Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

ОПОРНИЙ КОНСПЕКТ ЛЕКЦІЙ З КУРСУ

«Проектування комп'ютерних систем на програмованих логічних мікросхемах»

з освітньо-кваліфікаційного рівня «спеціаліст»
спеціальність «Комп'ютерні системи та мережі»

Тернопіль – 2012

Опорний конспект лекцій з курсу «Проектування комп’ютерних систем на програмованих логічних мікросхемах» з освітньо-кваліфікаційного рівня «спеціаліст» спеціальності «Комп’ютерні системи та мережі» / Ляпандра А.С., Майків І.М. – Тернопіль: ТНЕУ, 2012.–53 с.

Укладачі: А.С.Ляпандра, доцент кафедри КІ

Відповідальний за випуск: Березький О.М., д.т.н., доцент, завідувач кафедри комп’ютерної інженерії

Рецензенти: Р.Б..Трембач, к.т.н., доцент кафедри КІ
В.В.Яцків, к.т.н., доцент кафедри СКС

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри комп’ютерної інженерії протокол № __ від __ квітня 2012 р.

Вступ

На сучасному етапі розвитку інформаційних технологій важливе місце займає синтез цифрових пристроїв на основі функціональних моделей. Його суть полягає в автоматизованому формуванні логічної структури пристрою на основі необхідної поведінки. Логічна структура описується інженером за допомогою спеціальної мови програмування - HDL (Hardware Description Language - мова опису обладнання та апаратного забезпечення). Найбільшого поширення серед мов класу HDL набули такі мови як ABEL, Verilog та VHDL. На лекціях та лабораторних заняттях з курсу «Проектування комп'ютерних систем на програмованих логічних мікросхемах» буде вивчено мову VHDL, оскільки вона широко поширена в Європі.

Потреба застосування мов опису обладнання та апаратного забезпечення виникла внаслідок необхідності створення складних систем, проектування яких без застосування ЕОМ стало нераціональним. Існує стійка тенденція до створення складних системи, в тому числі й комп'ютерних, на програмованих логічних мікросхемах.

Лідерами у їх виробництві є такі фірми, як Xilinx, Altera, Lattice та Actel. Мікросхеми виготовляють за технологіями FPGA (Field Programmable Gate Array -- програмована користувачем вентильна матриця) та CPLD (Complex Programmable Logic Device -- комплексний програмований логічний пристрій).

Перевагою FPGA у порівнянні з CPLD є висока кількість системних вентилів та низьке енергоспоживання. Переваги CPLD полягають у більш високій швидкодії та забезпеченні можливості встановлення захисту від копіювання.

На основі ПЛІС-FPGA створюють широкий спектр електронних пристроїв, серед яких: засоби поєднання різних за живленням інтерфейсів, перетворювачі кодів, периферійні контролери, мікропрограмні пристрої керування, скінченні автомати, універсальні та спеціалізовані процесори, пристрої цифрової обробки сигналів тощо, оскільки рівень напруги на їх блоках введення/виведення (БВВ) може бути різним.

1. Лекція «Напрями розвитку елементної бази комп'ютерних систем.»

Швидкий розвиток обчислювальних технологій зумовлений не тільки успіхом в розробці нових алгоритмів, але і стрімким прогресом технології виготовлення елементної бази. Сучасні досягнення в області обчислювальних систем істотно змінили їх склад і архітектуру. Нині для реалізації алгоритмів застосовуються комп'ютерні системи з нетрадиційною архітектурою, що складаються з численних спеціалізованих і функціонально орієнтованих процесорів реалізованих на базі ПЛІС. Подальший розвиток архітектури ком-п'ютерних систем тісно пов'язаний з розвитком ПЛІС-технології, яка дозволяє зменшити енергоспоживання, масогабаритні характеристики, підвищити швидкодію та стимулює розвиток одних підходів проектування і відкидає інші. Створення ефективних архітектур комп'ютерних систем пов'язано з розвитком елементної бази у трьох напрямках:

- однокристальні програмовані мікропроцесори та мікро-ЕОМ з архітектурою орієнтованою на розв'язання обчислювальних задач;
- трансп'ютери та НВІС однорідних обчислювальних середовищ, які орієнтовані на масово- паралельні обчислення;
- замовні та напівзамовні спеціалізовані НВІС, які апаратно реалізують основні обчислювальні алгоритми.

1.1 Однокристальні програмовані мікропроцесори та мікро-ЕОМ з архітектурою орієнтованою на розв'язання обчислювальних задач

Мікропроцесори мають високий ступінь спеціалізації. В них широко використовують методи скорочення тривалості командного циклу, характерні і для універ- інструкцій, розміщення операндів більшості команд у регістрах, використання тіньових регістрів для зберігання стану обчислень у разі переключення контексту, поділ шин команд і даних (гарвардська архітектура). Водночас для характерною є наявність апаратного перемножувача, що дозволяє

виконувати множення двох чисел за один командний такт. В універсальних мікропроцесорах множення звичайно реалізується за декілька тактів, як послідовність операцій зсуву і додавання. Іншою особливістю є включення в систему команд таких операцій, як множення з підсумовуванням MAC (C = *A* x *B* + *C)* із зазначеним у команді числом виконань у циклі та з правилом зміни індексів використовуваних елементів масивів *A* і *B),* інверсія біта адреси, різноманітні бітові операції. У реалізується апаратна підтримка програмних циклів, кільцевих буферів і вибір з пам'яті за цикл виконання команди декількох операндів.

За обробкою даних розділяють на два класи: з фіксованою та з плаваючою крапкою. Використання з плаваючою крапкою обумовлено декількома причинами. Для багатьох задач, пов'язаних із виконанням інтегральних і диференціальних перетворень, особливе значення має точність обчислень, забезпечити яку дозволяє експоненційний формат представлення даних. Алгоритми компресії, декомпресії, адаптивної фільтрації в пов'язані з визначенням логарифмічних залежностей і дуже чутливі до точності представлення даних у широкому динамічному діапазоні. Робота з даними у форматі з плаваючою крапкою істотно спрощує і прискорює обробку, підвищує надійність програми, оскільки не потребує виконання операцій округлення і нормалізації даних, відслідковування ситуацій втрати значимості і переповнення. Платою за ці додаткові "комфорт і швидкість" є висока складність функціональних пристроїв, що обробляють дані у форматі з плаваючою крапкою, необхідність використання складніших технологій виробництва мікросхем, більший відсоток відбракування виробів і як наслідок - висока ціна мікропроцесорів.

Найпоширеніші виготовляють фірми *Motorola* (56002,96002), *Intel* (i960), *Texas Instruments* (TMS320Cxx), *Analog Devices (21xx,210xx)*. Порівняльний аналіз основних характеристик одного покоління різних фірм показав відсутність суттєвих відмінностей, що пояснюється близькістю архітектури та технології виготовлення. За повнотою сімейства, за наявністю інструментальних

технологічних засобів і розробленого програмного забезпечення із сімейства TMS320 переважають інших фірм.

Принципи проектування та особливості архітектури із сімейства TMS320 є характерними і для всіх інших . Тому архітектуру мікропроцесорів TMS320 можна розглядати як базову. В основу проектування мікропроцесорів TMS320 покладені такі принципи: застосування модифікованої гарвардської архітектури, широке використання конвеєрного режиму роботи, наявність спеціалізованого пристрою множення, існування спеціальних команд , короткий командний цикл.

Для мікропроцесорів серії TMS320 характерне використання апаратної реалізації деяких функцій, які звичайно виконуються програмно. Наприклад, апаратний перемножувач забезпечує множення двох чисел за один командний цикл. Є також апаратні паралельні зсувачі, які здійснюють зсув даних і результатів обчислень, індексні регістри, які забезпечують непряму адресацію даних в оперативному запам'ятовувальному пристрої (ОЗП) та режим автоінкременту (декременту) під час одноциклових маніпуляцій з таблицями даних.

Системи команд сімейства мікропроцесорів TMS320 є комплексними та сумісними знизу доверху. Вони містять інструкції різного функціонального призначення: універсальні - арифметичні, логічні і керуючі команди; спеціальні - призначені для розв'язання задач . Наявність комплексної системи команд розширює сфери застосування і спрощує розробку програм.

- *з фіксованою крапкою.* TMS320C1jc, *C2X,* C5* призначені для обробки чисел у форматі з фіксованою крапкою. Перше покоління мікропроцесорів TMS320C їх побудоване на базовій архітектурі мікропроцесора TMS320C10, структура якого наведена на рис. 1.17. Його адресний простір становить 4К 16-розрядних слів пам'яті програм і 144 16- розрядних слів пам'яті даних. Тривалість командного такту процесора складає 160-200 не. Арифметичні функції в процесорі реалізовані апаратно. Із зовнішніми пристроями процесор взаємодіє через 8,16-розрядних портів введення/виведення. Передбачено можливість обробки зовнішнього переривання. Інші мікропроцесори цього сімейства (CI4 - CI7) мають аналогічну архітектуру і відрізняються тривалістю

командного такту, конфігурацією пам'яті, наявністю (або відсутністю) додаткових периферійних пристроїв.

Мікропроцесори сімейства TMS320C2JC мають підвищену продуктивність і ширші функціональні можливості. Всі мікропроцесори сімейства можуть використовувати пам'яті програм і даних ємністю 64К слів, послідовний порт та шістнадцять паралельних портів введення/ виведення. Мікропроцесори сімейства TMS320C2* мають можливість використання зовнішнього контролера прямого доступу до пам’яті (ПДП). Пристрої множення мікропроцесорів, крім операцій множення, дають змогу виконувати за один такт піднесення до квадрата. У мікропроцесори включена апаратна підтримка кратного виконання команди, реалізований режим двійкової інверсно-непрямої адресації, призначений для ефективної реалізації швидкого перетворення Фур'є.

Мікропроцесори сімейства TMS320C5x, забезпечуючи сумісність за системою команд і наслідуючи загальні архітектурні особливості побудови мікропроцесорів попереднього покоління, відрізняються більшими функціональними можливостями, підвищеною тактовою частотою, меншим енергоспоживанням.

У реалізована апаратна підтримка кільцевих буферів, є можливість одночасного створення в пам'яті даних двох незалежних кільцевих буферів. Існує можливість кратного виконання блока програми. Мікропроцесор містить одинадцять тіньових регістрів, що використовуються для швидкого зберігання/відновлення стану основних регістрів у разі виникнення програмних або апаратних переривань. Паралельний логічний пристрій мікропроцесора дозволяє виконувати бітові і логічні операції над операндами, що утримуються в пам'яті і різних регістрах. Мікропроцесор TMS320C5x може використовувати 244К слів пам'яті, у тому числі:

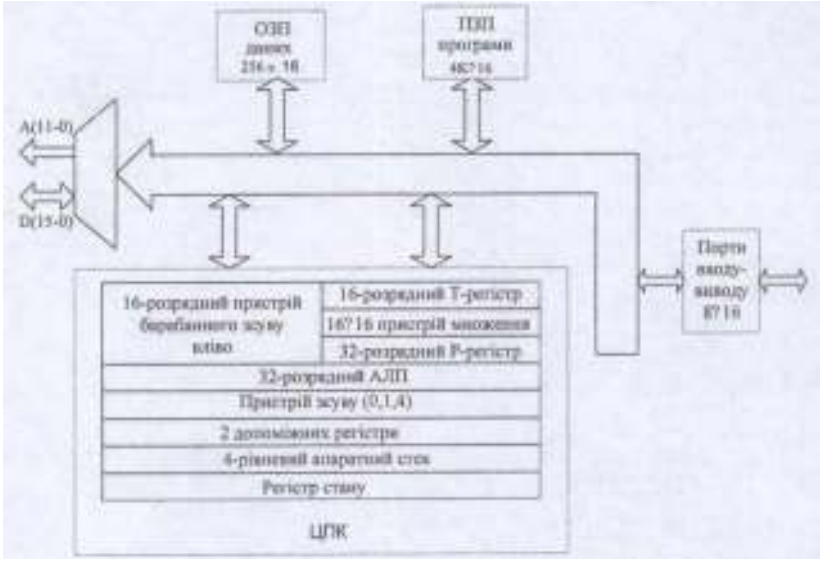


Рис. 1.17. Структура мікропроцесора TMS320C10

64К - пам'ять програм, 64К - пам'ять даних, 64К - 16-розрядні порти вводу/виводу, 32К - глобальна пам'ять. Для можливості роботи з повільною пам'яттю в мікропроцесор включений програмований генератор тактів чекання.

Під час створення мультипроцесорних систем виникає необхідність спільного використання єдиної області пам'яті. Для цього в процесорі передбачені сигнали запиту і готовності при звертанні до глобальної пам'яті, доступ до якої регулює спеціальний арбітр пам'яті. Різниці між мікропроцесорами - представниками сімейства TMS320C5* - полягають, в основному, у конфігурації внутрішньокристальної пам'яті.

Крім 16-розрядних портів введення/виведення, мікропроцесори сімейства мають 2 послідовні порти (у TMS320C52 - один), таймер, інтерфейс тестування і налагодження JTAG.

- ***з плаваючою крапкою.*** Першим представником класу мікропроцесорів із плаваючою крапкою є TMS320C30, він має гнучку систему команд, апаратну підтримку операцій із плаваючою крапкою, потужну систему адресації, розширений адресний простір і підтримку мови високого рівня - CI. Мікропроцесор виготовляють за 0,7 мікронною КМОН технологією з трьома рівнями металізації. Всі операції в мікропроцесорі виконуються за один такт. При тривалості такту 60 не процесор TMS320C30 має швидкодію близько 33 млн. операцій із плаваючою крапкою в секунду. Висока продуктивність мікропроцесора на алгоритмах забезпечується завдяки апаратному виконанню

ряду специфічних функцій, що в інших мікропроцесорах реалізуються програмно або мікропрограмно.

Мікропроцесор має конвеєрну регістро-орієнтовану архітектуру і може паралельно виконувати в одному такті множення і арифметико-логічні операції з числами у форматі з фіксованою або плаваючою крапкою. Структура мікропроцесора TMS320C30 наведена на рис. 1.18.

До складу мікропроцесора TMS320C30 входять: 32-розрядна шина команд і даних; 24-розрядна шина адреси; 2 блоки ОЗП; 32-розрядний перемножувач із плаваючою крапкою; кеш- пам'ять команд ємністю 64 слова; 8 регістрів для операцій із підвищеною точністю; два генератори адреси; регістровий файл; 40-розрядний АЛП, який працює і з цілими числами, і з числами у форматі з плаваючою крапкою. Вмонтований контролер ПДП дозволяє поєднувати в часі виконання обмінів даними з пам'яттю і обчислення.

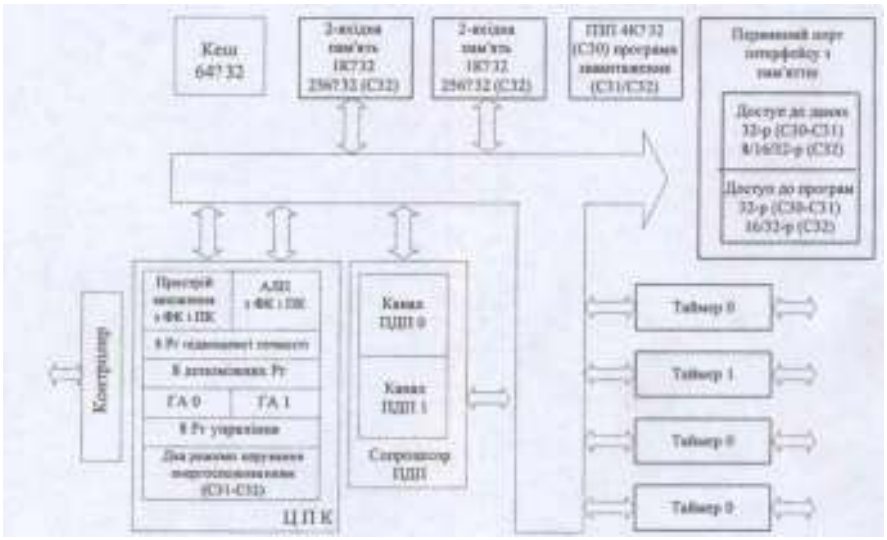


Рис. 1.18. Структура мікропроцесора TMS320C30

Наявність у TMS320C30 мультипроцесорного інтерфейсу, двох зовнішніх інтерфейсних портів, двох послідовних портів, розширеної системи переривань спрощує конструювання систем на його основі. Завдяки своїй високій продуктивності і простоті використання в обчислювальних системах TMS320C30 може застосовуватися і як головний процесор, і спеціалізований співпроцесор.

Наступними представниками із плаваючою точкою є мікропроцесори сімейства TMS320C4x. Завдяки своїй унікальній архітектурі мікропроцесори TMS320C40 набули поширення в мультипроцесорних системах і витіснили сімейство трансп’ютерів, яке раніше панувало в цій технологічній ніші.

Процесори TMS320C4x сумісні за системою команд із TMS320C3*, водночас мають більшу продуктивність і кращі комунікаційні можливості.

У сімейство TMS320C4x входять процесори TMS320C40, TMS320C44, TMS320LC40. TMS320C40 має продуктивність 3OMIPS/6OMFLOPS і максимальну пропускну спроможність підсистеми введення/виведення 384 Мбайт/с.

TMS320C40 містить на кристалі шість високошвидкісних (20 Мбайт/с) комунікаційних портів і шість каналів ПДП, 2К слів пам'яті, 128 слів програмного кеша. Дві зовнішні шини забезпечують доступ до 4Г слів.

Висока продуктивність і хороші комунікаційні властивості процесора дозволяють ефективно використовувати його для розв'язання задач обробки зображень, моделювання віртуальної реальності, розпізнавання мови, моніторингу.

- *Мультипроцесорні* . Подальшим розвитком сімейства *Texas Instruments* є мікропроцесори принципово нової архітектури - TMS320C8x. Мікропроцесори орієнтовані на ужитки, пов'язані з високопродуктивною цифровою обробкою сигналу в різноманітних галузях науки і техніки. Друга назва процесорів - MVP (*Multimedia Video Processor*) - характеризує їх високу ефективність у задачах обробки зображень, 2- і 3-вимірній графіці, у системах віртуальної реальності, компресії і декомпресії відео- і аудіоданих та ін.

Базовим сімейства TMS320C8x є TMS320C80 (рис 1.19), який об'єднує в одній мікросхемі п'ять повнофункціональних процесорів, чотири з яких - поліпшені процесори *(Advanced Digital Signal Processor).*

Кожний із ADSP дає змогу виконати за один командний такт декілька RISC-подібних операцій. П'ятий процесор, головний *{Master Processor* (MP)), являє собою 32-розрядний RISC- процесор із високопродуктивним обчислювачем із плаваючою крапкою, сумісним із стандартом IEEE 754.

Крім процесорного ядра на кристалі розміщені: контролер обміну (*Transfer Controller* (TC)) - інтелектуальний контролер ПДП, що підтримує інтерфейс із DRAM і SRAM, відеоконтролер (*Video Controller* (VC)), система контролю і налагодження - порт JTAG (IEEE 1149.1), 50Кб SRAM. Випускається також

спрощений варіант мікропроцесора TMS320C82, що відрізняється меншого об’ємом пам’яті, кількістю сигнальних процесорів ADSP (2), відсутністю відео-контролера і відповідно меншою вартістю.

Сумарна продуктивність TMS320C80 на регістрових операціях сягає 2 млрд. RISC- подібних команд у секунду. Завдяки високій продуктивності TMS320C80 може замінити при реалізації ряду ужитків більш ніж 10 високопродуктивних або мікропроцесорів загального призначення. Пропускна здатність шини TMS320C80 сягає 2,4 Гбайт/с - у потоці даних і 1,8 Гбайт/с - у потоці інструкцій.

TMS320C80 забезпечує високий ступінь гнучкості й адаптивності системи, побудованої на його базі, що досягається за рахунок наявності на кристалі процесорів які паралельно функціонують і головного RISC-процесора. Архітектура процесора TMS320C80 належить до класу MIMD-множинний потік даних, множинний потік команд. Процесори, що входять до складу TMS320C80, програмуються незалежно один від одного і можуть виконувати як різні, так і одну спільну задачу. Обмін даними між процесорами здійснюється через спільну внутрішньокристальну пам'ять. Доступ до розподіленої внутрішньокристальної пам'яті забезпечує матричний комутатор ***(Crossbar),*** що виконує також функції монітора у разі звертання декількох процесорів до одного сегмента пам'яті.

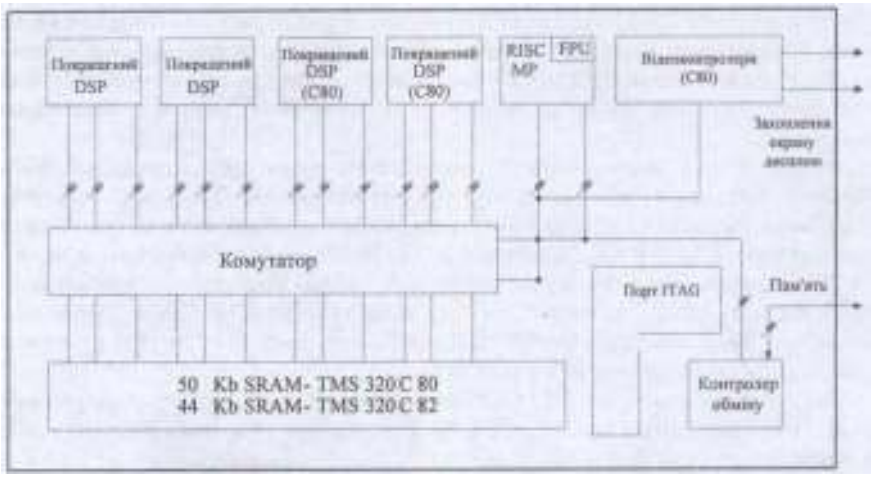


Рис. 1.19. Структура мікропроцесора TMS320C80

1.2 Трансп'ютери та НВІС однорідних обчислювальних середовищ, які орієнтовані на масово-паралельні обчислення

Трансп'ютери належать до класу RISC-процесорів. До сімейства трансп'ютерних НВІС, входять декілька НВІС трансп'ютерів, а також зовнішніх запам'ятовуючих пристроїв (ЗП), призначених для підключення до трансп'ютерів. Основними типами НВІС трансп'ютерів є: трансп'ютер зі статичним ОЗП; трансп'ютер-зв'язний адаптер; трансп'ютер-контролер графічних пристроїв, трансп'ютер-контролер масової пам'яті. Кожний з цих типів може містити декілька НВІС, що різняться типом процесора, типом і конфігурацією внутрішнього ЗП, швидкодією та іншими параметрами.

Структура трансп'ютера IMS T424 є характерною для більшості трансп'ютерів (рис. 1.20). Основними елементами такої структури є 32-розрядний процесор, внутрішнє ОЗП ємністю 4К байт, блок системних пристроїв, чотири блоки інтерфейсів каналів зв'язку, блок інтерфейсу зовнішнього ЗП і блок інтерфейсу периферійних пристроїв.

Трансп'ютер можна використовувати як в разі автономного однокристального обчислювального пристрою (мікро-ЕОМ), так і процесорного модуля (ПМ) для побудови високопродуктивних паралельних обчислювальних систем. Використовуючи трансп'ютер як однокрис- тальну мікро-ЕОМ, можна застосовувати стандартні алгоритмічні мови високого рівня. Але трансп'ютер спеціально спроектовано так, щоб забезпечити ефективну реалізацію конкретної, спеціально розробленої для нього мови паралельного програмування ***Occam.*** Ця мова дає змогу в явному вигляді програмувати функціонування паралельних систем. З одного боку, вона як асемблер забезпечує ефективне використання пам'яті і високу продуктивність при виконанні програм, а з іншого боку, подібно мовам високого рівня, - велику ефективність при розробці програмного забезпечення і його високу надійність. Мова Occa***m*** є основою методології проек- тування паралельних систем на базі трансп'ютерів подібно тому, як булева алгебра становить основу методології проектування сучасних електронних систем на базі логічних елементів.

Завдання розробки трансп'ютерних систем полегшується завдяки архітектурній відповідності між мовою Occam і трансп'ютером. Програма, яку виконує одним трансп'ютер, формально еквівалентна одному процесу мови

Occam. З цього випливає, що систему, яка складається з деякої сукупності трансп'ютерів, можна безпосередньо описати за допомогою програми на даній мові. Основне поняття, яке використовується в мові Occam, - це поняття процесу. Кожний процес розглядається як деякий об'єкт (чорна скринька), що характеризується своїм внутрішнім станом. Процес може змінювати свій внутрішній стан і взаємодіяти з іншими процесами шляхом обміну повідомленнями, використовуючи для цього двоточкові канали зв'язку. Під каналом розуміють мовну конструкцію, що забезпечує можливість організації зв'язку між процесами, а не фізичний канал ***(link),*** що використовується для зв'язку між процесорними модулями.

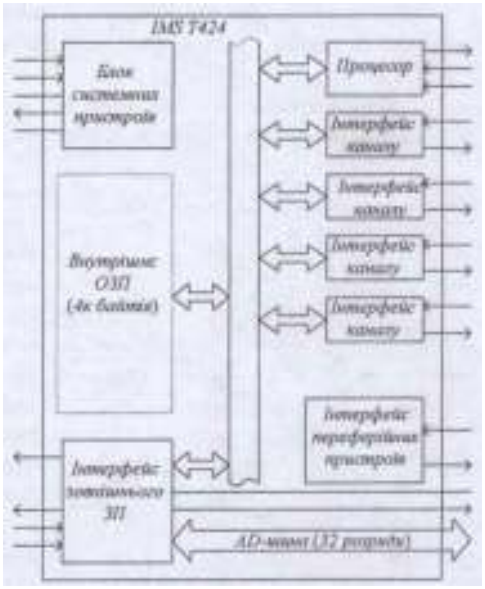


Рис. 1.20. Структура трансп'ютера IMS T424

Повідомлення, які відсилаються процесом, залежать від його внутрішнього стану. Цей стан, у свою чергу, змінюється при прийомі даним процесом повідомлень від інших процесів. Таке поняття процесу можна використовувати для опису поводження найрізноманітніших об'єктів, наприклад логічного елементу, мікропроцесора, верстата, закладу і так далі. Процеси є скінченні, тобто кожен процес починає своє виконання, виконує ряд дій і завершується. Кожна дія може являти собою або множину послідовних процесів, що виконуються один за одним, або множину паралельних процесів, що виконуються одночасно, або множину альтернативних процесів, тобто таких, з множини яких виконується тільки один, що є вибраний відповідно до заданого критерію. Кожен процес може складатися з інших процесів, причому деякі з цих

процесів можуть бути паралельними, тому процес може мати будь-який рівень внутрішньої паралельності і цей рівень може змінюватися залежно від моментів початку та завершення процесів.

Усі процеси складаються з таких примітивних (основних) процесів: присвоєння, введення, виведення, пропуску, зупинки. Процес присвоєння обчислює значення деякого виразу і присвоює це значення деякій змінній. Процеси введення і виводу використовуються для організації взаємодії між більш складними процесами.

Два паралельні процеси можуть підтримувати зв’язок один з одним використовуючи одно- напрямлений канал, який зв'язує два дані процеси. Під час виконання такого зв'язку один процес надсилає (виводить) повідомлення в канал, а інший процес приймає (вводить) це повідомлення з каналу. Ключова концепція мови ***Occam*** полягає в тому, що цей зв'язок є синхронним. Він виконується тільки в тому випадку, коли перший процес готовий виконати виведення, а другий процес - введення. Якщо один процес готовий до виконання відповідної дії раніше ніж другий, то він повинен чекати готовності другого процесу. Таким чином, спосіб взаємодії між процесами подібний до методу "рукопотиснення", що використовується для організації зв'язку в апаратних засобах. Якщо процес має внутрішню паралельність, то він може використовувати для зв'язку з іншими процесами одночасно декілька каналів, оскільки наявні в ньому паралельні процеси можуть одночасно виконувати введення і виведення для декількох каналів. Мова ***Occam*** є для трансп'ютера мовою того ж рівня, що і ассемблер. Трансп'ютер має спеціальні засоби для реалізації концепції паралельності процесів, що є прийняті в даній мові. Цю мову можна використовувати для написання програм, приз- начен их як для окремого трансп'ютера, так і для систем, що складаються із декількох трансп'ютерів, тобто мультитрансп'ютерних систем.

Однорідне обчислювальне середовище (ООС) (рис. 1.21) - це двовимірна регулярна матриця процесорних елементів (ПЕ), кожен з яких фізично зв'язаний за входом-виходом з чотирма сусідами - зверху, знизу, зліва та справа (комутація). Кожний ПЕ може виконувати набір бітових операцій перетворення

інформації з вхідних каналів у вихідні. ООС е універсальною системою, тобто в ньому можна реалізувати довільну обчислювальну функцію. Бітовий рівень ПЕ та повна система комутації дозволять реалізувати паралелізм та конвеєрність обчислень на найнижчому бітовому рівні. Це є істотною перевагою ООС при реалізації програмно налаштованих спеціалізованих паралельних комп'ютерних систем для задач ЦОС. В багатьох випадках при реалізації обчислень на ООС вимагається організація взаємодії такої системи з однорідним запам'ятовуючим середовищем, яке є матрицею регістрів зсуву, довжина котрих програмується. Промисловість випускає ООС з 20-92 ПЕ на кристалі з продуктивністю одного ГІЕ близько 10 млн. однобітових операцій в секунду

1.3 Замовні та напівзамовні спеціалізовані НВІС, які апаратно реалізують основні обчислювальні алгоритми

За способом проектування і виготовленням, тобто налаштуванням на виконання конкретного алгоритму, спеціалізовані НВІС поділяються на два класи: замовні і напівзамовні (рис. 1.22).

Замовні НВІС - це мікросхеми, розроблені на основі стандартних або спеціально створених елементів і вузлів за схемою замовника. Всі топологічні шари замовних НВІС проектують

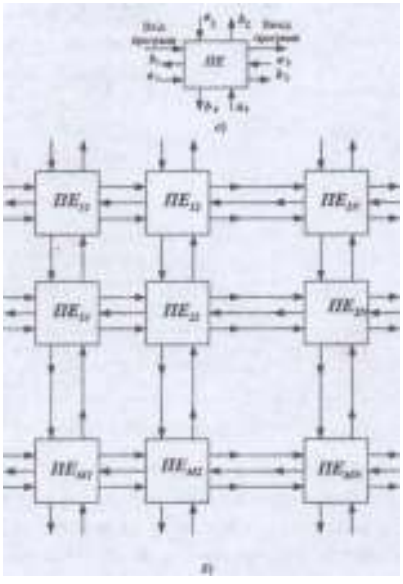


Рис. 1.21.Структури: а) процесорного елемента, б) однорідне обчислювальне середовище

і виготовляють індивідуально. Існують повністю замовні (ПЗ) НВІС, схеми яких оптимізовані на рівні окремих компонентів, та НВІС, побудовані на основі стандартних елементів (СЕ), які вибирають із раніше спроектованої і перевіреної бібліотеки елементів. До складу бібліотеки можуть входити прості логічні елементи типу І-НЕ, АБО-НЕ, тригери, а також складніші типу суматорів, регістрів, комутаторів та інші. Особливістю замовних НВІС є оптимізація елементів і зв'язків на реалізацію алгоритмів, що дозволяє досягнути граничних значень параметрів для кожного рівня технології.

Напівзамовні НВІС - це мікросхеми, що складаються з двох частин: наперед спроектованої постійної та змінної - замовної, структура якої визначає замовник. До напівзамовних НВІС належать мікросхеми на основі базових кристалів (БК) та програмовані користувачем логічні інтегральні схеми (ПЛІС).

Проектування пристроїв на БК здійснюється за рахунок нанесення відповідних шарів з'єднань. Основними елементами БК є базові комірки, що складаються з набору некомутованих елементів-транзис- торів і резисторів. На базі таких елементів реалізуються функціонально завершені вузли, які виконують елементарні функції.

Порівняно з БК технологія ПЛІС забезпечує рекордно малий проектно-технологічний цикл (від декількох годин до декількох днів), мінімальні витрати на проектування, максимальну гнучкість у разі необхідності модифікації апаратури.

На світовому ринку можна виділити декілька компаній-виробників - ***Х\ШС-ХИШХ, ALTERA, LATTICE, AT&T, INTEL,*** які виготовляють мікросхеми з архітектурою EPLD (***EPROM technology bazed complex Programmable Logic Device) -*** з можливістю багаторазового пере- програмування, і FPGA (***Field Programmable Gate Array) -*** з можливістю багаторазового реконфігуру вання.

Як пам'ять для зберігання конфігурації в ПЛІС EPLD використовується ППЗП з ультрафіолетовим стиранням, а у ПЛІС FPGA - статичний ОЗП.

Мікросхема FPGA являє собою матрицю логічних комірок, що з'єднані між собою логічними ключами. Статична пам'ять, яка є в мікросхемах FPGA, при

заповненні деякою бітовою послідовністю діє на логічні комірки і з'єднує їх через ключі, що дозволяє отримати необхідні електричні схеми (регістри, лічильники, логічні схеми та ін., що з'єднані в необхідній послідовності).

Кожна мікросхема FPGA має вхід для запису бітової послідовності, яка заповнює статичну пам'ять, а також елементи "вхід/вихід" для зв'язку з іншими мікросхемами. Таким чином, на основі однієї чи декількох мікросхем FPGA можна створювати реконфігурований процесор з перевагами спеціалізованого процесора на "жорсткій" логіці, але з можливістю зміни вмісту статичної пам'яті.

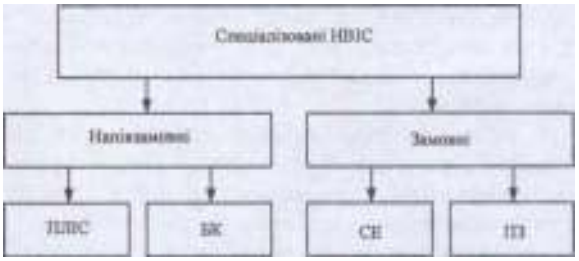


Рис. 1.22. Класифікація спеціалізованих НВІС

Відмінною особливістю ПЛІС архітектури FPGA різновидів *XILINXX* С3000, ХС3100, ХС4000 є наявність поля логічних блоків і блоків введення/виведення, які зв'язані між собою через комутаційні блоки. Логічні блоки, блоки введення/виведення і комутаційні поля конфігуруються при завантаженні в ПЛІС бітової послідовності, що отримана в результаті розробки схеми.

Залежно від різновиду ПЛІС логічні блоки, блоки введення/виведення, комутаційні блоки мають різний ступінь складності і різні функціональні можливості.

Логічний блок - один із базових елементів архітектури ПЛІС FPGA, може виконувати будь-яку логічну функцію відповідно до заданої бітової послідовності. Завантаженням іншої бітової послідовності можна змінювати виконувану функцію необмежену кількість разів. Блок введення/виведення, так само як і логічний блок, може бути налаштований на виконання будь-якого електричного з'єднання реалізованої всередині ПЛІС схеми з зовнішніми пристроями через відповідний контакт мікросхеми.

2. Лекція «Елементна база комп'ютерних систем на ПЛІС.»

1. Історія появи мікросхем програмованих цифрових пристроїв. Попередники ПЛІС.

Перші програмовані логічні пристрої створювалися на основі технології біполярних програмованих ПЗП з додатковими логічними можливостями і властивостями. Фірма Signetics випустила в 1972 році біполярну мікросхему програмованої логічної матриці. Вдосконалення архітектури привело до створення фірмою Monolithic Memories Inc. (MMI) у 1975-1976 роках мікросхем програмованої матричної логіки (PAL), що вмонтовуються в 20- і 24-вивідні корпуси і здатних замінити до 20 логічних вентилів, що були у продажу. У 1984-му фірма Altera випустила першу мікросхему CPLD що містить 300 вентилів. І вже сьогодні логічна ємкість мікросхем ПЛІС з конфігураційною флэш-памятью перевищує 1 млн вентилів. До класу програмованих логічних приладів відносяться прості і складні ПЛІС (SPLD і CPLD, відповідно), а також програмовані користувачем базові матричні кристали (FPGA).

Програмовані логічні інтегральні схеми (ПЛІС) дозволяють в стислі терміни створювати високошвидкісні периферійні модулі, шинні інтерфейси (PCI, USB), мережеві пристрої, контроллери і ін.

ПЛІС - це цифрова інтегральна схема з програмованою структурою.

1. одноразово-програмні (з плавкими перемичками, з пробиваними перемичками)
2. репрограмміруємиє (з ультрафіолетовим стиранням УПФ ПЗП, з електричним стиранням Flash, ЕСП ПЗП)

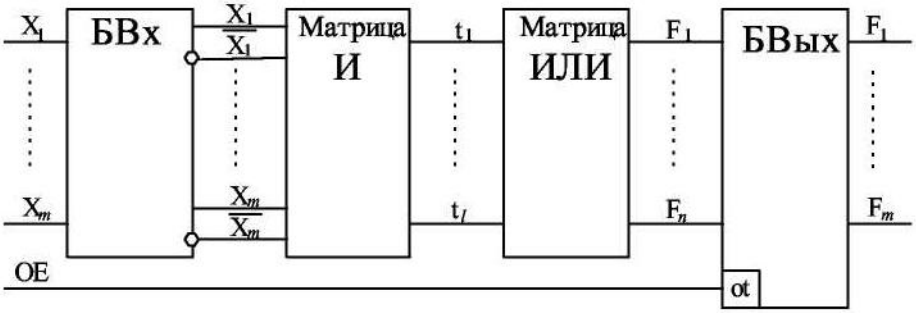
Види ПЛІС:

1. програмовані логічні матриці (PLA)
2. програмована матрична логіка (PAL)
3. базові матричні кристали (GA)

2. Мікросхеми типу програмована логічна матриця. Їх основні параметри. Спрощена схема.

Програмовані логічні матриці (ПЛМ) з'явилися до восьмидесятих років. Основою служать послідовність програмованих матриць елементів «І» і «АБО», а також блоки вхідних і вихідних буферних каскадів (Бвх і Бвих).

Структура ПЛМ:



Основні параметри ПЛМ:

- Число входів t;
- Число термів 1;
- Число виходів n;
- матриця І - кон'юнктори;
- матриця АБО - діз'юнктори.

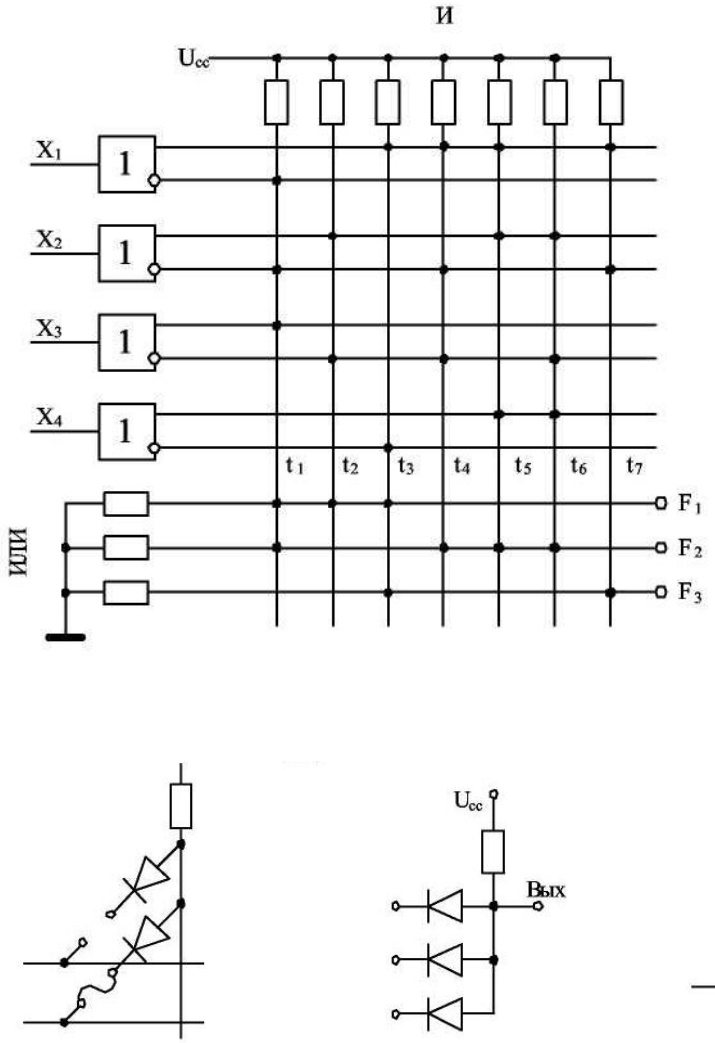
Число формованих термів рівне числу коньюнкторов (И). Число дізьюнкторов (АБО) рівне числу функцій n, що виробляються.

ПЛМ реалізує диз'юнктивну нормальну форму (ДНФ) відтворних функцій (дворівневу логіку).

Які саме терми будуть вироблені і які комбінації цих термів складуть вихідні функції, визначається програмуванням ПЛМ.

Схемотехніка ПЛМ

Випускаються ПЛМ як на основі біполярної технології, так і на МОП-транзисторах. У матрицях є системи горизонтальних і вертикальних зв'язків, у вузлах, перетини яких при програмуванні створюються або ліквідовуються вузли зв'язку.



Функції, що реалізовуються, можна ускладнювати, вводячи наприклад один з виходів на вхід. Тоді можна реалізувати складнішу, скобову форму функції.

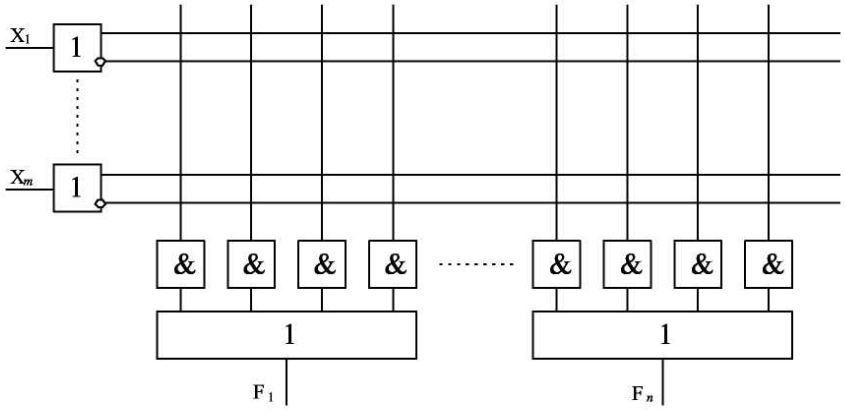
3. Мікросхеми типу програмована матрична логіка. Їх основні параметри.

Програмована матрична логіка (ПМЛ)

SPLD(Simple Programmable Logic Devices), тобто прості прогамуємі логічні пристрої. По архітектурі ці ПЛІС діляться на підкласи програмованих логічних матриць ПЛМ (PLA, Programmable Logic Arrays) і програмованої матричної логіки ПМЛ (PAL. Programmable Arrays Logic, або GAL, Generic Array Logic).

Обидва ці підкласу мікросхем реалізують диз'юнктивні нормальні форми (ДНФ) функцій перемикачів, а їх основними блоками є дві матриці: матриця елементів І і матриця елементів АБО, включені послідовно. Така структурна модель ПЛМ і

ПМЛ. Технічно вони можуть бути виконані і як послідовність двох матриць елементів АБО-НІ.



У ПМЛ вироблені матрицею І терми поступають на фіксовану (непрограмовану) матрицю елементів АБО. Це означає жорсткий заздалегідь заданий розподіл наявних термів між окремими діз'юнкторамі. Кожним діз'юнктору додаються свої власні терми, і якщо для разних діз'юнкторов виявляться потрібними однакові терми, доведеться виробляти їх в матриці І кілька разів. Проте при цьому программіруємость матриці АБО виключається, що для багатьох завдань у результаті істотно спрощує схему ПМЛ порівняно з схемою ПЛМ.

Тут виходи елементів матриці «І» жорстко розподілені між елементами матриці «АБО».

У схемі ПМЛ m входів, n виходів і 4n елементів «І», оскільки кожному елементу «АБО» додається по 4 коньюнктора. Порівняно з ПЛМ ці ПЛІС мають меншу функціональну гнучкість, але їх виготовлення і використання простіші. Переваги виявляються при проектуванні нескладних пристроїв.

4. Функціональні різновиди ПЛМ і ПМЛ.

SPLD, тобто прості програмовані пристрої, по архітектурі діляться на підкласи програмованих логічних матриць ПЛМ (Programmable Logic Arrays) і програмованої матричної логіки ПМЛ (PAL, Programmable Arrays Logic).

Обидва ці підкласу мікросхем реалізують диз'юнктивні нормальні форми (ДНФ) функцій перемикачів, а їх основними блоками є дві матриці: матриця елементів І

і матриця елементів АБО, включені послідовно. Технічно вони можуть бути виконані і як послідовність двох матриць елементів ІЛІ-НЕ.

Матриці АБО для ПЛМ і ПМЛ різні. У ПЛМ матриця програмується, а в ПМЛ вона фіксована.

Програмована матриця АБО мікросхем ПЛМ складена з АБО, що мають по q входів. На входи кожного АБО при програмуванні можна подати будь-яку комбінацію наявних термів, причому терми можна використовувати багато разів.

ПЛМ дозволяє реалізувати систему з n перемиканих функцій, залежних не більше ніж від m змінних і таких, що містять не більше ніж q термів. Тому функціональні можливості ПЛМ характеризуються трьома цифрами: n, q, m.

У ПМЛ вироблені матрицею І терми поступають на фіксовану (непрограмовану) матрицю елементів АБО. Це означає жорсткий заздалегідь заданий розподіл наявних термів між окремими АБО. Кожному АБО додаються власні терми, і якщо для разних АБО виявляться потрібними однакові терми, доведеться їх виробляти в матриці І кілька разів. Проте при цьому программіруємость матриці АБО виключається.

ПЛМ володіють більшою функціональною гнучкістю, всі відтворені ними функції можуть бути комбінаціями будь-якого числа термів, що формуються матрицею І. Ето корисно при реалізації систем функцій перемикачів, що мають великі взаємні перетини по термах (наприклад, завдання формування сигналів управління машинними циклами процесорів). ПМЛ поширені більше, ніж ПЛМ, і до їх числа відносяться більшість SPLD.

У складних програмованих логічних схемах CPLD (Complex Programmable Logic Devices) декілька блоків, подібних ПМЛ, об'єднуються засобами програмованої комутаційної матриці. У CPLD можуть входити сотні блоків і тисячі еквівалентних вентилів. Впливаючи на програмовані з'єднання комутаційної матриці і ПМЛ, що входять до складу CPLD, можна реалізувати необхідну схему.

5. Базові матричні кристали. Їх характеристики. Напівзамовні і замовлені ІС.

БМК - сукупність розташованих на кристалі базових осередків, між якими є вільні зони для створення з'єднань, які називаються канали.

Причому БЯ полягає: матриця Бя1, ОЗУ, ПЗП і матриці Бя2.

Напівзамовні інтегральні схеми на основі базових матричних кристалів (БМК) для розробників і виробників складної електронної апаратури є незамінними:

коли потрібно швидко розробити і почати виробництво виробу;

коли об'єм виробництва виробу відносно невисокий, а відповідних БІС серед тих, що випускаються немає;

при створенні специфічної апаратури з оригінальною схемотехнікою;

при переробці раніше створеної апаратури на нову елементну базу;

за бажання замовника самостійно розробити БІС.

Під замовленою розуміється мікросхема, проектування і виробництво якої відбувається по індивідуальному повному технологічному циклу, направленому на створення саме цій БІС. До замовлених відносяться мікропроцесори, мікроконтролери, периферійні і маса інших БІС, що поставляються як стандартні вироби.

Складні програмовані логічні пристрої. Узагальнена структура ПЛІС типу CPLD.

CPLD - мікросхеми високого рівня інтеграції, основними частинами яких є:

- PAL(GAL) - подібні функціональні блоки
- система комутації, що дозволяє об'єднати функціональні блоки в єдиний пристрій
- блоки введення-виводу (БВВ)

CPLD за рахунок PAL-подобной структури ідеально підходять для реалізації складних лічильників, високошвидкісних дешифраторів адресних шин великої розрядності, комутації-мультиплексування потоків даних і побудови різноманітних кінцевих автоматів. Вони застосовуються як периферія до стандартних мікроконтролерів і мікропроцесорів.

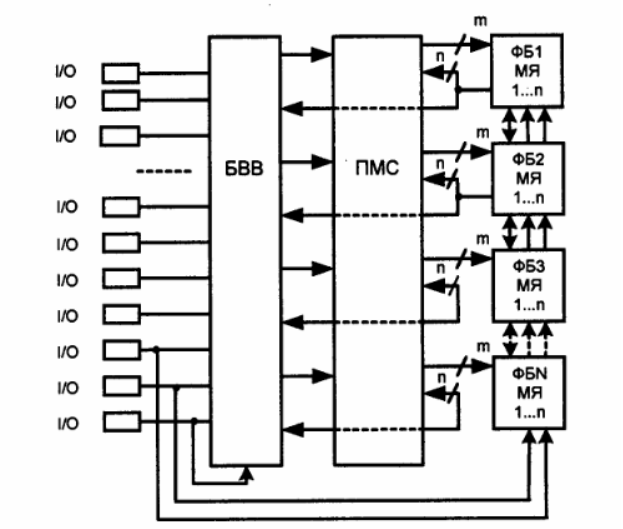


Рис. 1.4. Обобщенная структура CPLD

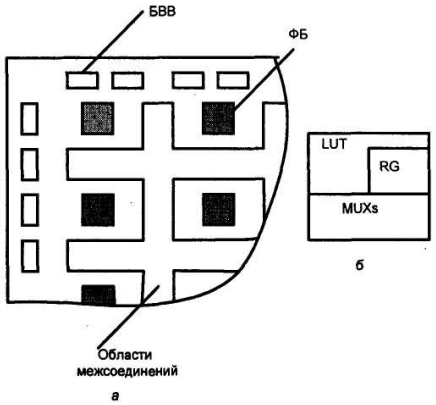
Всі складові частини CPLD програмуються. Узагальнена структура CPLD показана на малюнку.

Через ФБ (FB) позначені функціональні блоки, число яких N залежить від рівня інтеграції мікросхеми. У кожному ФБ є п макроосередків МЯ. Функціональні блоки отримують вхідні сигнали від програмованої матриці з'єднань ПМС. Число таких сигналів т. Вихідні сигнали ФБ поступають як в ПМС, так і в блоки введення/виводу CPLD (БВВ). ПМС забезпечує повну коммутіруємость функціональних блоків, тобто можливість подавати сигнали з будь-якого їх виходу на будь-який вхід. Блоки введення/виводу пов'язані із зовнішніми двонаправленими виводами I/o, які, залежно від програмування, можуть бути використані як входи або як виходи. Три нижні виводи або спеціалізуються для подачі на матрицю функціональних блоків сигналів GCK (Global Clocks) того, що глобального тактує, сигналів GSR (Global Set/reset) глобальної установки/скидання і сигналів GTS (Global 3-state Control) глобального управління третім станом вихідних буферів, або ці ж виводи можуть бути використані для операцій введення/виводу.

18. Програмовані користувачем вентильні матриці - ПЛІС типу FPGA.

У найбільш типовому варіанті Fpga є мікросхемою високого рівня інтеграції, матрицю ідентичних функціональних блоків, що містить у внутрішній області, і систему їх межсоєдіненій, розміщену між рядками і стовпцями матриці, а в периферійній області - блоки введення/виводу (мал. 18.а). Окрім цього варіанту

існують Fpga, в яких функціональні блоки розташовані по рядках (строкові Fpga). Всі частини Fpga (функціональні блоки ФБ, система межсоєдіненій і блоки введення/виводу БВВ) є такими, що конфігуруються або реконфігурованими, причому (на відміну від БМК) засобами самих користувачів.



Перераховані частини - основа Fpga. Окрім них сучасні варіанти Fpga, як правило, оснащені додатковими засобами для автопідстроювання затримок в системі того, що тактує (Pll, Phase Locked Loop або DLL, De1ay Locked Loop), засобами підтримки інтерфейсу Jtag і ін. При конфігурації Fpga функціональні блоки настроюються на виконання необхідних операцій перетворення даних, а система межсоєдіненій - на необхідні зв'язки між функціональними блоками. В результаті у внутрішній області Fpga реалізується схема потрібної конфігурації. Розташовані по краях кристала блоки введення/виводу забезпечують інтерфейс Fpga із зовнішнім середовищем. Блоки введення/виводу сучасних Fpga можна програмувати на виконання вимог безлічі стандартів передачі даних. На мал. 18, би укрупнено показаний склад типового функціонального блоку ФБ, в який входять функціональний перетворювач ФП, реалізований у вигляді програмованого пристрою (LUT, Look-up Table), що запам'ятовує, трігер (регістр) і мультиплексори, що грають роль засобів конфігурації ФБ. LUT - найбільш поширений різновид ФП в FPGA із статичною пам'яттю конфігурації. У схемах FPGA з одноразовим програмуванням перемичок знаходять застосування ФП у вигляді простих логічних вентилів (Slc, Simple Logic Cell) і логічних модулів на основі мультиплексорів.

22. ПЛІС з комбінованою архітектурою.

Структура мікросхем сімейства Flex

З'явилася архітектура поєднуючі переваги Cpld і Fpga (сімейство Flex (F1exble Logic E1ement matrix) фірми Altera). На мал. 22 приведена структура мікросхем сімейства Flex.

Мікросхеми сімейства Flex мають функціональні блоки (Labs, Logic Аггау Blocks) з логічними елементами ЛЕ (Les, Logic E1ements), що містять функціональні перетворювачі ФП табличного типу (Luts). Функціональні блоки розташовані у вигляді матриці, між їх рядками і стовпцями проходять горизонтальні і вертикальні канали трасувань, що характерний для Fpga. В той же час, траси в каналах не сегментовані, а безперервні, що типово для Cpld. Система комутації має два рівні межсоєдіненій - глобальний і локальний. Локальна програмована матриця з'єднань (локальна ПМС або ЛПМС) забезпечує межсоєдіненія логічних елементів ЛЕ, з яких складаються функціональні блоки Labs. До складу Lab входять 8 логічних елементів. З'єднання між блоками Lab забезпечуються глобальною програмованою матрицею з'єднань ГПМС, до кінців рядків і стовпців якої підключаються блоки введення/виводу (Iobs, Input/output Blocks).

У складі багатьох мікросхем ПЛІС є вбудовані блоки пам'яті ВБП (Eabs, Embedded Аггау B1oceks). Такий блок може конфігуруватися як ЗУ і використовуватися не тільки для зберігання даних, але і як табличний ФП для реалізації складних функцій з числом аргументів 8-10.

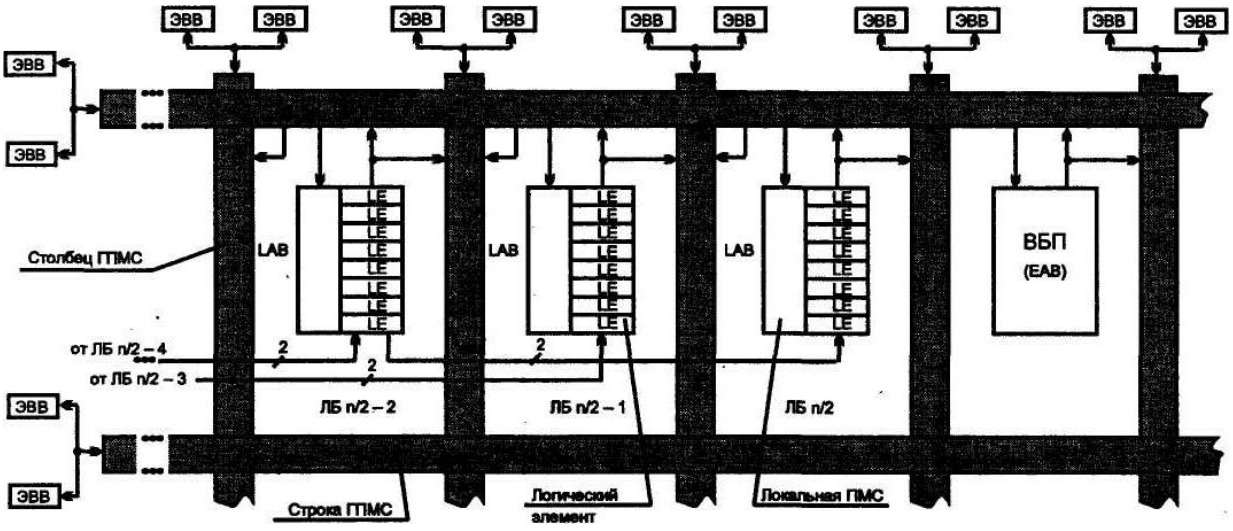


Рис.22 Структура мікросхеми сімейства FLEX

9. Сучасні ПЛІС. Їх різновиди, основні застосування

Сучасні зразки ПЛІС, виконані за 0,22-мікронною технологією, здатні працювати на частотах до 300 Мгц і реалізують до 3 млн. еквівалентних логічних вентилів. По формуванню структури сучасні ПЛІС підрозділяються на дві групи.

До першої відносяться пристрої, в яких необхідна структура пристрою створюється програмуванням зв'язків комутуючих матриць з використанням технологій перепрограмованих постійних пристроїв, що запам'ятовують, у тому числі і з електричним стиранням. Такі пристрої називаються багато разів програмованими ПЛІС або EPLD (EPROM) або CPLD. Особливість цих пристроїв полягає в тому, що сформована структура є незалежною, тобто, зберігається при виключенні живлення, а для зміни структури необхідно виконати операції стирання (очищення) EPROM і програмування (записи) нової структури.

У пристроях другої групи необхідні зв'язки комутуючих матриць забезпечуються логічними ключами, які управляються бітовою послідовністю, записуваною у внутрішнє статичне ОЗУ при конфігурації ПЛІС, тому пристрої цього класу отримали назву багатократно реконфігурованих або FPGA. Особливістю пристроїв цього класу є те, що необхідна структура цільового пристрою повинна відновлюватися (записуватися у внутрішнє статичне ОЗУ) після кожного включення живлення, що вимагає вживання додаткових заходів по збереженню і відновленню необхідної конфігурації - є їх перевагою, оскільки дозволяє створювати адаптивні системи із структурою, що динамічно змінюється в часі. Тобто, в різні моменти часу використовувати один і той же кристал для реалізації різних цифрових пристроїв, які найкращим чином відповідають зовнішнім умовам, що змінюються в часі (наприклад, забезпечувати роботу стільникового телефону в мережах з різними стандартами залежно від доступності тієї або іншої мережі в даний момент часу).

Обидва класи ПЛІС дозволяють реалізовувати будь-які цифрові схеми, проте, через ряд особливостей внутрішньої структури ПЛІС першої групи більш

пристосовані до реалізації складних комбінаційних схем, а ПЛІС другої - до реалізації цифрових (кінцевих) автоматів (state machine).

3. Лекція «Особливості структурної організації пам'яті комп'ютерних систем на ПЛІС.»

Багатомодульна пам'ять. Багатомодульна структура пам'яті з декількома шинами адреси і даних характерна для архітектури більшості процесорів обробки сигналів і зображень. Основною перевагою таких процесорів є можливість виконання двох і більше звертань до пам'яті протягом одного циклу виконання команди та забезпечення обміну інформацією між пам'яттю даних і пам'яттю команд. Прикладом двомодульної організації пам'яті є модифікована гарвардська архітектура, у якій пам'ять розділена на два незалежні блоки (пам'ять програм і пам'ять даних). Модифікована гарвардська архітектура з двома модулями пам'яті використовується у мікропроцесорах фірм *Texas Instruments* (TMS320Clx), *Analog Devices* (ADSP-21xx) і AT&T (DSP16xx). Модифіковану гарвардську архітектуру з тримодульною пам'яттю використовують у швидкодіючих мікропроцесорах фірм *Zilog* (Z893xx), *Motorola* (DSP560xx, 563xx і 96008), *Texas Instruments* (TMS320C2x-TMS320C5x).

Особливістю мікропроцесорів з двома і трьома модулями пам'яті на кристалі є порівняно мала ємність їхньої пам'яті. Розширення ємності пам'яті мікропроцесорів досягається за рахунок зовнішньої пам'яті. Більшість мікропроцесорів для підключення зовнішньої пам'яті використовують одну множину шин (адреси, даних і управління) поза кристалом. Така зовнішня пам'ять дозволяє лише одне звертання протягом одного командного циклу, що зменшує швидкодію мікропроцесора при виконанні команд, які вимагають декількох звертань до зовнішньої пам'яті. Одним із шляхів збільшення кількості звертань до зовнішньої пам'яті є використання швидкодіючої пам'яті, яка підтримує декілька послідовних звертань за один командний цикл. При переміщенні швидкодіючої пам'яті або її частини за межі кристала виникають

питання, які пов'язані з мінімізацією затримок та усунення завад під час обміну між мікропроцесором і зовнішньою пам'яттю.

Багатопортова пам'ять. Використання багатопортової пам'яті, особливістю якої є незалежні множини шин адреси і даних, дозволяє упродовж одного циклу виконання команди збільшити кількість звертань до пам'яті. Багатопортова пам'ять підтримує одночасний доступ до неї з усіх портів, кількість яких відповідає кількості шин адреси і даних. Така пам'ять забезпечує паралельний доступ до множини даних і вирішення усіх конфліктів, що виникають при такому доступі. Реалізація такої пам'яті вимагає великих апаратних витрат, які насамперед залежать від паралельності доступу, а також від ємності пам'яті. Переважно у комп'ютерних системах обробки сигналів і зображень багатопортову пам'ять використовують в парі з однопор- товою пам'яттю.

Кеш-пам’ять. Використання кеш-пам'яті в складі комп'ютерних систем обробки сигналів і зображень дозволяє підвищити їх швидкодію завдяки уникненню звертань до менш швидкодіючої пам'яті під час зчитування даних і команд. Кеш-пам'ять комп'ютерних систем обробки сигналів і зображень відрізняється невеликою ємністю та простотою функціонування. Найпростішим типом кеш-пам'яті є буфер повторення однієї інструкції. Це є кеш на одне слово команди з використанням спеціальної команди повторення. Команда, яка буде виконуватись декілька разів завантажується в буфер перед її першим виконанням. Цю команду виконують зчитуючи її з кеш-пам'яті, що дозволяє звільнити пам'ять програм для доступу до даних. Цей тип кеш-пам'яті використовується в мікропроцесорах TMS320C2x і TMS320C5x, де він дає змогу у разі повторного виконання команди забезпечити продуктивність, яка може бути досягнута при трьох звертаннях до пам'яті за один командний цикл. Очевидним недоліком використання буферу повторення є те, що він працює тільки з однією командою. Якщо для деяких алгоритмів цей метод використання кеш-пам'яті є ефективним, то для алгоритмів з повторенням блоків команд він є недоцільним.

Концепцію буферу повторення можна вдосконалити, розширяючи його на запам'ятовування кодів декількох команд. Прикладом такої конфігурації кеш-пам'яті може бути буфер повторення на 16 команд мікропроцесора DSP16xr (ATT&T). В цьому мікропроцесорі при першому виконанні блоку команд він копіюється в буфер, а при кожному його повторі команди читаються з буферу, звільняючи тим самим процесор від додаткових звертань до пам'яті. Такий тип кеш-пам'яті є ефективним при виконанні алгоритмів ЦОС і зображень, тому що більшість з них використовують цикли з декількох команд.

Завдання узгодження буферу повторення декількох команд вирішує проста односекторна кеш-пам'ять команд. Ця кеш-пам'ять дозволяє зберігати деяку кількість команд, які найчастіше використовуються. Односекторна кеш-пам'ять команд завантажується кожною командою, що виконується і відстежує адрес команд в кеші. Коли процесор звертається за командою, то у разі наявності її в кеш-пам'яті вона зчитується, а за її відсутності - вміст кеш-пам'яті стає недійсним. Дана кеш-пам'ять є ефективною при доступі до єдиної неперервної ділянки пам'яті програм. Прикладом мікропроцесора з використанням односекторної кеш-пам'яті є ZR3800x фірми *Zoran.* Як і буфери повторення для декількох команд, так і односекторна кеш-пам'ять використовується для вирішення завдань повторного виконання невеликих груп команд.

Гнучкішою структурою є кеш-пам'ять з декількома незалежними секторами. Кеш-пам'ять такого типу функціонує подібно до простої односекторної кеш-пам'яті, але може зберігати дві або і більше незалежні ділянки пам'яті програм. Такий тип кеш-пам'яті використовують у високопродуктивних процесорах обробки сигналів і зображень. Так, у процесорі TMS320C3x ця пам'ять складається з двох секторів ємністю 32 слова. Кожен сектор може використовуватися для зберігання команд з незалежних ділянок пам'яті програм об'ємом 32 слова. Дана кеш- пам'ять працює так: у разі потрапляння адреси в один із секторів кеш-пам'яті (співпадіння кешу), слово зчитується з кешу, а при не співпадінні адреси - генерується сигнал промаху сектора. В цьому випадку весь вміст одного зі секторів оновлюється за алгоритмом LRU (*least recently used),* за яким вибирається той сектор, до якого довгий час не було звертань.

У деяких процесорах для управління кеш-пам'яттю використовуються спеціальні команди або біти управління, які дають змогу програмісту заблокувати вміст кешу в деякій точці виконання програми або заборонити кеш-пам'ять взагалі. Ці можливості дозволяють програмісту здійснювати ручний контроль за кеш-пам'яттю, який іноді допомагає підвищити продуктивність комп'ютерних систем обробки сигналів і зображень.

4. Лекція «Аналіз способів доступу до пам'яті комп'ютерних систем на ПЛІС.»

Доступ з використанням станів очікування. Пам'ять комп'ютерних систем обробки сигналів і зображень є багаторівневою, що передбачає спільне використання різних за швидкодією, ємністю і вартістю пристроїв пам'яті. Ємність пам'яті на кожному із рівнів збільшується в напрямку від процесора, а швидкодія - в протилежному напрямку. Для організації ефективного обміну з повільною зовнішньою пам'яттю в комп'ютерних системах обробки сигналів і зображень використовується генератор станів очікування, яким керують за допомогою відповідних маніпуляцій з регістрами управління шинами. У цих регістрах є два поля, одне з яких слугує для вибору режиму генерації стану очікування, а друге - для завантаження внутрішнього таймера, який формує внутрішній сигнал готовності. В мікропроцесорі TMS320C3x використовується чотири режими генерації стану очікування: зовнішній сигнал готовності - RDY, внутрішній сигнал готовності - RDYwtcnt, логічне "I" сигналів - RDY і - RDYwtcnt, логічне "АБО" сигналів - RDY i-RDYwtcnt. Всі чотири режими можуть використовуватись для генерації внутрішнього сигналу *RDYint,* який керує доступом. Доки цей сигнал дорівнює "1", поточний зовнішній доступ затримується, а після встановлення його в "0" поточний доступ завершується.

Окрім звертання до повільної пам'яті стани очікування можуть виникати у разі спільного використання шин, при спробі процесора здійснити одночасний доступ до пам'яті, яка не підтримує множинний доступ, та при реалізації конвеєра команд. Тривалість стану очікування відносно тривалості командного

циклу може змінюватись в діапазоні від четвертини командного циклу до декількох командних циклів. Менша тривалість стану очікування дає змогу мінімізувати час доступу до зовнішньої пам'яті.

Спеціалізовані команди доступу до пам 'яті В більшості процесорів обробки сигналів і зображень використовуються спеціалізовані команди запису даних в пам'ять паралельно до читання команд і даних. Для виконання такого доступу в процесорах використовується внутрішній паралелізм, що забезпечується декількома множинами шин адреси і даних. Внутрішній паралелізм підтримується відповідним набором команд, які характерні для алгоритмів ЦОС. До цього набору команд можна віднести такі: одночасне читання пам’яті даних і пам'яті програм, обчислення з читанням з пам'яті, обчислення із записом в пам'ять, обчислення з одночасним читанням даних і команд.

Однією з найчастіше використовуваних операцій в ЦОС є операція обчислення суми добутків, яку виконують так: завантажити два операнди, перемножити операнди і додати добуток до суми попередніх добутків.

Прямий доступ до пам 'яті Прямий доступ до пам'яті (ПДП) - це режим введення-виведення даних в пам'ять без участі процесора. Для здійснення операцій передачі даних між пам'яттю і пристроями введення-виведення на кристалі у більшості мікропроцесорів обробки сигналів і зображень розташований контролер ПДП, який дозволяє виконувати дані операції без зниження продуктивності мікропроцесора. Переважно контролер ПДП мікропроцесорів ЦОС дозволяє звертатися до будь-якої комірки адресного простору процесора. Крім того, такі контролери характеризуються високою швидкодією. Наприклад, контролер ПДП процесора TMS320C4x в комбінації з внутрішньокристальною пам'яттю дозволяє виконувати упродовж командного циклу одне звертання до пам'яті.

Подальший розвиток ПДП-контролерів процесорів ЦОС спрямований на забезпечення декількох паралельних передач. Такі контролери ПДП повинні мати декілька каналів, кожний з яких може забезпечувати обмін як із повільною зовнішньою пам'яттю, так із пристроями введення-виведення. Для забезпечення

багатоканального обміну до складу контролера повинні входити генератори адреси, регістри управління та лічильники передач. Кількість каналів, які може обслуговувати контролер ПДП, в процесорах різних фірм є неодинаковою. Наприклад, контролер ПДП процесора TMS320C40 обслуговує 6 каналів, а процесор ADSP2106x - 10 каналів, причому кожен з каналів може використовуватися для обміну типу пам'ять-пам'ять і пам'ять-периферія.

10.1. Методи проектування логічної структури ПЛІС

Комп’ютерні системи на програмованих логічних мікросхемах створюють із використанням логічного, схемного та проектування за допомогою мов опису апаратного забезпечення.

Проектування за допомогою мов опису апаратного забезпечення виконує:

а) розробник, який описує необхідну поведінку цифрового пристрою за допомогою поведінкової моделі на функціональному рівні. Для цього він може скористатися спеціалізованими мовами опису апаратного забезпечення, графів скінченних автоматів та ін. В подальшому основну увагу буде приділено методам синтезу на основі мови опису апаратного забезпечення VHDL.

б) система автоматизованого проектування синтезує логічну структуру. Після цього апаратне забезпечення відтворює бажану поведінку.

На рис. 10.1 зображено один з можливих варіантів побудови процесу проектування цифрового пристрою на базі ПЛІС від формулювання проектного завдання до кінцевого результату -- реалізованого зразка.

Розробка технічного завдання на проектування цифрового пристрою:

- формування вимог та обмежень до проекту;
- вибір способу реалізації та елементної бази;
- формування плану роботи над проектом

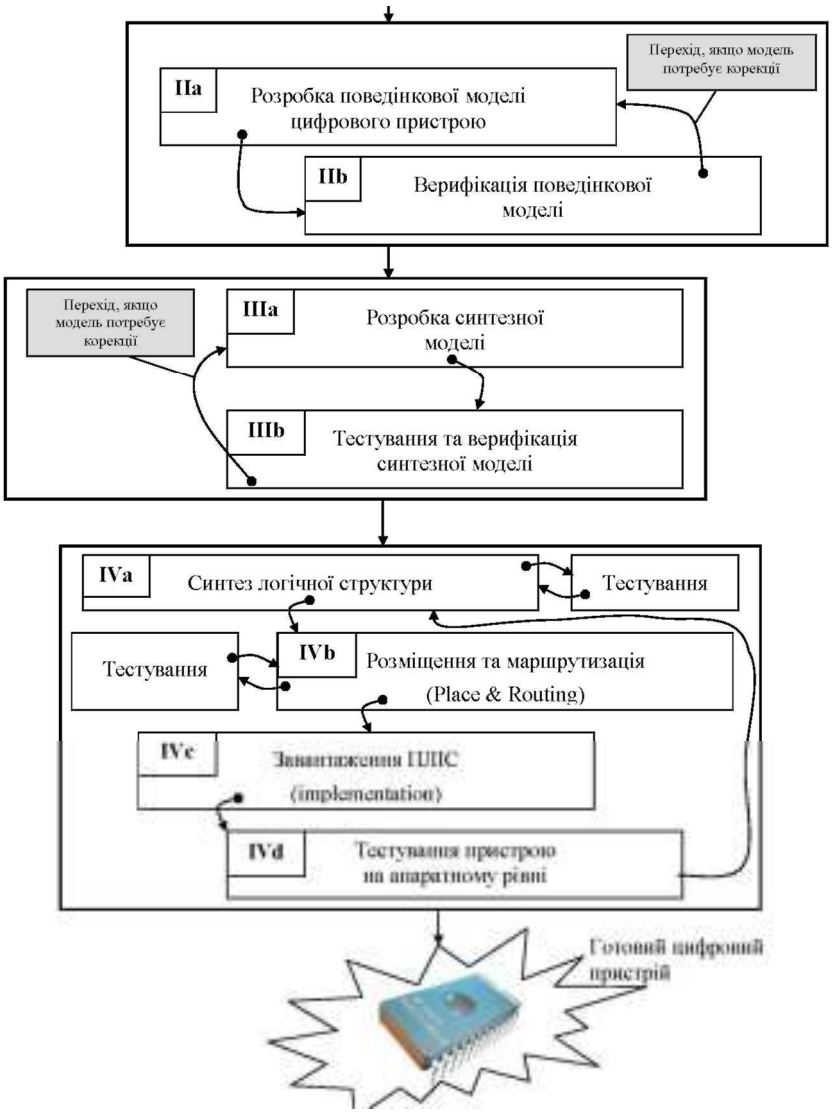


Рис. 10.1. Поведінкове проектування цифрових пристроїв на базі ПЛІС

У процесі поведінкового проектування на всіх його етапах створюється ряд моделей різних аспектів роботи цифрових пристроїв та різних рівнів абстракції. Причому процес проектування полягає в переході від моделей з вищим рівнем абстракції до моделей з нижчим рівнем абстракції, тобто поступовій деталізації опису цифрового пристрою. Наведемо класифікацію моделей, що застосовуються в процесі проектування.

Поведінкова модель (Behavioral model, Interpreted model) показує реакцію моделі на зміну вхідних сигналів з урахуванням часу реакції, але не містить детального опису апаратної реалізації пристрою, що моделюється. Рівень абстракції залежить від детальності опису моделі. Наприклад, поведінкова модель може описувати процесор, що виконує абстрактний алгоритм, або це може бути модель процесора на нижчому рівні абстракції - рівні множини

інструкцій. Точність деталізації вхідних і вихідних даних залежить від рівня абстракції моделі.

Функціональна модель (Functional model) описує функції системи без вказання способу реалізації цих функцій. Дана модель показує лише реакцію системи чи її компонента без урахування часового фактора (визначає значення виходу, але не час його встановлення). Рівень абстракції залежить від ступеня деталізації моделі. Рівень деталізації вхідних та вихідних даних залежить від рівня абстракції.

Структурна модель (Structural model) представляє компоненти чи системи з погляду взаємозв'язків між підкомпонентами, що їх утворюють. Структурна модель має відповідати фізичній ієрархії в описуваному об' єкті. Ієрархія, в свою чергу, визначається фізичною організацією конкретної реалізації. Структурна модель описує фізичну структуру конкретної реалізації шляхом визначення компонентів та топології їх взаємозв' язків. Компоненти можуть бути описані на структурному, функціональному чи поведінковому рівнях. Моделювання структурної моделі вимагає наявності поведінкових моделей всіх нижчих гілок ієрархії, отже, ступінь деталізації модельного часу, значень об' єктів даних та функціональності моделі залежить від ступеня деталізації моделей компонентів.

Модель продуктивності (Performance Model, Uninterpreted Model) дозволяє визначити лише часові аспекти роботи цифрового пристрою, що моделюється (тобто швидкість реакції на зміну вхідного сигналу, але не значення вхідного сигналу).

Модель інтерфейсу (Interface Model, bus functional) називають ще чорним ящиком. Може містити деталізацію по всіх аспектах процесів обміну інформацією між об' єктом та зовнішнім середовищем, включаючи функціональність, часові характеристики, значення даних тощо. Така модель не містить інформації про внутрішню структуру об' єкта.

Модель змішаного рівня, гібридна модель (Mixed-Level Model, Hybrid Model) - модель, що містить компоненти, описані на різних рівнях абстракції або різними класами моделей.

Віртуальний прототип (Virtual Prototype) - це комп'ютерна імітаційна модель кінцевого продукту, компонента чи системи. При цьому від такої моделі не вимагається дотримання жодних спеціальних умов щодо її характеристик. Термін "віртуальний прототип" позначає клас моделей, що відіграють певну роль у процесі проектування, зокрема:

• показує можливі варіанти реалізації проекту;

• демонструє концепцію проекту;

• дає можливість перевірки проекту на відповідність вимогам та адекватності поставленій задачі.

Повернемося до процесу проектування, наведеному на рис. 1.4. Весь процес розбито на 4 головні етапи.

На першому етапі формується технічне завдання для проектування цифрового пристрою. На цьому етапі формуються інтерфейсна, продуктивна та функціональна моделі, із застосуванням яких можна постійно перевіряти моделі, сформовані на наступних етапах на відповідність технічному завданню. Так, продуктивна модель визначає швидкодію проектованого пристрою, інтерфейсна -спосіб його інтеграції до вищого ієрархічного рівня, а функціональна модель визначає алгоритм перетворення інформації в проектованому пристрої.

На другому етапі формується поведінкова модель. У VHDL під поведінковою розуміється модель, написана із застосуванням всіх існуючих в цій мові конструкцій та типів даних, наприклад, дійсних чисел, файлів, вказівників динамічної пам' яті тощо. Поведінкова модель, розроблена на другому етапі, повинна повністю відповідати вимогам, сформованим на першому етапі. Саме це й перевіряється на підетапі IIb.

Після формування остаточного вигляду поведінкової моделі розробник переходить до етапу III - створення синтезної моделі. Синтезна модель також належить до класу поведінкових, однак може бути написана лише за допомогою певної підмножини конструкцій мови VHDL, які підтримуються засобами синтезу логічної структури. На даному етапі розвитку засоби синтезу (перетворення VHDL-програми на схему логічних елементів) підтримують не всі можливі у VHDL мовні конструкції, наприклад, не підтримуються операції з

дійсними числами чи вказівники. Такі конструкції, як, наприклад, файли, із зрозумілих причин відносяться до природно-несинтезованих. Перехід від поведінкової моделі до синтезованої характеризується зниженням рівня абстракції описання пристрою.

На четвертому етапі здійснюється перехід від синтезної моделі до логічної структури та бітового потоку, який завантажується

11. Лекція «Особливості структурної організації пам'яті комп'ютерних систем на ПЛІС.»

11.1. Проектування комбінаційних схем

Функціональні вузли комбінаційного типу характеризуються однозначною відповідністю вихідних сигналів допустимим комбінаціям сигналів на вході пристрою та не залежать від послідовності їх зміни. Для побудови комбінаційного функціонального вузла необхідно задати всю множину вхідних слів (кодів) і відповідний їм набір вихідних кодів, або систему логічних рівнянь для кожного розряду вихідної коду.

До комбінаційних функціональних вузлів відносяться перетворювачі коду (окремим випадком яких є шифратори і дешифратори), мультиплексори, демультиплексори, комбінаційні суматори та ін.

Для реалізації логічних функцій можуть використовуються мультиплексори або дешифратори.

Якщо логічна функція має N вхідних змінних, то загальне число комбінацій вхідних змінних складає $M = 2^N$.

Реалізація логічної функції на базі мультиплексора припускає, що використовується мультиплексор з числом керуючих входів, що дорівнює числу вхідних змінних логічної функції (N), і відповідно у мультиплексора є 2^N інформаційних входів. Для реалізації логічної функції на базі мультиплексора необхідно подати всі N вхідних сигнали на керуючі входи мультиплексора. Далі необхідно подати логічні «1» на всі інформаційні входи мультиплексора, адреса яких входить в набір комбінацій вхідних змінних, для яких вихідне значення функції визначене як «1». На решту всіх інформаційних входів мультиплексора

необхідно подати «0». На рис. 11.1 представлений приклад реалізації на базі мультиплексора логічної функції, яка задана таким рівнянням (1):

$$y = \bar{x}_0 \cdot \bar{x}_1 \cdot \bar{x}_2 + x_0 \cdot x_1 \cdot \bar{x}_2 + x_0 \cdot \bar{x}_1 \cdot x_2 + x_0 \cdot x_1 \cdot x_2 \quad (1)$$

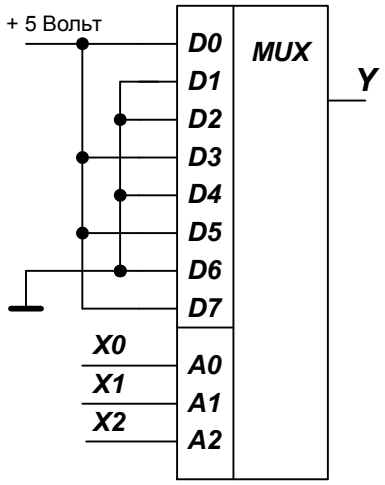


Рис. 11.1. Реалізація логічної функції на базі мультиплексора

Для реалізації логічної функції від N вхідних змінних на базі дешифратора необхідно об'єднати через багатовхідний логічний елемент АБО всі виходи дешифратора, адреса яких належить до набору комбінацій вхідних змінних, для яких вихідне значення функції визначене як «1». На рис. 11.2 представлений приклад реалізації логічної функції на базі дешифратора.

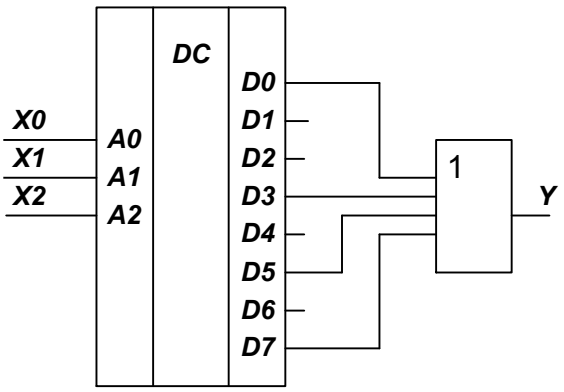


Рис. 11.2. Реалізація логічної функції на базі дешифратора

Мова VHDL дає можливість застосувати багаторівневий підхід до реалізації проектів, коли робочий проект (верхнього рівня) містить низку

проектів (нижнього рівня), кожен з яких реалізує частину функцій системи. В такому випадку проекти нижнього рівня називаються компонентами та розглядаються як елементи архітектури основного проекту (рис. 11.3). Використання компонентів вимагає їх оголошення та підключення в проекті.

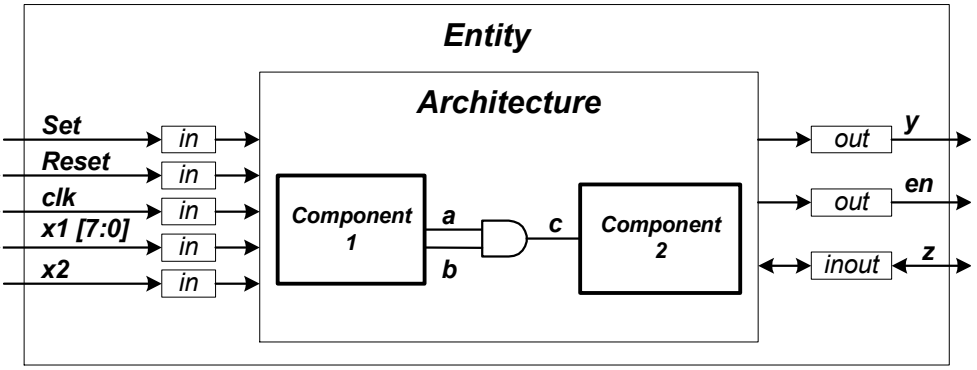


Рис. 11.3. Елементи архітектури основного проекту

Оголошення компоненту проводиться в декларативній частині архітектури головного проекту:

component ім'я_компонента
port (порт_1 : режим тип
порт_2 : режим тип
. . .
порт_N : режим тип);
end component ;

При оголошенні компонента необхідно вказати в полі *ім'я_компонента* початкову назву проекту нижнього рівня, вказану при його оголошенні (Entity), а також описати його інтерфейс. Імена, режими та типи портів інтерфейсу компонента повинні відповідати іменам, режимам і типам портів в операторі ***Entity*** його початкового файлу. Нижче представлений приклад оголошення паралельного восьмирозрядного синхронного регістра:

component parallel_reg

port (clk : **in** std_logic ; -- вхід синхронізації
rst : **in** std_logic ; -- вхід скидання
load : **in** std_logic ; -- вхід дозволу запису
data_in : **in** std_logic_vector (7 downto 0) ;
data_out : **out** std_logic_vector (7 downto 0));
end component ;

Далі у виконуваній частині архітектури основного проекту після слова ***begin*** виконується підключення компоненту:

Ідентифікатор **: ім'я_**компонента
port map (порт_1 => сигнал_1
порт_2 => сигнал_2
. . .
порт_N => сигнал_N);
end component ;

Підключаючи компонент необхідно дотримуватися таких правил:

- тип порту компоненту повинен відповідати типу сигналу, до якого він підключається;
- якщо порт компоненту не використовується, то записується слово ***open***;
- якщо розрядність порту не відповідає розрядності сигналу, то необхідно вказати розряди сигналу які підключаються до порту.

Наприклад, якщо компонент має восьмирозрядний порт *data_out*, а підключається до старшого байта 16-розрядного сигналу (шини) *data_bus*, то в рядку, де записується підключення порту, слід записати:

data_out => data_bus (15 downto 8).

12. Лекція «Аналіз способів доступу до пам'яті комп'ютерних систем на ПЛІС.»

12.1. Проектування цифрових автоматів

Цифровим автоматом називається послідовна схема з внутрішньою пам'яттю, яка в процесі роботи приймає низку станів. На відміну від регулярних послідовних схем (лічильники, дільники), де перехід з одного стану в інший відбувається послідовно, напрям переходу в цифровому автоматі визначається станом цифрового автомата та рівнем сигналів на його входах. Така властивість визначає область використання цифрових автоматів для керування складними цифровими системами.

Для опису будь-якого цифрового автомата використовують:

- безліч допустимих станів цифрового автомата $S = \{S_0, S_1, S_2, \dots S_K\}$, де K – кількість станів. При цьому один із станів S_0 є початковим.
- безліч входів $I = \{I_0, I_1, I_2, \dots I_N\}$, де N – загальна кількість входів цифрового автомата;
- безліч виходів $O = \{O_0, O_1, O_2, \dots O_M\}$, де M – загальна кількість виходів цифрового автомата;
- функція переходів δ визначає стан цифрового автомата у момент часу $t+1$ залежно від стану автомата та вхідних сигналів у момент часу t: $S_{t+1} = \delta(S_t, I_t)$.
- функція виходів λ визначає значення вихідних сигналів за станом цифрового автомата $O = \lambda(S)$ (для цифрового автомата Мура) та за станом цифрового автомата і вхідних сигналів $O = \lambda(S, I)$ (для автомата Міля).

Узагальнена структура цифрового автомата (рис. 12.1) містить:

- ***State Register*** - схема пам'яті на базі n-розрядного паралельного регістра. Використовується для зберігання інформації про поточний стан (***Current State***) цифрового автомата;
- ***Next State Logic*** - вхідна комбінаційна схема. Використовуючи значення вхідних сигналів і сигналу ***Current State*** формує сигнал ***Next State***, що визначає перехід цифрового автомата з поточного в

наступний стан (реалізує функцію переходів);

– ***Output Logic*** – вихідна комбінаційна схема. Формує вихідні сигнали на основі стану цифрового автомата та вхідних сигналів (реалізує функцію виходів).

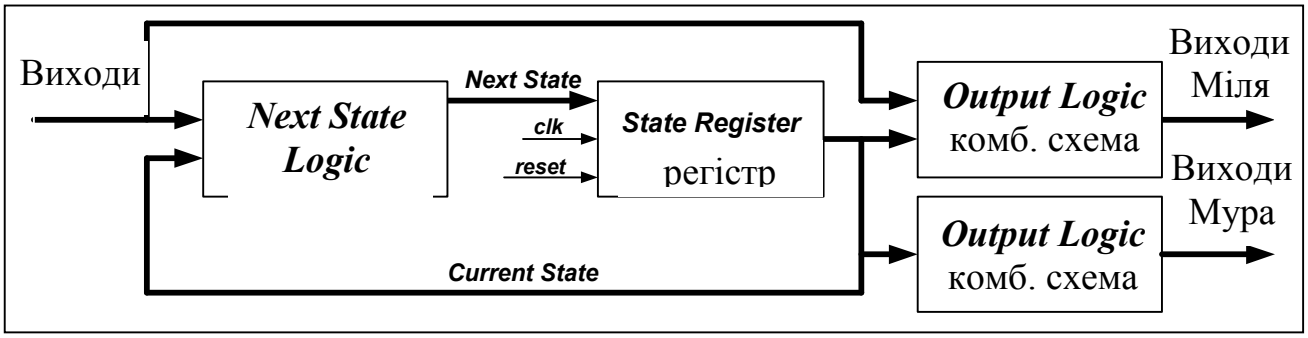


Рис. 12.1. Узагальнена структура цифрового автомата

На практиці використовують два типи цифрових автоматів: автомат Міля та автомат Мура. В автомата Мура значення вихідних сигналів визначається тільки станом самого цифрового автомата (рис. 12.1). В автомата Міля значення вихідних сигналів визначається як станом автомата, так і вхідними сигналами (рис. 12.1).

Опис алгоритму роботи цифрового автомата визначають за допомогою таблиці переходів або ж за допомогою орієнтованого графа, вершини якого відповідають внутрішнім станам автомата, а направлені ребра визначають напрям переходу між станами.

Наприклад, в таблиці 1 представлено суміщену таблицю переходів і виходів автомата Міля в якого є:

– безліч допустимих станів S = {RED, BLUE, GREEN, WHITE};
– безліч входів ***I = {X1, X2}***;
– безліч виходів O = {led_0, led_2, led_3};
– функція переходів δ і функція виходів λ задається суміщеною таблицею переходів і виходів (табл. 12.1).

Таблиця 12.1

Суміщена таблиця переходів і виходів цифрового автомата Міля

Вхідні сигнали	Стани			
	RED	BLUE	GREEN	WHITE
X(0)	BLUE / led0, led3	WHITE / led1, led2	RED / led2, led3	RED / led0, led1
X(1)	WHITE / led1, led3	GREEN / led0, led2	BLUE / led3	GREEN / led2

Комірка в таблиці, що знаходиться на перетині рядка вхідного сигналу ***X(0)*** або ***X(1),*** а також стовпця поточного стану автомата (***RED, BLUE, GREEN, WHITE***) вказує наступний стан автомату. Після косої межі вказуються активні вихідні сигнали в поточному стані, якщо відповідний вхід цифрового автомата є активним (присутня логічна «1»). Граф вказаного автомата Міля представлений на рис. 12.2.

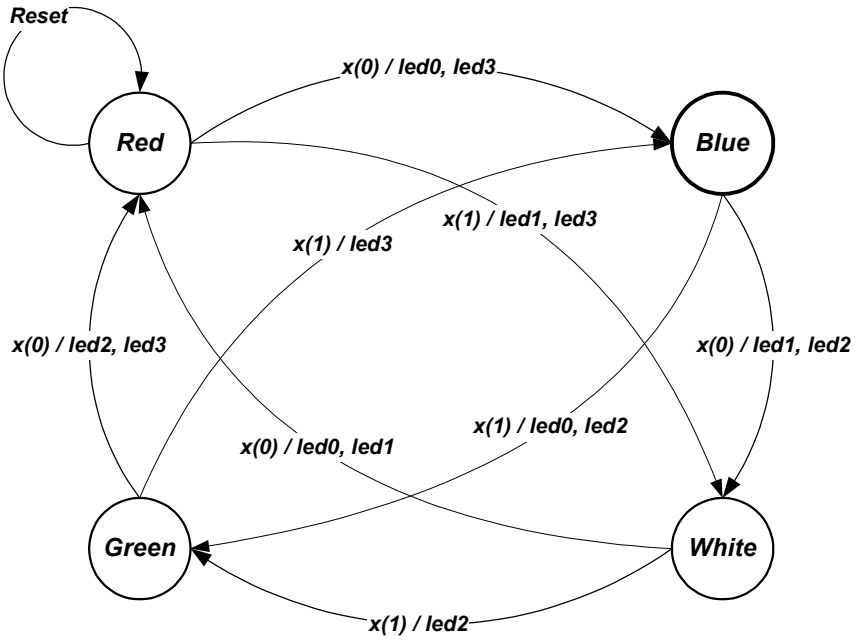


Рис. 12.2. Граф автомата Міля

Кожна вершина представленого графа співвідноситься з одним із можливих станів автомата. Ребра графа визначають перехід з поточного стану. При цьому умова переходу в наступний стан визначатися сигналом вказаному в

чисельнику. В знаменнику вказані активні вихідні сигнали для поточного стану при вхідному активному сигналі.

В табл. 12.2 представлено суміщену таблицю переходів і виходів для цифрового автомата Мура, а на рис. 12.3 представлено його граф.

Таблиця 12.2

Суміщена таблиця переходів і виходів цифрового автомата Мура

Вхідні сигнали	Стани				
	RED	BLUE	GREEN	WHITE	
X(0)	BLUE	WHITE	RED	RED	
X(1)	WHITE	GREEN	BLUE	GREEN	
	led0, led3	led1, led2	led2, led3	Led0, led1	Вихідні сигнали

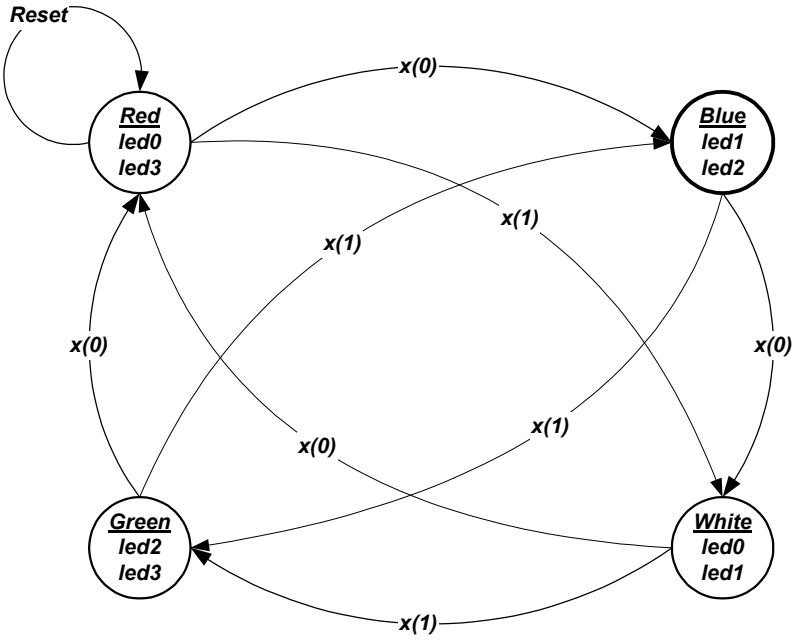


Рис. 12.3. Граф автомата Мура

Для цифрових автоматів Мура характерна залежність значення вихідних сигналів тільки від стану автомата. Тому вершини графа, які відображають його стан, сполучені ребрами, на яких вказаний активний вхідний сигнал, що визначає умова переходу. Перелік активних вихідних сигналів для кожного стану записують безпосередньо у вершині графа.