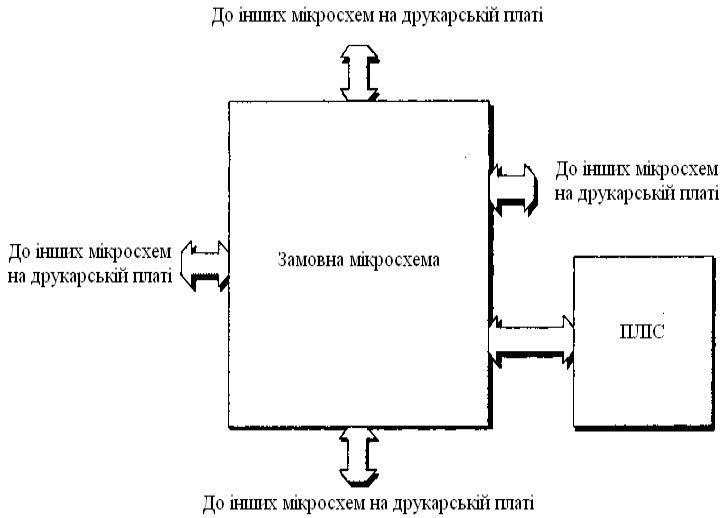


Рис. 4.30. Використання ПЛІС в комплексі із замовними мікросхемами

Застосування цієї схеми виправдане великою трудомісткістю і дорожнечею операцій з усунення помилок і зі зміни функціональності системи при використанні замовних мікросхем. Навіть якщо виріб на замовних мікросхемах не містить помилок, застосування ПЛІС можна використовувати для виконання низькорівневої модифікації і модернізації системи. Недоліком такої технології є тривалість проходження сигналів від замовної мікросхеми до ПЛІС і назад. Усунути цей недолік можна, вбудувавши ядро ПЛІС всередину замовної мікросхеми. У результаті одержимо такий гібрид: ПЛІС — замовна інтегральна схема (FPGA-ASIC).

При цьому треба мати на увазі, що системи автоматизованого проектування і методи проектування замовних мікросхем і ПЛІС істотно розрізняються. Наприклад, про замовні мікросхеми можна сказати, що вони дрібномодульні**,** оскільки в основному вони створюються на рівні простих

логічних елементів. Це означає, що традиційні методи проектування, такі, як логічний синтез, традиційні методи розміщення елементів і трасування з'єднань також застосовні до проектування дрібномодульних замовних інтегральних схем.

Що стосується ПЛІС, то про них можна сказати, що вони є середньо модульними (або крупномодульними, оскільки фізично вони реалізуються з використанням високорівневих блоків, таких як, програмовані логічні блоки). В цьому випадку для проектування краще використовувати специфічні для ПЛІС методи синтезу, розміщення елементів і трасування з'єднань.

Областю застосування гібридів вигляду ПЛІС — замовна мікросхема є структуровані спеціалізовані мікросхеми, або структуровані ASIC, оскільки вони також відповідають принципам блокової побудови. Коли постачальники структурованих спеціалізованих мікросхем обирають засоби проектування, вони частіше спілкуються з постачальниками ПЛІС-ОРІЄНТОВАНИХ засобів синтезу, технологій розміщення елементів і трасування з'єднань, ніж з їх колегами, що пропонують традиційні засоби. Таким чином, для проектування гібридів вигляду ПЛІС-ЗАМОВНА мікросхема, заснованих на структурованій ASIC, автоматично можна користуватися єдиними засобами розробки, оскільки ті самі методи блокового синтезу, розміщення і трасування можуть використовуватися для виготовлення як «замовної», так і «ПЛІС»-частини мікросхеми.

Деякі частини пристроїв подібні до програмованих логічних блоків і до базових сполучних структур, і в цьому випадку постачальники борються за кожен квадратний мікрон і кожну частинку наносекунди. Такими частинами пристроїв є транзистори і провідники, що виготовляються вручну за технологією замовних мікросхем. Ці частини пристрою є відносно малими і часто повторюються, тобто будучи один раз створені, вони можуть тисячі разів відтворюються на поверхні кристала.

Існують і допоміжні частини пристроїв, такі як схеми управління, що конфігуруються, які зустрічаються в одиничному екземплярі і до яких не ставляться жорсткі вимоги за розмірами і продуктивністю. Ці частини виготовляються методами, які використовуються для створення схем на стандартних елементах.

4.5.1. Архітектура базисного модуля ПЛІС. Реалізація на комірках ОЗП та на мультиплексорах

При побудові ПЛІС використовуються раніш розглянуті технології: метод нарощуваних перемичок, використання комірок статичного ОЗП, ЕСППЗП та ін. До складу ПЛІС також можуть входити вбудовані блоки: мультиплексори, суматори, блоки ОЗП, мікропроцесорні ядра та ін.

Більшість ПЛІС використовує для зберігання конфігурації комірки пам'яті статичного ОЗП, які можуть бути багато разів перепрограмовані. Головною перевагою цієї технології є те, що вона дозволяє легко і швидко реалізувати і протестувати всі нові ідеї, відносно легко підстроюючись під нові стандарти і протоколи. Крім того, при включенні системи така ПЛІС може бути спочатку запрограмована для виконання певних функцій, наприклад для самотестування або тестування всієї системи, а потім може бути перепрограмована для виконання своєї головної задачі.

Інша істотна перевага використання комірок статичного ОЗП полягає у тому, що ця технологія є передовою. На постачальників ПЛІС також діє той факт, що багато компаній, які спеціалізуються на пристроях пам'яті, витрачають величезні ресурси на дослідження і розвиток комірок статичного ОЗП. Більш того, ці елементи пам'яті створюються за такою ж КМОП-технологіїю, як і решта частин ПЛІС, тобто для створення цих компонентів не потрібні якісь спеціальні технології.

Раніше для апробації виробництва мікросхем за новими технологіями часто використовувалися мікросхеми пам'яті. Останнім часом сукупність таких характеристик, як розмір, складність і безперебійність останніх поколінь ПЛІС зробили можливим їх застосування для розв'язання і цих задач. На відміну від пристроїв пам'яті, ПЛІС дозволяють легко ідентифікувати і знаходити дефекти структури, тобто встановлювати факт помилки і навіть визначати, що і де відбулося. Наприклад, коли компанії IBM і UMC впроваджували технологію 0.09 мкм (90 нм), ПЛІС фірми Xilinx стали першими пристроями, виготовленими за такою технологією [62].

Недоліком пристроїв на основі статичного ОЗП є те, що потрібно змінювати їх конфігурацію кожного разу при включенні системи. Для цього доводиться використовувати спеціальну зовнішню пам'ять, що збільшує вартість системи і вимагає місця на друкарській платі, або використовувати вбудований мікропроцесор, або реалізувати якісь інші варіанти.

На відміну від пристроїв, заснованих на комірках статичного ОЗП, які програмуються при включенні системи, пристрої на основі нарощуваних перемичок програмуються у вимкненому стані за допомогою так званого програматора.

ПЛІС на основі нарощуваних перемичок мають ряд переваг. По-перше, ці пристрої є енергонезалежними, тобто їх конфігураційні дані не стираються при відключенні живлення системи. Це означає, що вони готові до роботи відразу після включення системи. Будучи незалежними, ці пристрої не вимагають додаткової мікросхеми зовнішньої пам'яті для зберігання конфігураційних даних. Це дозволяє зменшити вартість всієї системи і зберегти вільне місце на друкарській платі.

Одна з переваг ПЛІС на основі нарощуваних перемичок заслуговує особливої уваги: структура їх внутрішніх з'єднань є дійсно «наджорсткою», а також такі пристрої відносно стійкі до радіації. Ця властивість перемичок може становити певний інтерес для військових і космічних додатків. Стан конфігураційних комірок компонентів

статичного ОЗП може бути «перемкнутий» в довільний стан, якщо ці комірки будуть уражені радіацією, якої так багато в космосі. Порівняно з ними, одного разу запрограмована нарощувана перемичка не може бути змінена у такий спосіб. При цьому слід мати на увазі, що будь-які тригери в цьому пристрої залишаються чутливими до дії радіації, тому мікросхеми, призначені для роботи в умовах радіаційної дії, повинні мати захист тригерів у вигляді системи потрійного резервування. Іншими словами, у системі повинні бути три копії кожного регістра, і рішення про значення вихідного сигналу ухвалюється більшістю голосів. У ідеальному випадку всі три регістри міститимуть однакове значення, але якщо один з регістрів «перемикається» так, що два з них містять 0, а третій 1, то вихідним значенням буде 0, і, навпаки, якщо два регістри містять 1, а третій 0, то вихідним значенням вважатиметься 1.

Але, можливо, найважливішою перевагою ПЛІС на основі нарощуваних перемичок є те, що їх конфігураційні дані приховані глибоко всередині. За визначенням програматор може прочитати ці дані, оскільки це частина його роботи. Після нарощування кожної перемички програматор повинен провести тестування, щоб переконатися, що елемент успішно запрограмований, і лише потім перейти до наступної перемички. Крім того, програматор може використовувати автоматичну перевірку успішної конфігурації пристрою. Щоб це зробити, програматору потрібно буде здійснити читання стану нарощуваних перемичок і отримані результати порівняти з необхідним станом, який визначається в конфігураційному файлі.

Після програмування пристрою слід скористатися можливістю нарощувати спеціальний захист перемичок, і, таким чином, забезпечити захист від зчитування будь-яких запрограмованих даних з пристрою (у формі наявності або відсутності перемичок). Навіть якщо пристрій буде розкритий, запрограмовані і незапрограмовані перемички залишаються ідентичними, до того ж той факт, що нарощувані перемички втоплені у внутрішніх шарах металізації, робить конструкцію недоступною для зворотного проектування.

Пристрої на нарощуваних перемичках є одноразово програмованими, і це їх головний недолік, оскільки, якщо ПЛІС вже була одного разу запрограмована, змінити її конфігурацію вже неможливо.

Конфігураційні комірки ПЛІС на основі ЕСППЗП або Flash-пам'яті так само, як і елементи пам'яті статичного ОЗУ, утворюють довгий ланцюжок, подібно до регістру із зсувом. Ці пристрої можуть програмуватися у відключеному стані за допомогою програматора. Деякі версії пристроїв можна програмувати, не вимикаючи живлення, але при цьому час їх програмування приблизно в три рази перевищує час програмування пристроїв на статичному ОЗП.

Після програмування дані, що містяться в пристроях, будуть енергонезалежними, тобто ці пристрої будуть відразу готові до роботи після подачі напруги на систему. Що стосується захисту, то деякі з цих пристроїв використовують принцип мультибітного ключа, розмір якого приблизно становить від 50 до декількох сотень бітів. Після програмування пристрою можна завантажити в нього особистий ключ, тобто бітову

комбінацію, для захисту конфігураційних даних. Річ у тому, що, якщо завантажити ключ, то прочитати дані з пристрою або записати в нього нові дані можна буде тільки, завантаживши копію ключа через JTAG-порт. Якщо при цьому врахувати, що JTAG-порт сучасних пристроїв працює на частоті 20 Мгц, то злом ключа методом перебору кожного можливого значення забере мільйони років.

Двотранзисторні ЕСППЗП- і Flash-елементи пам'яті приблизно в два з половиною рази більші за розміром ніж однотранзисторні, але вони істотно менші, ніж комірки статичного ОЗУ. Це означає, що пристрій може бути виконаний компактнішим, з меншими затримками на внутрішніх з'єднаннях.

Разом з тим ці пристрої вимагають приблизно п'ять додаткових технологічних кроків після завершення стандартного технологічного циклу КМОП. Саме в цьому криється причина їх відставання від пристроїв на статичному ОЗП. Ще один суттєвий недолік полягає в тому, що ці пристрої мають тенденцію зберігати відносно високу потужність споживання в статичному режимі через велику кількість резисторів навантажень, що міститься в них.

Деякі виробники ПЛІС часто пропонують різні комбінації технологій програмування. Розглянемо, наприклад, пристрій, кожен конфігураційний елемент якого є комбінацією Flash- (або ЕСППЗП-) елемента пам'яті і пов'язаного з нею елемента пам'яті статичного ОЗП.

У цьому випадку Flash-елементи можуть бути заздалегідь запрограмовані. Потім, після включення системи, вміст Flash-елементів пам'яті масово паралельно копіюється у відповідні до них елементи пам'яті статичного ОЗП. Така технологія забезпечує незалежність, властиву пристроям на основі нарощуваних перемичок, а це означає, що пристрій буде негайно готовий до роботи після включення системи. Але, на відміну від пристроїв на основі нарощуваних перемичок, елементи пам'яті статичного ОЗП можна використовувати для перепрограмування пристрою у включеному стані. Аналогічно можна перепрограмувати Flash-елементи пам'яті пристрою, не знімаючи напруги живлення, або виконати процедуру перепрограмування за допомогою програматора після виключення системи.

У загальному випадку ПЛІС підрозділяються на дрібномодульні та крупномодульні. Щоб зрозуміти цю класифікацію, треба пам’ятати, що головною особливістю ПЛІС є їх внутрішня структура, яка переважно складається з великої кількості простих програмованих логічних блоків та програмованих внутрішніх зв'язків (рис. 4.29). У дрібномодульній архітектурі кожен логічний блок може використовуватися для реалізації тільки дуже простої функції.

Дрібномодульні структури використовуються при реалізації зв'язуючої логіки та неоднорідних структур, подібних до кінцевих автоматів. Дрібномодульні структури також ефективні при реалізації алгоритмів систол (функції, які надзвичайно ефективні за рахунок реалізації масового паралелізму). Ці структури мають певну перевагу при використанні технології традиційного логічного синтезу, яка базується на дрібнодульній архітектурі замовних мікросхем.

Пік інтересу до дрібномодульної архітектури ПЛІС був відмічений у середині 90-х років. Проте з часом переважна більшість представників цього сімейства припинила своє існування, а залишилися лише представники крупномодульної архітектури. У подібній архітектурі кожен логічний блок містить відносно велику кількість логіки. Так, наприклад, логічний блок може містити чотири 4-входових таблиці відповідності, чотири мультиплексори, чотири D-тригери і деяку кількість логіки швидкого переносу.

Для дрібномодульних ПЛІС характерна велика кількість з'єднань усередині блоків і між ними. У міру збільшення модульності пристроїв до середньомодульних і вище кількість з'єднань в блоках зменшується. Це важлива властивість, оскільки внутрішні зв'язки визначають величину переважної більшості затримок, пов'язаних з проходженням сигналів через ПЛІС.

У класифікації ПЛІС є деяка неоднозначність. Деякі компанії створюють крупномодульні пристрої. Ці пристрої містять масиви вузлів, де кожен вузол є складним елементом, що реалізовує алгоритмічні функції, наприклад швидке перетворення Фур'є (ШПФ), або навіть ядро мікропроцесора загального призначення. Суть полягає у тому, що насправді ці пристрої не класифікуються як ПЛІС. З цієї причини архітектуру ПЛІС на основі таблиць відповідності (LUT) часто класифікують як середньомодульну, тим самим, звільняючи термін «крупно модульний» для позначення таких пристроїв, як пристрої на основі вузлів.

Існують два основні способи реалізації програмованих логічних блоків, що використовуються для формування середньомодульних пристроїв на основі мультиплексорів (MUX — від multiplexer) і на основі таблиць відповідності (LUT — від lookup table).

Як прикладу реалізації пристроїв на основі мультиплексорів розглянемо 3-входовую функцію $y = (a \,\&\, b) \mid c$, реалізовану за допомогою блока, що містить тільки мультиплексори (рис. 4.31).

Пристрій може бути запрограмований таким чином, що на кожен його вхід може подаватися логічний 0 або логічна 1, або істинне, або інверсне значення вхідного сигналу (у прикладі *a*, *b* або *c*), що приходить з іншого блока або з входу мікросхеми. Такий підхід дозволяє для кожного блока створювати величезну кількість варіантів конфігурації для виконання різноманітних функцій (*x* на вході центрального мультиплексора на рис. 4.31 означає, що на вхід можна подавати будь-який сигнал — 0 або 1).

Основна концепція таблиць відповідності відносно проста. У таких мікросхемах група вхідних сигналів використовується як індекс (покажчик, або адреса комірки) таблиці відповідності. Вміст цієї таблиці організований таким чином, що комірки, на які вказує кожна вхідна комбінацією, містять необхідне вихідне значення. Припустимо, що вимагається реалізувати функцію $y = (a \,\&\, b) \mid c$ (рис. 4.32).

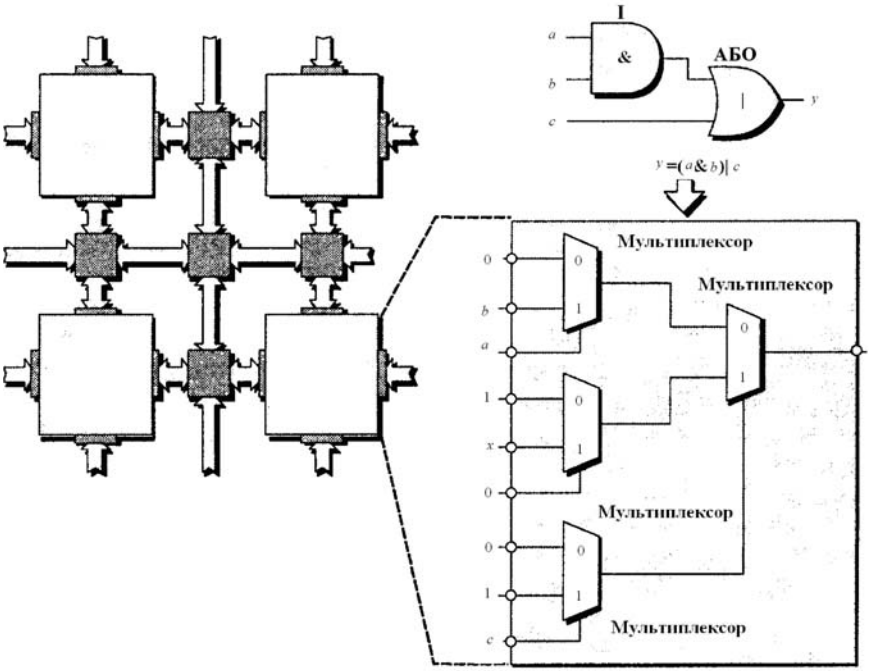


Рис. 4.31. Логічний блок на мультиплексорах

Якщо узяти групу логічних вентилів глибиною в декілька шарів, то таблиця відповідності може бути досить ефективною з погляду використання ресурсів і затримки розповсюдження сигналу. Тут під «глибиною» мається на увазі кількість логічних елементів між входом і виходом ланцюжка (на рис. 4. 33 глибина складає два шари). Проте недоліком архітектури на таблицях відповідності є те, що якщо з їх допомогою реалізувати невелику функцію, наприклад 2-входовий логічний елемент І, то для цього доведеться використовувати всю таблицю. Результуюча затримка для такої простої функції виявиться достатньо великою.

Таблиця істинності

a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Рис. 4.32. Логічна функція та відповідна до неї таблиця істинності

Для цього треба завантажити 3-входову таблицю відповідними значеннями. Таблиця відповідностей може формуватися з елементів пам'яті статичного ОЗП. Вона може також бути сформована нарощуваними перемичками, ЕСППЗП- або Flash-елементами пам'яті. Для вибору необхідної комірки ОЗП за допомогою каскаду передавальних вентилів використовуються вхідні сигнали, як показано на рис. 4.33. При цьому елементи пам'яті статичного ОЗП для завантаження конфігураційних даних повинні бути поєднані в довгий ланцюжок, але ці ланцюжки на рис. 4.33 не показані з метою його спрощення.

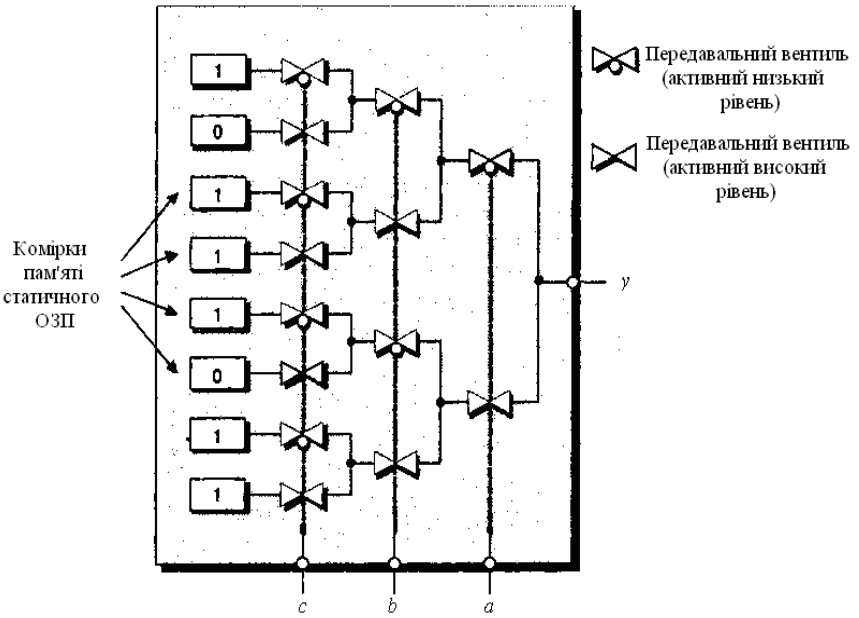


Рис. 4.33. Таблиця істинності на основі передавальних вентилів

На цій схемі відкритий, або активний, передавальний вентиль пропускає сигнал з входу на вихід. Закритий вентиль відключає свій вихід від провідника, до якого він приєднаний.

Передавальні вентилі, на позначеннях яких зображене невелике «коло», активуються при подачі на управляючий вхід логічного 0. І навпаки, вентилі, на позначеннях яких немає «кола», активуються при подачі на управляючий вхід рівня логічної 1. Виходячи з цього, легко прослідити, як різні вхідні комбінації можуть використовуватися для вибору вмісту необхідного елемента пам'яті.

До появи сучасних систем автоматизованого проектування, коли інженери уручну виготовляли схеми, деякі стверджували, що застосування архітектури на основі мультиплексорів дозволить досягти кращих результатів. Говорять також, що мультиплексорна архітектура має перевагу при розробці управляючої логіки. Проте деякі реалізації цієї архітектури не забезпечують роботу високошвидкісних ланцюжків логічного перенесення. Їх аналоги на таблицях відповідності залишаються лідерами у всіх додатках, в яких передбачені арифметичні дії.

Порівняно з таблицями відповідності, при використовуванні мультиплексорної архітектури, що містить суміш мультиплексорів і логічних вентилів, часто вдається здійснити доступ до проміжних значень сигналів, що проходять між логічними елементами і мультиплексорами. В цьому випадку при реалізації невеликих функцій непотрібні (зайві) логічні блоки можуть бути відключені. Отже, мультиплексорна архітектура може мати переваги з погляду продуктивності і використання ресурсів кристала при реалізації пристроїв, що містять велику кількість простих незалежних логічних функцій.

У 90-х роках ПЛІС набули великого поширення у області мереж передачі даних і зв'язку. Обидві сфери мають на увазі передачу великих потоків даних, в яких архітектура на основі таблиць відповідності зарекомендувала себе дуже добре. У міру того як пристрої і їх пропускна спроможність ставали більшими, то і технологія синтезу мікросхем ставала складнішою, а ручне виготовлення схем разом з мультиплексорною архітектурою поступово зменшувалося. У результаті більшість сучасної архітектури ПЛІС базується на таблицях відповідності.

Важливою особливістю багатовходової таблиці відповідності є те, що вона може реалізувати будь-яку w-входову комбінаційну, або комбінаторну логічну функцію. За допомогою більшої кількості входів можна реалізувати складніші функції, але при цьому слід враховувати, що додавання кожного нового входу супроводжується подвоєнням кількості комірок статичного ОЗП.

Перші ПЛІС базувалися на 3-входових таблицях відповідності. Згодом було проведено аналіз порівняльних експлуатаційних характеристик 3-, 4-, 5- і навіть 6-входових таблиць відповідності. Результати досліджень показали, що оптимальній рівновазі всіх «за і проти» відповідають 4-входові таблиці відповідності.

У минулому деякі постачальники використовували в своїх пристроях таблиці відповідності різних розмірів, наприклад, на три і на чотири входи, оскільки вони обіцяли найбільш оптимальне використання пристрою. Проте це рішення не задовольняло більшість інженерів, які вважали за краще використовувати програми синтезу схем. Тому в даний час вся дійсно вдала архітектура базується тільки на 4-входових таблицях відповідності. Це зовсім не означає, що архітектура на змішаних за розміром таблицях відповідності не з'явиться знов, оскільки програмне забезпечення систем проектування продовжує ускладнюватися

Ядро таблиці відповідності в пристрої на статичному ОЗП використовує для своєї роботи декілька елементів пам'яті. Це дозволяє використовувати деякі цікаві можливості. Крім основного призначення, тобто формування таблиці відповідності, пристрої деяких постачальників дозволяють використовувати комірки, що формують таблицю, як невеликі блоки оперативної пам'яті. Наприклад, 16 елементів пам'яті, що формують 4-входову таблицю, можуть виступати в ролі блока ОЗП 16x1. Такі ділянки пам'яті називаються розподіленим ОЗП, оскільки, по-перше, таблиці відповідності розкидані (розподілені) по всій поверхні кристала, а по-друге, ця назва відрізняє їх від великих блоків ОЗП.

Інший варіант альтернативного використання таблиць заснований на тому, що всі конфігураційні комірки, включаючи і ті, які формують таблицю відповідності, ефективно зв'язані разом в один довгий ланцюжок (рис. 4.34).

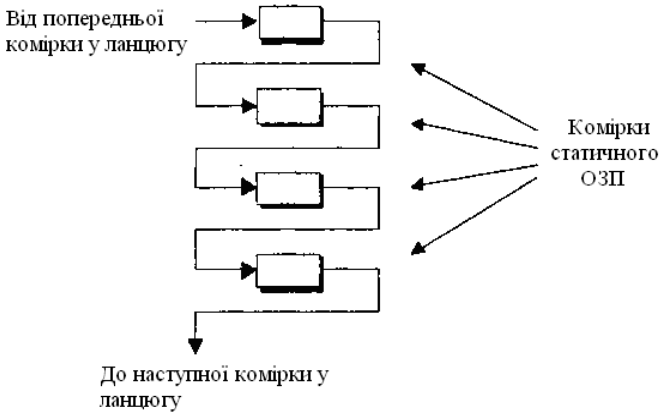


Рис. 4.34. Конфігураційні комірки, зв'язані в ланцюжок

Річ у тому, що в деяких пристроях комірки пам'яті, що формують таблицю відповідності, після програмування можуть розглядатися незалежно від головної структури ланцюжка і використовуватися як регістр зсуву (рис. 4.34). Таким чином, кожну таблицю відповідності можна розглядати як багатофункціональний компонент.

4.5.2. Архітектура базисного модуля, що конфігурується (логічна комірка фірми Xilinx)

Блоки, з яких складається сучасна ПЛІС фірми Xilinx, називаються логічними комірками (logic cell). Крім всього іншого, логічна комірка містить

4-входову таблицю відповідності, яка може працювати як ОЗП 16x1 або як 16-бітовий регістр зсуву, а також мультиплексор і регістр (рис. 4.35) [63].

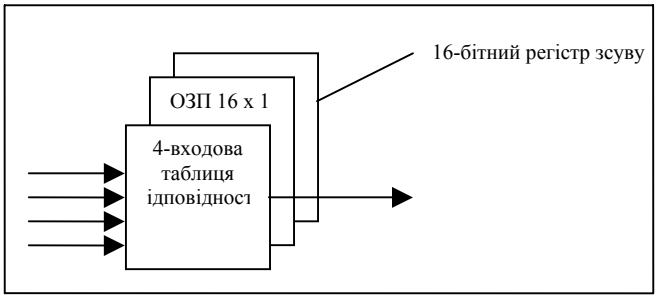


Рис. 4.35. Багатофункціональна таблиця відповідності

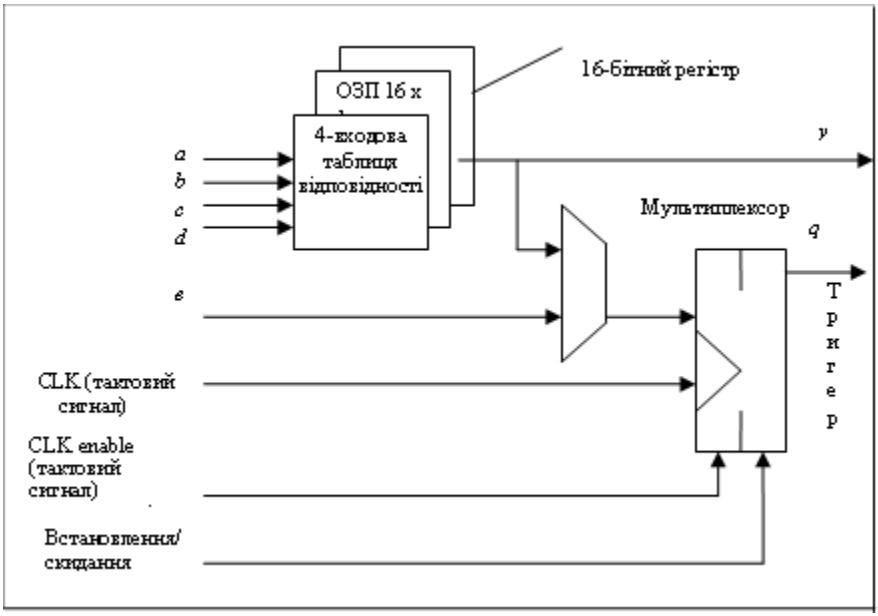


Рис. 4.36. Спрощений вигляд логічної комірки Xilinx

Схема, наведена на рис. 4.35, сильно спрощена, але, проте, вона задовольняє контекст даного матеріалу. Регістр може бути конфігурований для роботи як тригер або як клямка. Полярність тактового сигналу (реакція тригера на фронт або спад синхронізуючого імпульсу) може задаватися програмно, так

само, як полярність сигналів «тактовий сигнал дозволено» і «встановлення/скидання» (активний високий або низький рівень).

Окрім таблиць відповідності, мультиплексорів і регістрів, логічні комірки містять невелику кількість інших елементів, включаючи спеціальну логіку швидкого перенесення для використання в арифметичних діях [63].

Блоки, з яких складаються ПЛІС компанії Altera, називаються логічними елементами (logic element). Між логічними комірками Xilinx і логічними елементами Altera існує ряд відмінностей, але в цілому їх концепції дуже схожі.

4.5.3. Конфігурація логічних блоків ПЛІС (комірки, секції, логічні масиви, функціональні модулі)

Наступним ступенем в ієрархії побудови мікросхем програмованої логіки є так звана, за визначенням фірми Xilinx, секція (slice). Під час введення цієї термінології секція містила дві логічні комірки (рис. 4.36) [62].

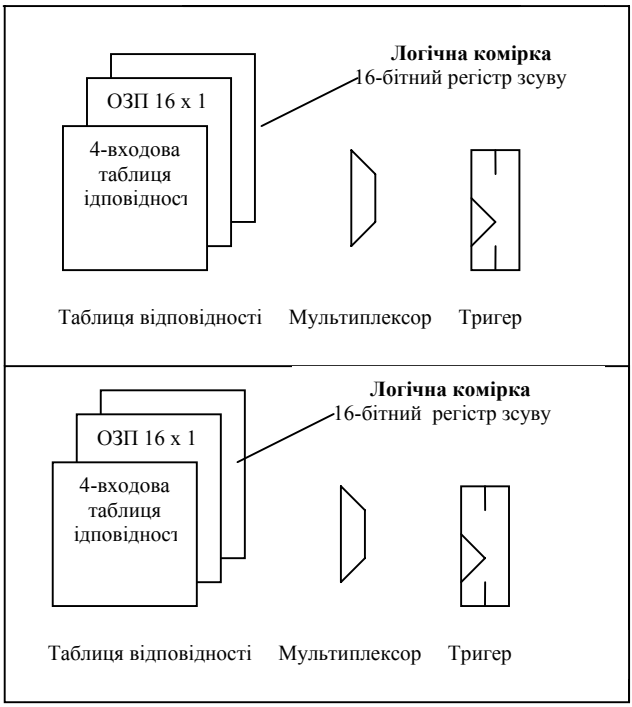


Рис. 4.37. Секція, що містить дві комірки

Компанія Altera і інші постачальники називають цей ступінь якось інакше, використовуючи власні визначення.

На рис. 4.36, для його спрощення, не показані внутрішні зв'язки. Необхідно відзначити, що таблиці відповідності, мультиплексори і регістри кожної логічної комірки мають власні входи і виходи даних. Секція має загальні тактові сигнали, дозвіл тактових сигналів і встановлення/скидання для обох логічних комірок.

Наступний рівень ієрархії, за термінологією компанії Xilinx, має назву «логічний блок КЛБ, що конфігурується» (CLB — configurable logic block). Компанія Altera, у свою чергу, називає його блоком логічних масивів або LAB (LAB — logic array block). Інші постачальники ПЛІС дають їм свої еквівалентні назви [61].

Визначення логічного блока (КЛБ), що конфігурується, змінюється з часом. На початку свого існування КЛБ містили дві 3-входові таблиці відповідності і один регістр. Пізніше в них стали включати дві 4-входові таблиці відповідності і два регістри. Далі кожен блок містив дві або чотири секції, кожна з яких складалася з двох 4-входових таблиць відповідності і двох регістрів. І цей процес постійно триває.

Використовуючи конфігураційні логічні блоки, як приклад можна відзначити, що деякі ПЛІС фірми Xilinx містять по дві секції в кожному блоці, інші пристрої — по чотири секції в кожному блоці. КЛБ візуально можна подати у вигляді осередків програмованої логіки з програмованими з'єднаннями (рис. 4.37) [62].

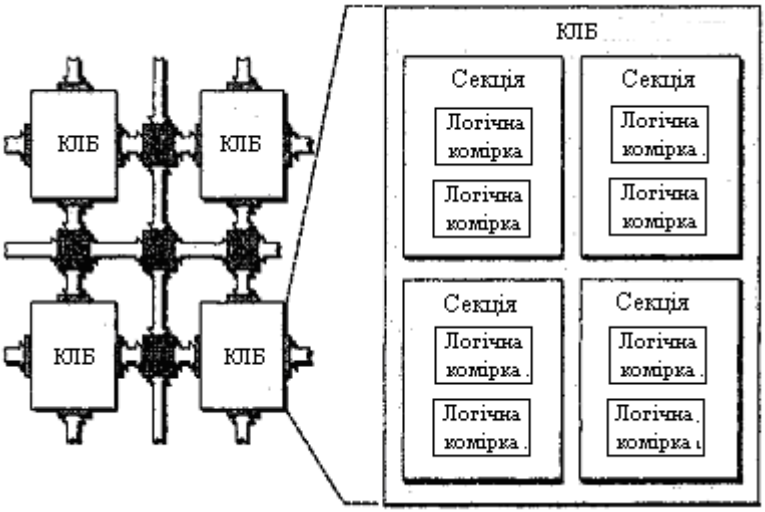


Рис. 4.38. КЛБ, що містить чотири секції

Усередині логічного блока знаходяться швидкі програмовані внутрішні з'єднання. Ці провідники використовуються для з'єднання сусідніх секцій (на рис. 4.37 для простоти викладу вони не показані).

Причина існування такої логіко-блокової ієрархії, тобто логічна комірка → секція з двома логічними комірками → КЛБ з чотирма секціями, полягає у тому, що вона доповнюється еквівалентною ієрархією внутрішніх з'єднань. Тобто, існують швидкі внутрішні з'єднання між логічними комірками усередині секції, потім менш швидкі з'єднання між секціями усередині логічного блока і з'єднання між блоками. Подібна ієрархія відображає покрокове досягнення оптимального компромісу між простотою з'єднання внутрішніх структур і надмірними затримками сигналу на внутрішніх з'єднаннях.

Кожна 4-входова таблиця відповідності може використовуватися як блок ОЗП 16х1. Якщо узяти за основу один чотирисекційний КЛБ (рис. 4.38), та всі таблиці відповідності усередині цього блока можуть бути конфігуровані для реалізації наступних функцій:

- однопортовий блок ОЗП 16х8 бітів;
- однопортовий блок ОЗП 32х4 біти;
- однопортовий блок ОЗП 64х2 біти;
- однопортовий блок ОЗП 128х1 біт;
- двопортовий блок ОЗП 16х4 біти;
- двопортовий блок ОЗП 32х2 біти;
- двопортовий блок ОЗП 64х 1 біт.

Набір сигналів управління і даних, що розглядаються як єдине ціле, зазвичай називають портом. У однопортовому ОЗП дані записуються та зчитуються з комірки через загальну шину даних. У двопортовому ОЗП дані записуються та зчитуються через різні шини (порти). На практиці в цьому випадку операції читання та запису, як правило, працюють із своїми адресними шинами, що використовуються для визначення необхідної комірки усередині ОЗП, тобто операції читання та запису в двопортовому ОЗП можуть виконуватися одночасно.

Кожна 4-бітова таблиця відповідностей може також використовуватися як 16-бітовий регістр зсуву. Для цього існують спеціальні з'єднання між логічними комірками всередині секції і між секціями, які дозволяють з'єднати останній біт одного регістру зсуву з першим бітом іншого регістру без залучення до цього процесу вихідних сигналів таблиць відповідності.

Останні також можуть використовуватися для перегляду вмісту визначеного біта в 16-бітовому регістрі. Це дозволяє при необхідності з'єднати разом таблиці відповідності усередині одного логічного блока і реалізувати регістр зсуву величиною до 128 біт.

Ключовою особливістю сучасних ПЛІС є те, що вони містять спеціальну логіку і внутрішні з'єднання, необхідні для реалізації схем прискореного переносу. У контексті програмованих логічних блоків, розглянутих вище, слід

згадати про те, що кожна логічна комірка містить спеціальну логіку переносу. Ця логіка доповнюється спеціальними внутрішніми з'єднаннями між двома логічними комірками в межах кожної секції, між секціями в рамках кожного логічного блока і між блоками.

Спеціальна логіка швидкого переносу і виділена маршрутизація сприяють виконанню логічних функцій, таких, як лічильники, і арифметичних функцій, таких, як суматори. Можливості схем прискореного переносу сумісно з можливостями інших засобів, аналогічних регістрам зсуву на основі таблиць відповідності, вбудованим помножувачам і іншим блокам, забезпечують необхідний набір засобів для використання ПЛІС в додатках цифрової обробки сигналів (ЦОС).

У процесі реалізації більшості додатків виникає необхідність використовувати елементи пам'яті, тому сучасні ПЛІС містять досить великі блоки вбудованої пам'яті, що називаються блоками вбудованого ОЗП. Залежно від архітектури мікросхеми, ці блоки можуть бути розташовані по периметру кристала, розкидані по його поверхні і відносно ізольовані один від одного або організовані в стовпці, як показано на рис. 4.39.

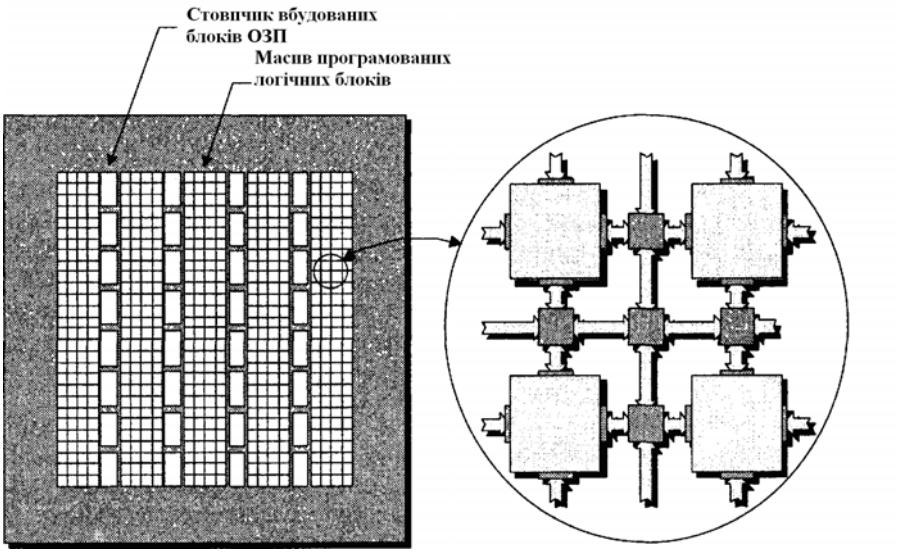


Рис. 4.39. Кристал із стовпцями вбудованих блоків ОЗП

Залежно від пристрою обсяг блоків ОЗП може змінюватися від декількох тисяч до декількох десятків тисяч бітів. Кожна мікросхема може містити від

декількох десятків до декількох сотень таких блоків. Таким чином, повна місткість складає від декількох сотень тисяч бітів до декількох мільйонів бітів.

Кожен блок ОЗП може використовуватися або як незалежний запам'ятовуючий пристрій, або як пристрій, що знаходитися у зв'язці з декількома блоками для реалізації масивів пам'яті великого обсягу. Блоки можуть використовуватися для різних цілей, наприклад, як стандартні одно- і двопортові блоки ОЗП, черги FIFO (first-in first-out), кінцеві автомати і так далі.

Деякі типи функцій, наприклад помножувачі, за своєю суттю є досить повільними, якщо їх реалізовувати з допомогою великої кількості програмованих логічних блоків, об'єднаних разом. Оскільки ці функції використовуються в численних додатках, багато ПЛІС містять спеціальні апаратні блоки множення. Ці блоки зазвичай розташовані в безпосередній близькості від блоків вбудованого ОЗП, оскільки вони часто використовуються разом (рис. 4.40).

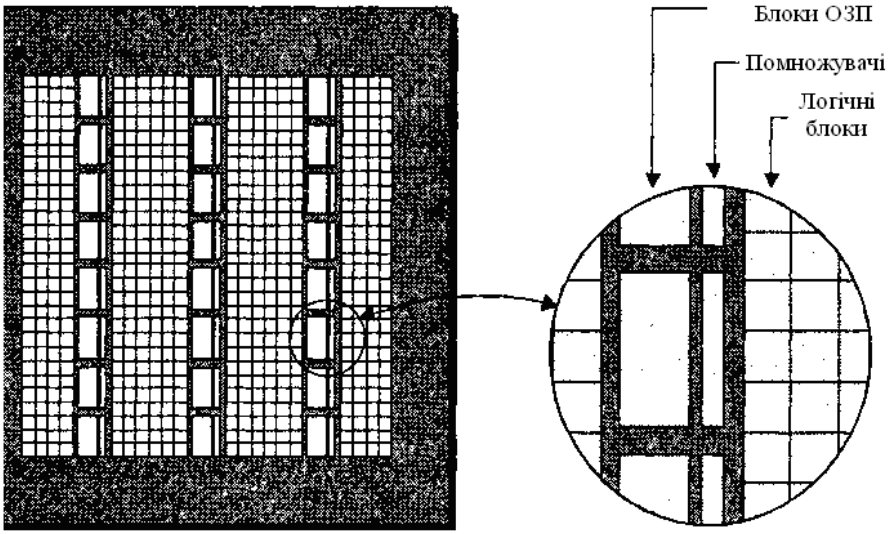


Рис. 4.40. Кристал із вбудованими помножувачами та блоками ОЗП

Деякі виробники ПЛІС також пропонують виділені суматори. У той же час однією з найпоширеніших операцій, що використовуються у додатках цифрової обробки сигналів, є множення з накопиченням {multiply-and-accumulate або MAC) (рис. 4.41). Як підказує назва, ця функція перемножує два числа і підсумовує результат з поточним числом, збереженим в акумуляторі.

При роботі з ПЛІС, яка містить тільки вбудовані помножувачі, для реалізації цієї функції необхідно з'єднати помножувач з суматором, сформованим з

декількох програмованих логічних блоків. Результат зберігатиметься або в тригерах логічних блоків, або в блоках вбудованого ОЗП, або в розподіленому ОЗП. Задача спрощується, коли ПЛІС вже містять вбудовані суматори, а окремі їх види навіть підтримують вбудовані помножувачі з накопиченням.

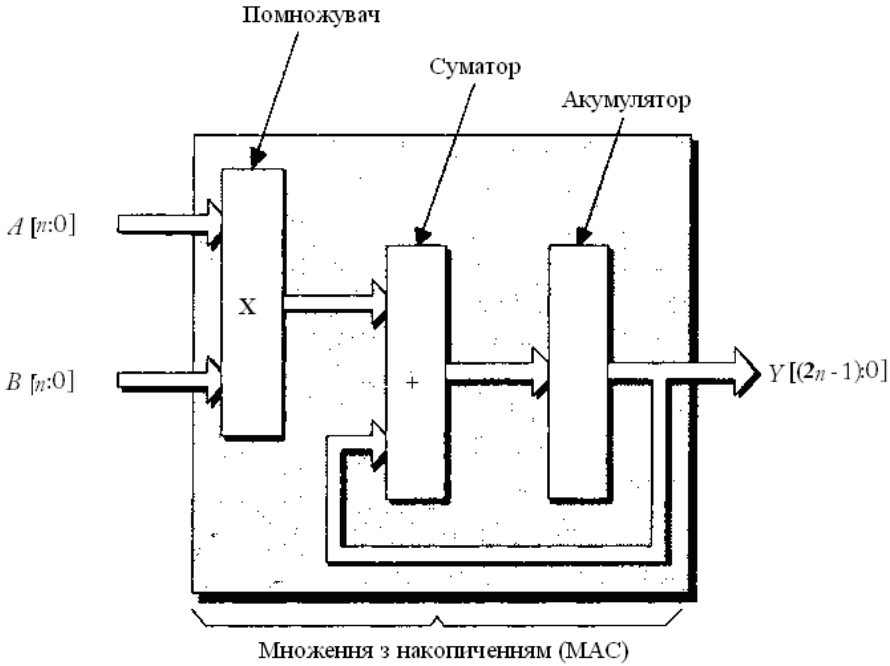


Рис. 4.41. Функції, що формують операцію множення з накопиченням

Важливою особливістю ПЛІС є те, що в них майже всі частини електронного пристрою можуть бути реалізовані або апаратно (з використанням логічних вентилів, регістрів і т. д.), або програмно (у вигляді інструкцій мікропроцесора). Одним з головних критеріїв вибору між апаратною і програмною реалізаціями функції є час, за який ця функція повинна виконувати свої задачі [61]:

• пікосекундна і наносекундна логіка повинна працювати дуже швидко і реалізовуватися апаратно (у структурі ПЛІС);

• мікросекундна логіка є помірно швидкою і може реалізовуватися як апаратно, так і програмно. Це є тип логіки, при якому основна маса часу витрачається на визначення того, яким шляхом піти;

• мілісекундна логіка використовується при реалізації інтерфейсів, таких, як опит стану перемикачів або запалення світлодіодів. Основні зусилля будуть направлені на уповільнення апаратної частини при реалізації цих функцій; для цього, наприклад, використовують величезні лічильники для генерації затримок. Таким чином, часто такі задачі краще реалізовувати в мікропроцесорному коді, оскільки процесор дозволяє знизити швидкість в порівнянні з апаратною частиною, але при цьому задача може виявитися дуже складною.

Фактично більшість пристроїв в тій або іншій формі використовує мікропроцесори. До останнього часу мікропроцесори були дискретними елементами, що розташовуються на друкарській платі. Останнім часом з'явилися високотехнологічні ПЛІС, що містять один або декілька вбудованих мікропроцесорів, які звичайно називаються мікропроцесорними ядрами. При застосуванні таких мікросхем часто має сенс перекласти всі задачі, що виконуються зовнішнім мікропроцесором, на вбудоване ядро. Такий підхід забезпечує ряд переваг, не останніми з яких будуть зниження вартості, усунення великої кількості доріжок, посадочних місць і виводів на друкарській платі, а також зменшення розміру і ваги друкарської плати.

Апаратні мікропроцесорні ядра виготовляються як окремі визначені блоки. Існує два способи інтеграції таких ядер в ПЛІС. Перший передбачає розташування ядра у вигляді смуги (смуга — stripe) уздовж однієї із сторін головної частини, або головної структури ПЛІС (рис. 4.42).

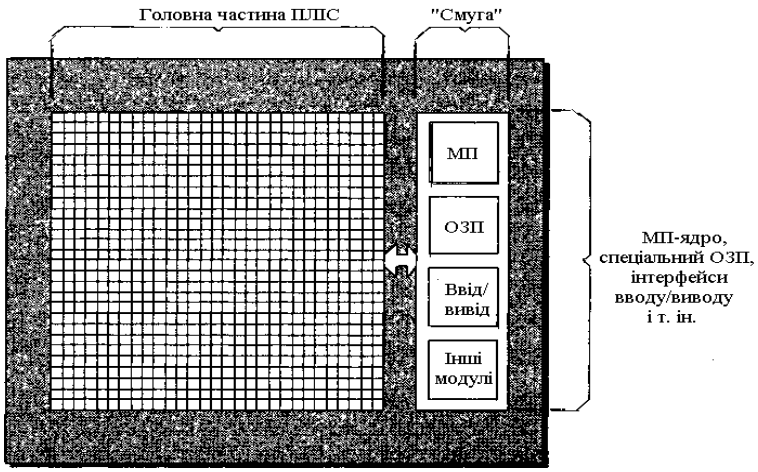


Рис. 4.42. Кристал з вбудованим ядром, що знаходиться за межами головної частини

При такому підході всі компоненти зазвичай формуються на одному кремнієвому кристалі, хоча вони можуть бути виконані і на двох кристалах і

розміщені у вигляді багатокристального модуля (Multichip module — MCM). Головна частина ПЛІС також включає вбудовані блоки ОЗП, помножувачі і інші розглянуті вище блоки, які на цьому рисунку не показані з метою його спрощення.

Перевага такої реалізації виявляється в головній частині (структурі) ПЛІС, яка виходить ідентичною для пристроїв з вбудованим і без вбудованого мікропроцесорного ядра. До того ж постачальники ПЛІС можуть зв'язати всі додаткові функції, такі, як пам'ять, пристрої вводу/виводу і інші, в одну смугу для доповнення мікропроцесорного ядра.

Одним з можливих рішень є вбудовування одного або більш мікропроцесорних ядер прямо в головну частину ПЛІС. Існують мікросхеми з одним, двома і навіть чотирма ядрами (рис. 4.43).

Головна частина ПЛІС містить вбудовані блоки ОЗП, помножувачі і інші розглянуті вище блоки, які на цьому рисунку не показані з метою його спрощення.

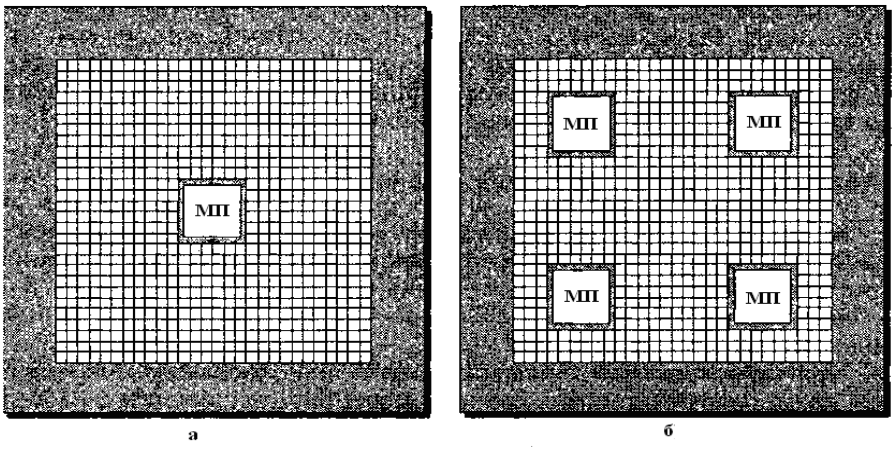


Рис. 4.43. Кристал з ядром, вбудованим до головної частини:
а – з одним вбудованим ядром; б – з чотирма вбудованими ядрами

При використанні цього способу засоби проектування, що застосовуються, повинні враховувати присутність мікропроцесорів у структурі мікросхеми. Пам'ять, що використовується ядром, формується з вбудованих блоків ОЗП, а будь-які функції сполучення реалізуються за допомогою груп програмованих логічних блоків загального призначення. Прихильники цієї схеми стверджують, що розміщення мікропроцесорного ядра в безпосередній близькості до головної частини (структури) ПЛІС забезпечує їй перевагу в швидкості.

Окрім фізичного вбудовування мікропроцесора в структуру кристала, можна конфігурувати групу програмованих логічних блоків для роботи в якості мікропроцесора. Таку групу блоків зазвичай називають програмним ядром, але точніше вони можуть бути класифіковані як програмні і мікропрограмні, залежно від способу, за допомогою якого функціональність мікропроцесора реалізована логічними блоками. Програмні ядра простіші і повільніші, ніж їх апаратні аналоги. Проте у них є одна перевага — при необхідності можна реалізувати ядро або декілька ядер в тому обсязі, якого можна досягти поки не будуть вичерпані всі ресурси у вигляді програмованих логічних блоків.

Швидкість роботи програмного ядра звичайно складає 30...50 % від швидкості апаратного ядра.

Всі синхронні елементи всередині ПЛІС, наприклад регістри всередині програмованого логічного блока, конфігуровані для роботи у вигляді тригерів, необхідно синхронізувати за допомогою тактового сигналу. Тактові сигнали звичайно виробляються за межами мікросхеми і надходять до неї через спеціальні входи синхронізації, а потім розподіляються через спеціальні пристрої і подаються на відповідні регістри.

Розглянемо спрощене зображення дерева синхронізації (рис. 4.44, програмовані логічні блоки не показані).

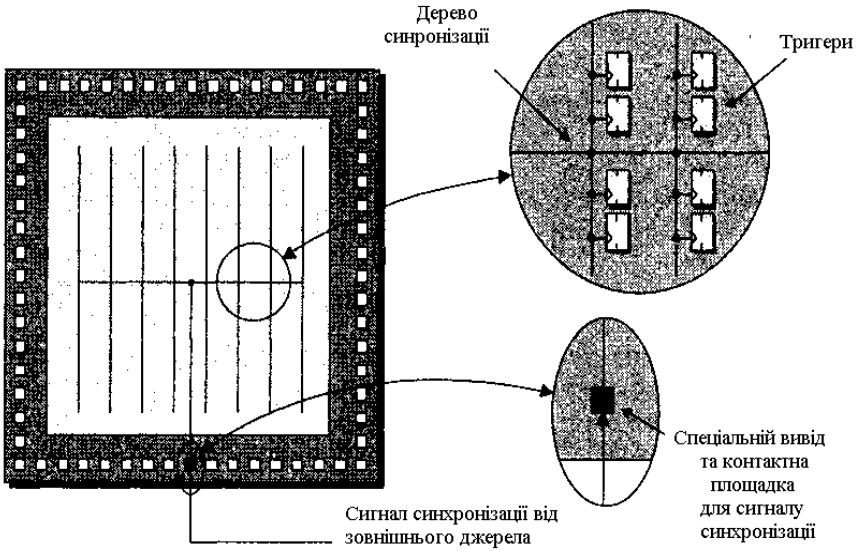


Рис. 4. 44. Просте дерево синхронізації

Назва «дерево синхронізації» виникла тому, що головний синхроімпульс розгалужується подібно до гілок дерева, при цьому тригери можуть розглядатися як «листя» на кінцях віток. Така структура вселяє упевненість у тому, що всі

тригери отримають свої тактові сигнали одночасно, на скільки це можливо. Якби тактові сигнали розповсюджувалися по одному довгому провіднику, синхронізуючи всі тригери по черзі, то тригер, розташований ближче до виводу синхронізації мікросхеми, отримав би синхроімпульс набагато раніше, ніж останні тригери в цьому ланцюжку. Подібна ситуація називається фазовим зсувом, і вона породжує цілий ряд проблем. Навіть при використанні дерева синхронізації може виникнути деякий зсув фаз між регістрами, що знаходяться на одній гілці, а також між гілками.

Дерево синхронізації реалізується за допомогою спеціальних провідників, які відокремлені від внутрішніх з'єднань загального призначення. Принцип дії «дерева синхронізації», розглянутий вище, насправді сильно спрощений. На практиці в мікросхемах існує множина виводів синхронізації (виводи синхронізації, що не використовуються, можуть бути використані як виводи загального призначення), а усередині пристрою є множинні домени синхронізації (дерева синхронізації).

Вивід синхронізації мікросхеми може бути підключений до дерева синхронізації. Проте, як правило, цей вивід підключають не напряму до дерева синхронізації, а до входу пристрою (або блока) управління синхронізацією, що має назву «диспетчер синхронізації», який генерує дочірні тактові сигнали (рис. 4.45).

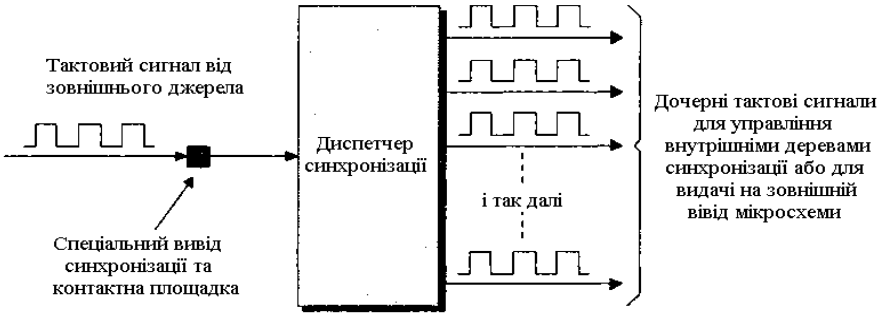


Рис. 4. 45. Диспетчер синхронізації, що генерує дочірні тактові сигнали

Дочірні тактові сигнали можуть використовуватися для управління внутрішніми деревами (доменами) синхронізації або для видачі сигналів на зовнішні виводи мікросхеми, які, у свою чергу, можна використовувати для синхронізації інших пристроїв, розташованих на друкарській платі. Кожне сімейство мікросхем ПЛІС має в своєму розпорядженні власний тип диспетчера синхронізації, і в одному пристрої можуть знаходитися безліч модулів диспетчера синхронізації. Різні диспетчери синхронізації можуть підтримувати всі або тільки деякі з наступних властивостей: усунення флуктуації, частотний синтез, фазовий зсув, автокорекція фазового зсуву.

Усунення флуктуації. Для спрощення прикладу припустимо, що сигнал синхронізації має частоту 1 Мгц. На практиці, звичайно ж, ця частота в багато-багато разів вища. У ідеальному випадку кожен фронт тактового сигналу від зовнішньої системи синхронізації приходить рівно через одну мільйонну частку секунди після попереднього. Проте в реальному житті фронт імпульсу може прийти небагато раніше або небагато пізніше.

Для візуалізації цього ефекту, що називається флуктуацією, зобразимо фронти імпульсів один під іншим, щоб отримати їх суперпозицію; в результаті може вийти «змазаний» тактовий сигнал (рис. 4.46).

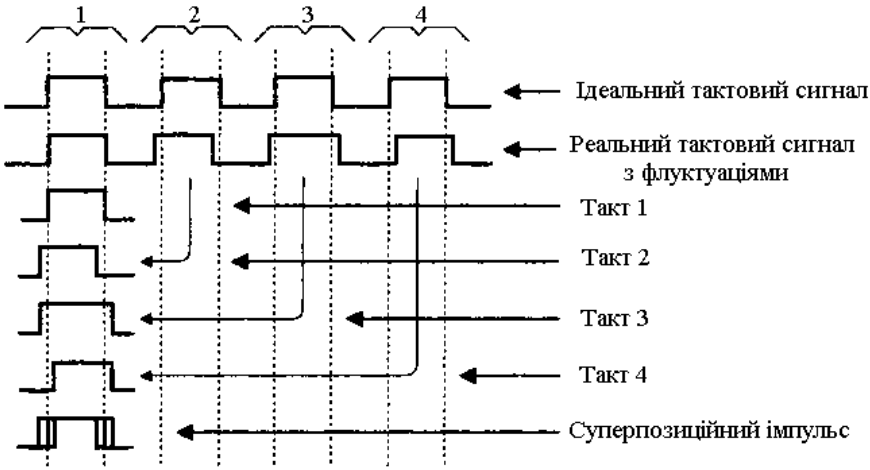


Рис. 4. 46. Флуктуація як результат «змазування» тактового сигналу

Диспетчер синхронізації ПЛІС може використовуватися для виявлення і корекції подібних флуктуації й може забезпечувати очищення дочірніх тактових сигналів для їх використання усередині пристрою .

Частотний синтез. Може трапитися так, що частота тактового сигналу, що надходить на ПЛІС від зовнішнього джерела синхронізації, не відповідає частоті, необхідній для вирішення поставлених задач. У цьому випадку диспетчер синхронізації може генерувати дочірні тактові сигнали, частота яких є похідною величиною від початкового сигналу і виходить шляхом його ділення або множення.

Як простий приклад розглянемо три дочірні тактові сигнали: 1) перший з частотою, еквівалентною початковій частоті тактового сигналу; 2) з частотою, що дорівнює частоті початкової послідовності, помноженої на два; 3) з частотою, що дорівнює половині частоти початкового сигналу (рис. 4.47).

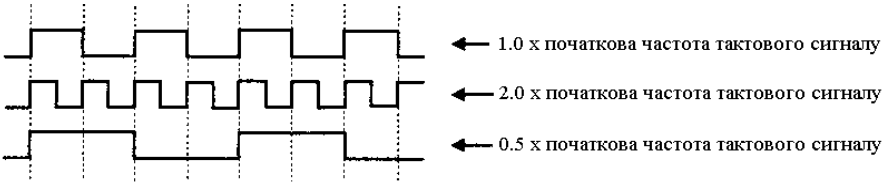


Рис. 4. 47. Використання диспетчера синхронізації для частотного синтезу

На рис. 4.47 наведено дуже простий приклад. На практиці ПЛІС можуть синтезувати всі типи внутрішніх тактових сигналів, наприклад чотири п'ятих від частоти первинної послідовності тактового сигналу.

Фазовий зсув. Деякі вироби вимагають використання тактових сигналів, фаза в яких зсунута (затримана) по відношенню один до одного. Деякі диспетчери синхронізації дозволяють вибирати загальні фіксовані значення фазових зсувів, наприклад 120° і 240° (для трифазної системи синхронізації) або 90°, 180° і 270° (для чотирифазної системі синхронізації). Інші диспетчери дозволяють конфігурувати точне значення фазового зсуву на вимогу користувача для кожного дочірнього тактового сигналу.

Припустимо, що ми отримуємо чотири внутрішні тактові сигнали з головної послідовності тактових сигналів, де перший перебуває у фазі з початковим сигналом, другий зсунутий по фазі на 90°, третій – на 180° і т.ін. (рис. 4.48).

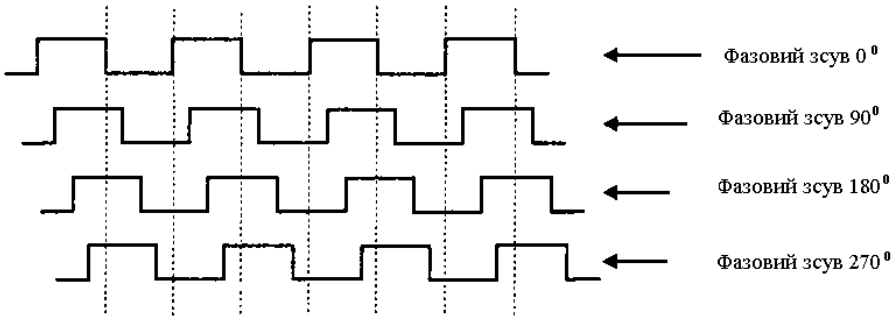


Рис. 4.48. Використання диспетчера синхронізації для фазового зсуву дочірніх тактових сигналів

Автокорекція зсуву. З метою спрощення припустимо, що йдеться про дочірні тактові сигнали, сформовані на тій же частоті і з тією ж фазою, що і головний тактовий сигнал, що приходить на вхід ПЛІС. Проте диспетчер синхронізації, за визначенням, додаватиме до сигналів деяку затримку. Крім того, ще більші затримки додають логічні вентилі і внутрішні з'єднання, що використовуються в розподілі тактових сигналів. Якщо не вжити корекцію,

то дочірні тактові сигнали відставатимуть від вхідних тактових сигналів на деяку величину. Різниця між фазами двох сигналів називається фазовим зсувом.

Залежно від того, як головні і дочірні тактові сигнали використовуються ПЛІС і рештою елементів друкарської плати, зсуви фаз можуть стати причиною різних проблем. Тому диспетчер синхронізації може містити спеціальний вхід для подачі на нього дочірнього тактового сигналу. В цьому випадку диспетчер синхронізації порівнює два сигнали і точно додає певну затримку до дочірніх сигналів, достатню для вирівнювання їх з початковим тактовим сигналом (рис. 4.49).

У такий спосіб буде очищений тільки первинний, тобто з нульовим фазовим зсувом, дочірній сигнал, а всі інші дочірні сигнали будуть фазовані вже щодо цього сигналу.

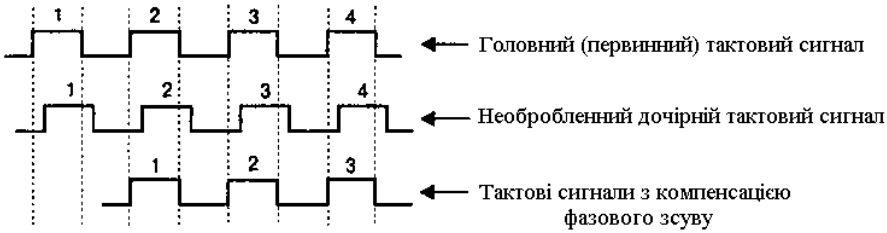


Рис. 4.49. Компенсація фазового зсуву відносно первинного тактового сигналу

Деякі диспетчери синхронізації ПЛІС працюють за принципом фазового автопідстроювання частоти (ФАПЧ), інші – за принципом цифрового автопідстроювання по затримці (digital delay-locked loop — DLL). Системи ФАПЧ можуть бути реалізовані аналоговим або цифровим способом, а системи цифрового автопідстроювання по затримці за своєю природою можуть бути тільки цифровими. Прихильники ФАПЧ стверджують, що ці системи мають переваги в точності, стабільності і споживаній потужності, в нечутливості до шумів і флуктуацій.

Сучасні ПЛІС можуть містити 1000 і більш виводів, які розташовуються по всьому корпусу мікросхеми. Вони також підходять до кристала усередині корпусу мікросхеми. Всередину корпусу кристал встановлюють в перевернеиому вигляді. Це дозволяє під'єднувати загальний дріт, провідники живлення, синхронізації і сигналів вводу/виводу до будь-якої точки на його поверхні. Припустимо, що всі виводи на кристалі розташовані по його контуру, як і було насправді багато років тому.

Якщо подивитися на електронний пристрій з погляду інженерів, що розробляють друкарську плату, залежно від того, що вони роблять, від типів використовуваних пристроїв, від навколишнього середовища, в якому працюватиме

плата і так далі, розробники вибиратимуть певний стандарт передачі даних. (У цьому контексті поняття «стандарт» відноситься до електричних параметрів сигналів, таких, як рівень логічного 0 або логічної 1 у вольтах.)

Проблема у тому, що існує велика різноманітність стандартів, і було б важко створювати спеціальні ПЛІС для підтримки кожного варіанта. З цієї причини ввід/вивід загального призначення ПЛІС може бути конфігурований для прийому і передачі сигналів, відповідних до будь-якого стандарту. Ці сигнали вводу/виводу розділяються на декілька банків. Вважатимемо, що існує вісім банків, пронумерованих від 0 до 7 (рис. 4.50).

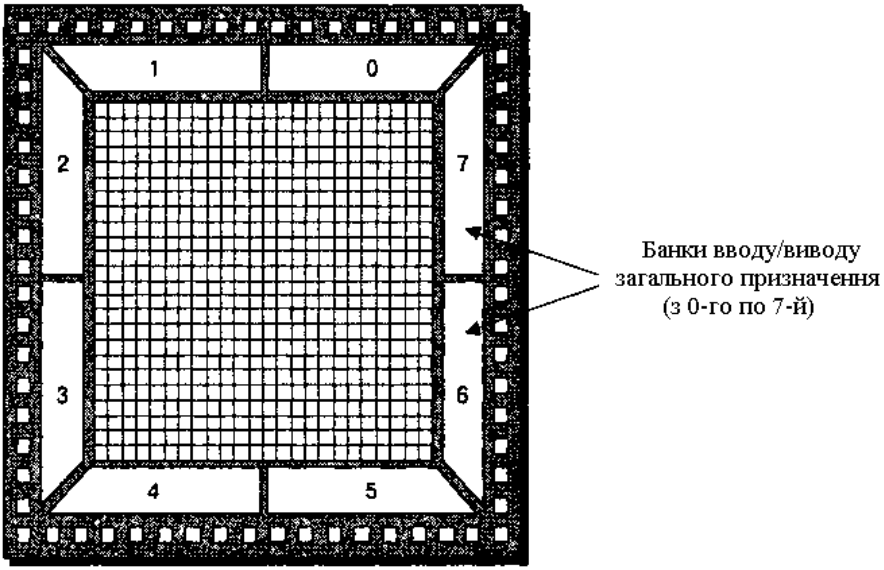


Рис. 4. 50. Вид на кристал з банками вводу/виводу

Кожен банк може бути індивідуально конфігурований для підтримки визначених стандартів вводу/виводу. Таким чином, ПЛІС може працювати з пристроями, використовуючи численні стандарти вводу/виводу. Ці мікросхеми також можуть служити інтерфейсом між різними стандартами вводу/виводу, і здійснювати зв'язок між різними протоколами, які можуть базуватися на різних електричних стандартах.

Сигнали в сучасній друкарській платі часто мають швидкий час перемикання. Мається на увазі час, який необхідний сигналу для перемикання з одного логічного рівня на інший. Щоб запобігти віддзеркаленню сигналів, що приводить до виникнення пульсацій, до виводів

мікросхеми, тобто до входу або виходу, необхідно підключити відповідні узгоджувальні резистори, тобто термінатори.

У минулому ці резистори застосовувалися як окремі компоненти, які розміщувалися на друкарській платі за межами ПЛІС. Проте такий підхід ставав все більш проблематичним, оскільки кількість виводів у мікросхеми збільшувалася, а крок, тобто відстань між ними, зменшувався. У зв'язку з цим сучасні ПЛІС дозволяють використовувати внутрішні узгоджувальні резистори, або внутрішні термінатори, опір яких може задаватися користувачем, для узгодження різного устаткування на друкарській платі і різних стандартів вводу/виводу.

У минулі часи, приблизно до 1995 року, більшість цифрових мікросхем використовувала напругу землі 0 В і напругу живлення +5 В. Крім того, сигнали вводу/виводу перемикалися між рівнями 0 В (логічний 0) і +5 В (логічна 1).

З часом розміри структур на кремнієвих кристалах ставали меншими, оскільки транзистори менших розмірів були дешевшими, а також більш швидкими і споживали менше енергії. Проте для цього вимагалося знизити напруги живлення. Зміна напруги живлення відображена у табл.. 4.2 [61].

Таблиця 4.2 – Залежність напруги живлення від технологічного процесу

Рік	Джерело живлення (напруга ядра) [В]	Технологічний процес [нм]
1998	3,3	350
1999	2,5	250
2000	1,8	180
2001	1,5	150
2003	1,2	130

Джерела живлення, що зазначені у цій таблиці, використовується для живлення внутрішньої логіки ПЛІС, тому ця напруга називається напругою ядра (насправді для підключення живлення і «землі» використовуються декілька виводів мікросхеми). Різні стандарти вводу/виводу використовують сигнали з напругами, рівні яких значно відрізняються від напруги ядра, тому кожен банк вводу/виводу даних може мати свій додатковий вивід живлення.

Починаючи з 350 нм процесу, напруга ядра змінювалася прямо пропорційно технологічному процесу. Проте існують фізичні причини, що не дозволяють напрузі опуститися значно нижче 1 вольт (ці причини засновані на технологічних аспектах, таких, як вхідний поріг перемикання транзисторів і перепад напруг).

4.6. Методи та засоби програмування архітектури ПЛІС

У кожного виробника ПЛІС своя термінологія, свої методи, засоби роботи. Механізми програмування ПЛІС повинні істотно змінюватися від

сімейства до сімейства. Тому подальші міркування будуть присвячені виключно загальним питанням програмування мікросхем.

4.6.1. Конфігураційні файли

Інструменти та підходи, які можуть бути використані для проектування принципових схем і для реалізації ПЛІС-систем, дозволяють розробникам створити конфігураційний, або бітовий, файл, який містить інформацію, призначену для завантаження в мікросхему, тобто для її програмування з метою виконання певних функцій.

Якщо мікросхема використовує для зберігання своєї конфігурації комірки статичного ОЗП, та конфігураційний файл містить деяку сукупність конфігураційних даних, або біти, які використовуються для визначення стану елементів програмованої логіки і конфігураційних команд, тобто інструкцій, які вказують пристрою, що йому необхідно робити з конфігураційними даними. При завантаженні конфігураційного файла в пристрій інформацію, що передається, називають конфігураційним двійковим потоком [61].

Пристрої на елементах пам'яті ЕСППЗП або Flash-пам'яті програмуються аналогічно пристроям на комірках статичного ОЗП. На відміну від них, в пристроях, заснованих на методі нарощуваних перемичок, конфігураційний файл містить тільки конфігураційні дані, які використовуватимуться для нарощування перемичок.

4.6.2. Конфігураційні комірки

Основна концепція програмування ПЛІС відносно проста і є завантаженням в пристрій конфігураційного файла. Оскільки це досить-таки складний процес, то щоб він був зрозумілий, почнемо з основ. Спершу розглянемо елементарний пристрій, що складається з масиву дуже простих програмованих логічних блоків, оточених програмованими внутрішніми з'єднаннями (рис. 4.29) [61].

Всі можливості програмованих пристроїв реалізуються за допомогою спеціальних конфігураційних комірок. Більшість ПЛІС використовує комірки статичного ОЗП, інші використовують ЕСППЗП або осередки Flash-пам'яті, треті базуються на нарощуваних перемичках.

Незалежно від базової технології, внутрішні з'єднання пристрою містять велику кількість зв'язуючих (з'єднуючих) комірок, які можна використовувати для його конфігурації, щоб з'єднати входи і виходи мікросхеми з програмованими логічними блоками, а також блоки між собою. Всі блоки воду/виводу, які не показані на рис. 4.29, мають в своєму розпорядженні декілька з'єднуючих комірок, які використовуються для їх конфігурації

відповідно до певних стандартів інтерфейсів вводу/виводу або інших параметрів. Припустимо, що кожен програмований логічний блок містить всього лише 4-входову таблицю відповідності, мультиплексор і регістр (рис. 4.51).

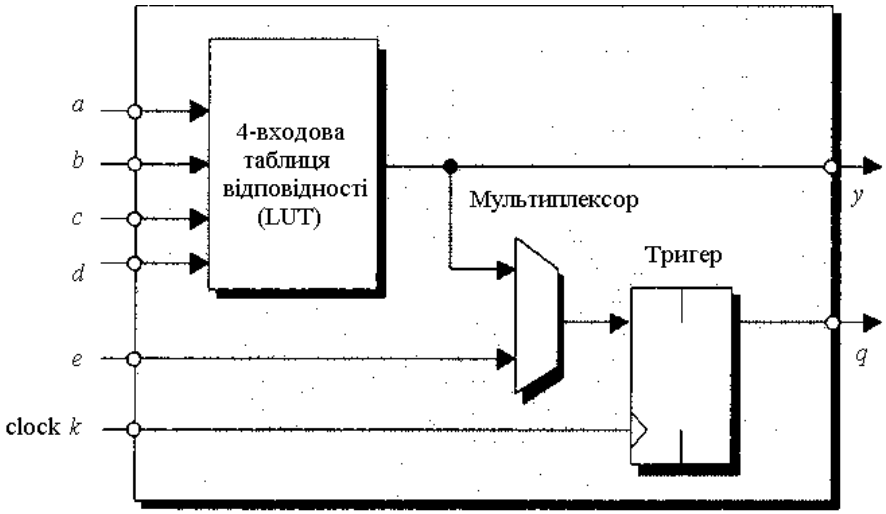


Рис. 4. 51. Простий логічний блок, що програмується

Мультиплексор потребує з'єднуючої конфігураційної комірки, щоб визначити вхід, сигнал з якого передаватиметься на його вихід. Регістр потребує з'єднуючих конфігураційних комірок, щоб визначити, чи буде він:

• виступати в ролі тригера (як показано на рис. 4.51) або в ролі клямки;

• перемикатися по фронту або по спаду синхроімпульсу (при роботі в режимі тригера) або перемикатися високим або низьким рівнем сигналу (у режимі клямки);

• ініціалізуватися логічним 0 або логічної 1.

У свою чергу, 4-входова таблиця відповідності містить 16 конфігураційних комірок.

4.6.3. ПЛІС на нарощуваних перемичках

У мікросхемах на нарощуваних перемичках конфігураційні комірки, як правило, розподілені по всій поверхні пристрою по певних ключових напрямах. Для програмування мікросхема поміщається в спеціальний програматор, в який з управляючого комп'ютера завантажується конфігураційний або бітовий файл. Цей файл використовується програматором

як керівництво по видачі імпульсів щодо високої напруги і великого струму на певні виводи мікросхеми, унаслідок чого відбувається почергове нарощування перемичок.

Щоб краще зрозуміти процес програмування, уявимо, що кожна нарощувана перемичка має віртуальні координати х-у на поверхні кристала і ці значення х-у подаються у вигляді чисел. Далі уявимо, що одна група контактів вводу/виводу мікросхеми призначена для подання значення х, а інша група контактів – для подання значення у. У реальності все виглядає набагато складніше, але цей примітивний приклад дозволяє зрозуміти суть процесу [61].

Після нарощування всіх перемичок мікросхема витягується з програматора і поміщається на друкарську плату. Звичайно ж, пристрої на нарощуваних перемичках є одноразово програмованими, і з початком процесу програмування що-небудь поміняти в конфігурації пристрою буде неможливо.

4.6.4. ПЛІС на комірках статичного ОЗП

Розглянемо ПЛІС-структуру на основі комірок статичного ОЗП. Ці пристрої є енергозалежними. Це означає, що вони програмуються всередині системи, тобто на друкарській платі, причому програмуються кожного разу після включення системи.

Всі конфігураційні комірки статичного ОЗП можна представити у вигляді одного довгого регістра зсуву. Вважатимемо, що початок і закінчення, тобто вхід і вихід цього регістрового ланцюжка напряму підключені до зовнішніх контактів мікросхеми. Проте при цьому слід мати на увазі, що це можливо тільки в тому випадку, коли програмування здійснюється за допомогою комунікаційного порту в режимі послідовного завантаження. При послідовному завантаженні ПЛІС може знаходитися у двох режимах: «ведучій» (master) та «ведений» (slave) [61].

Також слід мати на увазі, що вихідні контакти і вихідні сигнали конфігураційних даних використовуються тільки у тому випадку, коли декілька ПЛІС конфігуровані для роботи в режимі каскадування, тобто послідовного опиту, або якщо з тих або інших причин від пристрою вимагається прочитати конфігураційні дані.

Пристрої на основі елементів пам'яті Flash (і ЕСППЗП) звичайно програмуються таким же способом, яким і мікросхеми на основі статичного ОЗП, але при цьому є незалежними. Ця властивість дозволяє Flash-пристроям зберігати конфігураційну інформацію при відключенні живлення системи. Іншими словами, такі пристрої не треба перепрограмувати при включенні системи, хоча при необхідності така процедура може бути виконана. ПЛІС на основі Flash-елементів можуть програмуватися внутрішньосистемно, тобто на друкарській платі, або позасистемно за допомогою окремого програматора.

Найпростіший спосіб зрозуміти внутрішню структуру програмованих комірок статичного ОЗП полягає в поданні їх у вигляді довгого регістру зсуву. Так було б у тому випадку, коли б кожна комірка була реалізована у вигляді

тригера, а всі тригери були з'єднані в ланцюжок та управлялися загальним синхросигналом.

Проблема полягає у тому, що ПЛІС може містити величезну кількість конфігураційних комірок. Вже до 2003-го року високотехнологічні пристрої могли містити 25 мільйонів таких комірок. Тут необхідно відмітити, що для реалізації тригера потрібно вісім транзисторів, а для клямки – всього лише чотири. Тому конфігураційні комірки в пристроях на статичному ОЗП виконуються на клямках. Так, в пристрої з 25 мільйонами конфігураційних комірок такий підхід дозволяє заощадити 100 мільйонів транзисторів.

Проблема криється у тому, що не можна створити такий великий регістр зсуву з клямок. Постачальники ПЛІС обходять цю проблему за допомогою групи тригерів, скажімо, з 1024-х, що об'єднані загальним синхросигналом і що конфігуруються як класичний регістр зсуву. Таку групу називають фреймом.

Усі 25 мільйонів конфігураційних комірок-клямок в пристрої з нашого прикладу також розбивалися на фрейми, при цьому довжина кожного дорівнювала довжині спеціального фрейму тригера. Зовні процес був простим завантаженням 25 мільйонів бітів конфігураційних даних в пристрій. Проте усередині пристрою, як тільки перші 1024 біти послідовно завантажувалися у фрейм тригера, спеціальна внутрішня схема автоматично паралельно копіювала ці дані в перший фрейм клямок. Потім наступні 1024 біти завантажувалися у фрейм тригера і автоматично паралельно копіювалися в другий фрейм клямок і так далі до повного заповнення пристрою. При читанні даних з пристрою процес перебігав в зворотному порядку.

Програмування ПЛІС може займати значний час. Якщо розглядати високотехнологічну мікросхему, що містить 25 мільйонів комірок статичного ОЗП, конфігурацію такого пристрою в послідовному режимі з тактовою частотою 25 Мгц займуть одну секунду.

4.6.5. Програмування вбудованих блоків розподіленого ОЗП та інших ОЗП

Якщо ПЛІС містить великі вбудовані блоки ОЗП, то ядра цих блоків виконуються з клямок на основі елементів статичної пам'яті. Кожна клямка, у свою чергу, є конфігураційною коміркою, яка формує частину уявного регістрового ланцюжка, який розглядався в попередньому розділі.

Інтерес представляє той факт, що кожна 4-входова таблиця відповідності (рис. 4.49) може конфігуруватися для роботи як таблиця відповідності, або як невеликий блок (16x1) розподіленого ОЗП, або як 16-бітовий регістр зсуву. Всі ці конфігурації використовують для своєї роботи групу з 16 клямок на основі статичного ОЗП, причому кожна клямка є конфігураційною коміркою з складу розглянутого регістрового ланцюжка.

Особливість схеми полягає у тому, що в цьому випадку використовується концепція ємнісних клямок, які за багатьма параметрами перевершують класичні клямки. Дуже схожим способом розробники робили зовнішні тригери

з дискретних транзисторів, резисторів і конденсаторів на початку 60-х років минулого століття [61].

4.6.6. Мультипрограмування конфігураційних ланцюжків

Оскільки кількість конфігураційних осередків може досягати декількох десятків мільйонів, ланцюжок дійсно може виявитися дуже довгим. У деяких мікросхемах ланцюжок розбивається на частини, і окремі ділянки цього ланцюжка підключаються до конфігураційного порту. Такий підхід дозволяє конфігурувати окремі частини пристрою і спрощує реалізацію різних концепцій, таких, як модульне і покрокове проектування

Регістри програмованих логічних блоків сполучені з конфігураційними осередками, в яких міститься їх початкове значення: логічний 0 або логічна 1. Кожне сімейство ПЛІС підтримує деякий механізм, наприклад вивід ініціалізації, який при активації дає команду регістрам повернутися до початкових значень. Такий механізм не ініціалізує вбудовані блоки пам'яті або розподілене ОЗП [61].

4.6.7. Конфігураційний порт

Перші ПЛІС використовували інструмент, що називається конфігураційним портом. Навіть сьогодні, коли доступні більш складні методи, подібні до JTAG-інтерфейсу, конфігураційний порт все ще знаходить широке застосування, оскільки є досить простим інструментом.

Почнемо розгляд з невеликої групи спеціальних контактів режиму конфігурації, які використовуються для установлення режиму конфігурації пристрою. У перших ПЛІС для цих цілей використовувалися тільки два виводи, які дозволяли задавати чотири режими конфігурації (табл. 4.3) [61].

Таблиця 4.3 – Чотири первинних режими ініціалізації

Виводи режиму установлення	Найменування режиму
00	Послідовне завантаження, ПЛІС у режимі «ведучий»
01	Послідовне завантаження, ПЛІС у режимі «ведений»
10	Паралельне завантаження, ПЛІС у режимі «ведучий»
11	Паралельне завантаження, ПЛІС у режимі «ведений»

Слід мати на увазі, що назви режимів, а також відповідність коду на виводах установлення режиму до назви режиму приведені тільки як приклад. Справжні коди і назви режимів у кожного постачальника свої.

Виводи, або контакти, установлення режиму звичайно підключаються до необхідних значень логічного 0 або логічної 1 на друкарській платі. Ці виводи можуть підключатися до інших логічних пристроїв, які дозволяють змінювати режим програмування, але така схема рідко зустрічається на практиці.

Окрім виводів установлення режиму, використовується додатковий вивід для інформування ПЛІС про початок процесу конфігурації, і ще один вивід для

видачі сигналу, що свідчить про закінчення процесу конфігурації. Існують також сигнали для визначення помилок, що виникають у ході конфігурації. Крім конфігурації ПЛІС при включенні системи, при необхідності вона може бути реініціалізована і повернена до первинних конфігураційних даних.

Конфігураційний порт також використовує додаткові виводи мікросхеми для управління завантаженням даних і для їх вводу. Кількість цих виводів залежить від вибраного режиму конфігурації. Важливе значення має той факт, що при завершенні конфігурації більшість цих виводів може використовуватися як входи/виходи загального призначення.

4.6.8. Послідовне завантаження, ПЛІС у режимі «ведучий»

Цей режим є найпростішим зі всіх режимів програмування. Раніше для роботи в цьому режимі використовувалися зовнішні мікросхеми ППЗП. Згодом їх замінили СППЗП, потім ЕСППЗП а тепер, як правило, використовуються пристрої на осередках flash-пам'яті. Ці компоненти спеціальної пам'яті містять один вивід для виходу даних, який під'єднується до контакту вводу конфігураційних даних ПЛІС (рис. 4.52) [61].

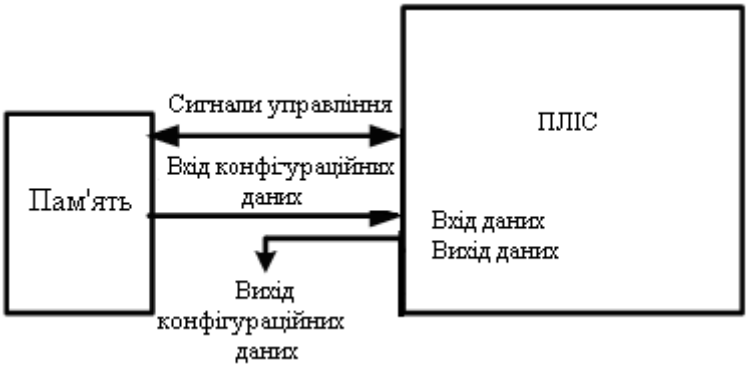


Рис. 4.52. Послідовне завантаження, ПЛІС у режимі «ведучий»

ПЛІС також використовує декілька сигналів для управління зовнішньою пам'яттю. До них відноситься сигнал скидання, який видається ПЛІС при готовності нею почати читання даних, а також синхросигнал, призначений для синхронізації конфігураційних даних.

У цьому режимі ПЛІС не потребує зовнішньої пам'яті з послідовною адресацією. При такому режимі використовуються прості імпульси сигналу скидання для передачі сигналу готовності почати читання даних з початку конфігураційної послідовності, а потім застосовується послідовність синхроімпульсів для синхронізації видачі конфігураційних даних з пам'яті.

При необхідності вихідний сигнал з ПЛІС може використовуватися для читання конфігураційних даних пристрою. Такий підхід має місце при

використанні декількох мікросхем ПЛІС на одній друкарській платі. В цьому випадку кожна мікросхема може мати власний виділений пристрій зовнішньої пам'яті і конфігуруватися окремо, як показано на рис. 4.52. Згідно з іншим підходом, ПЛІС можуть з'єднуватися каскадом, тобто за схемо послідовного опитування, і використовувати спільно один пристрій зовнішньої пам'яті (рис. 4.53).

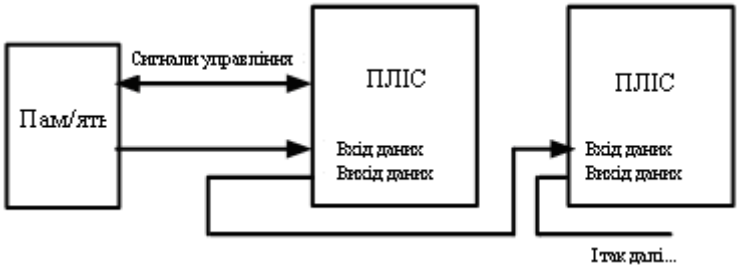


Рис. 4.53. Послідовне включення мікросхем FPGA

У схемі, показаній на рис. 4.53, перша ПЛІС в ланцюжку (вона з'єднується напряму із зовнішньою пам'яттю) конфігуруватиметься для роботи в послідовному режимі як ведуча (master). Усі подальші ПЛІС в цьому ланцюжку повинні бути конфігуровані для роботи в послідовному режимі як ведені (slave) [61].

4.6.9. Паралельне завантаження, ПЛІС у режимі «ведучий»

У багатьох відношеннях цей режим подібний до попереднього, але при цьому дані читаються 8-бітовими фрагментами з пам'яті, яка містить також вісім виводів даних. Групи з восьми бітів досить широко поширені і називаються байтом.

Окрім забезпечення управляючих сигналів, ПЛІС формує для зовнішньої пам'яті сигнали адреси, що використовуються для визначення байта конфігураційних даних, який завантажуватиметься на наступному такті (рис. 4.54).

Подібна схема працює за умови, що ПЛІС містить внутрішній лічильник, який використовується для генерації адреси для зовнішньої пам'яті. Перші ПЛІС містили 24-бітові лічильники, які дозволяли їм адресувати 16 мільйонів байтів даних. На початку конфігураційної послідовності лічильник скидався в нуль. Після того, як байт даних лічильника адреси був зчитаний, вміст лічильника збільшувався для адресації наступного байта. Цей процес тривав доти, поки конфігураційні дані повністю не завантажувалися в мікросхему [61].

Рис. 4.54. Паралельне завантаження в режимі «ведучий» (перший спосіб)

Здавалося б, що такий метод паралельного завантаження має перевагу в швидкості в порівнянні з розглянутим раніше послідовним способом. Проте спочатку це було не так. У перших мікросхемах, як тільки байт даних зчитувався ПЛІС, він, як і раніше, послідовно передавався у внутрішній конфігураційний регістр зсуву. Ця ситуація була виправлена в більш сучасних пристроях.

Спеціальні пристрої пам'яті, що створюються для використання сумісно з ПЛІС в даний час, є відносно недорогими і виготовляються за Flash-технологіями, тобто є пристроями багатократного використання. Сучасні ПЛІС використовують нові варіанти паралельного завантаження. У цих випадках зовнішня пам'ять є спеціальним пристроєм, який не вимагає зовнішньої адресації, тобто для цих цілей ПЛІС більше не потребує внутрішнього лічильника (рис. 4.55) [61].

Рис. 4.55. Паралельне завантаження в режимі «ведучий» (сучасна реалізація)

В цьому випадку так само, як і в послідовному режимі, ПЛІС просто видає сигнал скидання зовнішньої пам'яті для визначення того, що вона готова почати читання даних з початку конфігураційної послідовності, а потім видає синхроімпульси для синхронізації конфігураційних даних, що надходять із зовнішньої пам'яті.

4.6.10. Паралельне завантаження, ПЛІС у режимі «ведений»

Розглянуті вище режими конфігурації, в яких ПЛІС виступала в ролі ведучої (master), досить привабливі завдяки своїй простоті і невибагливості, оскільки для роботи потрібна тільки ПЛІС і одна мікросхема зовнішньої пам'яті.

Проте на багатьох друкарських платах містяться мікропроцесори, які звичайно використовуються для виконання різноманітних внутрішніх задач. В цьому випадку розробники можуть вирішити використовувати мікропроцесори для завантаження конфігураційних даних у ПЛІС (рис. 4.56) [61].

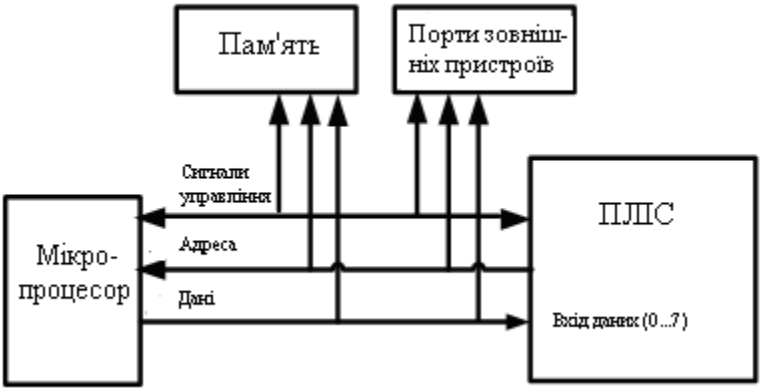


Рис. 4.56. Паралельне завантаження у режимі «ведений»

Ідея полягає у тому, що мікропроцесор є управляючим пристроєм. У цьому режимі мікропроцесор повідомляє ПЛІС, що він готовий почати процес конфігурації, після чого він здійснює читання байта конфігураційних даних з певного пристрою пам'яті або із зовнішнього пристрою, або з інших пристроїв, записує ці дані в ПЛІС, зчитує наступний байт даних з пам'яті, записує його в ПЛІС і так до закінчення процесу конфігурації.

Такий підхід має ряд переваг, не останньою з яких є те, що мікропроцесор може використовуватися для запиту устаткування, що належить сусідній системі, і обирати за певною ознакою конфігураційні дані для завантаження в ПЛІС.

4.6.11. Послідовне завантаження, ПЛІС у режимі «ведений»

Цей режим аналогічний паралельному, але при цьому для завантаження даних в ПЛІС використовується тільки один біт. Мікропроцесор, як і раніше,

здійснює читання одного байта даних з пристрою пам'яті і потім перетворює цей байт в послідовність бітів для запису в ПЛІС.

Головною перевагою такого підходу є те, що його реалізація вимагає меншої кількості виводів, задіяних у ПЛІС. Іншими словами, у процесі конфігурації використовується тільки один контакт вводу/виводу, який за допомогою додаткового провідника підключається до шини даних мікропроцесора [61].

4.6.12. JTAG-порт

Багато сучасних пристроїв, у тому числі і ПЛІС, обладнано спеціальним інтерфейсом, який називається JTAG-порт. Цей порт підтримується Об'єднаною робочою групою по автоматизації тестування (Joint Test Automation Group, JTAG) і відомий як стандарт IEEE 1149.1; спочатку він розроблявся для реалізації методу периферійного сканування, який застосовувався для тестування друкарської плати та цифрових мікросхем.

У даному контексті важливо, що ПЛІС має деяку кількість виводів-контактів, які використовуються як JTAG-порт. При цьому один з цих виводів використовується для вводу даних, інший – для виводу, а інші – для зв'язку з послідовно сполученими JTAG-регістрами або тригерами.

Ідея послідовного сканування полягає в наступному: через JTAG -порт можна послідовно передавати дані в JTAG-регістри, які зв'язані з вхідними виводами мікросхеми. У результаті пристрій, в даному випадку ПЛІС, дістає можливість працювати з цими даними, зберігати результати цієї роботи в JTAG -регистрах, зв'язаних з вихідними виводами мікросхеми, і послідовно видавати ці результати назад в JTAG-порт.

JTAG-пристрої містять різну додаткову управляючу логіку. У випадку ПЛІС JTAG-порт, окрім операцій послідовного сканування, може виконувати і інші функції. Наприклад, він може видавати спеціальні команди, які завантажуються в спеціальний командний JTAG-регістр через вхід JTAG-порту. Одна така команда указує ПЛІС під'єднати свій внутрішній конфігураційний регістр зсуву до скануючого ланцюжка JTAG-порту. В цьому випадку JTAG-порт може використовуватися для програмування ПЛІС. Таким чином, сучасні ПЛІС підтримують п'ять різних режимів програмування, і, отже, для визначення того, який з них використовуватиметься, потрібно вже три виводи, як показано у табл. 4.4 (в майбутньому також можуть з'явитися і додаткові режими).

Можна відмітити, що JTAG-порт завжди доступний до використання, тобто пристрій може бути спочатку конфігурований через звичний конфігураційний порт, шляхом установлення одного із стандартних конфігураційних режимів, а потім, при необхідності, може бути реконфігурований за допомогою JTAG-порту. Також існує можливість програмувати пристрій за допомогою тільки JTAG -порта [61].

Таблиця 4.4 – П'ять сучасних конфігураційних режимів

Виводи режиму установлення	Найменування режиму
000	Послідовне завантаження, ПЛІС у режимі «ведучий»
001	Послідовне завантаження, ПЛІС у режимі «ведений»
010	Паралельне завантаження, ПЛІС у режимі «ведучий»
011	Паралельне завантаження, ПЛІС у режимі «ведений»
1xx	Використовується тільки JTAG-порт

Контрольні питання

1. Що таке ПЛІС?
2. Що таке замовні інтегральні схеми ASIC і ASSP?
3. Сфери застосування ПЛІС.
4. У чому суть методів плавких та нарощуваних перемичок?
5. Структура комірки ППЗП на основі транзистора з плавкою перемичкою.
6. Що таке ППЗП з електричним стиранням?
7. Класифікація архітектурних особливостей ПЛП.
8. Основні фірми-виробники та постачальники ПЛІС.
9. Класифікація спеціалізованих замовних ВІС (ASIC).
10. Що таке вентильна матриця?
11. Що таке канальна логічна матриця?
12. З чого складається типова спеціалізована структурована ІС?
13. З чого складається простий програмований логічний блок?
14. Структура логічного блока на мультиплексорах.
15. Що являє собою ланцюжок конфігураційних комірок?
16. Архітектура базисного модуля, що конфігурується, (логічна комірка фірми Xilinx).
17. Що таке логічний блок, що конфігурується (КЛБ)?
18. Що таке дерево синхронізації ПЛІС?
19. Які функції виконують диспетчери синхронізації?
20. Що таке банки вводу/виводу?
21. Що таке конфігураційний файл?
22. Що таке конфігураційна комірка?
23. Структура ПЛІС на нарощуваних перемичках.
24. Структура ПЛІС на комірках статичного ОЗП.
25. Суть паралельного та послідовного завантаження ПЛІС у режимі «ведучий»?
26. Суть паралельного та послідовного завантаження ПЛІС у режимі «ведений»?
27. Що таке JTAG-порт?
28. Перелік конфігураційних режимів ПЛІС.