**СТРУКТУРНА ТЕОРІЯ ЦИФРОВИХ АВТОМАТІВ**

У порівнянні з абстрактною теорією автоматів у структурній теорії робляться подальші кроки в напрямку врахування більшого числа властивостей реально існуючих дискретних автоматів. Головна відмінна риса структурної теорії автоматів полягає в тому, що, на відміну від абстрактної теорії, вона враховує структуру вхідних і вихідних сигналів автомата, а також його внутрішню структуру на рівні так званих структурних схем. Основною задачею структурної теорії є вивчення композиції автоматів, тобто методів побудови складних автоматів з автоматів, що являються відносно більш простими.

Варто підкреслити, що структурна теорія на ставить своєю задачею відбити усі властивості реально існуючих автоматів. У ній, наприклад, зовсім не враховуються перехідні процеси в автоматах, питання надійності роботи автоматів, фізичні властивості сигналів і багато чого іншого. У цьому розумінні структурна теорія автоматів також залишається в значній мірі абстрактною, хоча вона і відрізняється в значно меншим ступенем абстракції, чим власне абстрактна теорія автоматів. Таким чином, структурна і абстрактна теорії автоматів є двома різними ступенями загальної теорії автоматів.

Однак, при синтезі реальних автоматів багато питань простіше й ефективніше вирішуються на рівні абстрактної теорії. До числа таких питань відносяться визначення необхідного обсягу пам'яті автомата, переходів у пам'яті, а також питання, що відносяться до мінімізації числа станів автомата. У той же час, мається ряд питань – таких, наприклад, як питання композиції автоматів, - сама постановка яких виводить за рамки абстрактної теорії автоматів. Таким чином, абстрактна теорія автоматів і структурна теорія автоматів взаємно доповнюють одна одну і мають власні природні області застосування.

Як відзначалося раніше, основною задачею структурною теорією автоматів є побудова структурних схем автоматів на основі композиції автоматів, що належать до наперед заданого скінченного числа типів автоматів. Більш того *основна задача структурної теорії автоматів* може бути сформульована в такий спосіб:

Нехай нам задана деяка скінченна множина автоматів у тому самому структурному алфавіті Х. Назвемо ці автомати *елементарними* і, припустимо, що кожний з елементарних автоматів мається в нашому розпорядженні в необмеженому числі екземплярів. Нехай далі заданий довільний скінченний автомат А у тому ж самому структурному алфавіті Х. Необхідно знайти алгоритм, що дозволяє по заданому абстрактному автомату А будувати деяку композицію елементарних автоматів так, щоб отриманий у результаті композиції структурний автомат породжував відображення, що еквівалентне відображенню, яке породжує автомат А.

Система елементарних автоматів називається *структурно* *повною*, якщо за допомогою автоматів цієї системи можна побудувати структурний автомат, еквівалентний будь-якому наперед заданому абстрактному автомату A.

Відзначимо, що далеко не для кожної структурно повної системи елементарних автоматів основна задача структурної теорії автоматів має *«ефективне»* рішення. Більш точно: в загальному випадку не існує алгоритмів, істотно більш простих, ніж метод перебору всіх варіантів, які вирішують цю задачу. Тобто, не для всякої структурно повної системи елементарних автоматів існує ефективний алгоритм, який дозволяє для будь-якого наперед заданого *АА* побудувати еквівалентний йому структурний автомат. Однак, якщо елементарні автомати структурно повної системи вибрати деяким спеціальним чином, то такі «ефективні» алгоритми існують.

В подальшому будемо розглядати один із можливих таких алгоритмів - *канонічний метод структурного синтезу автоматів з пам’яттю* (КМ), який оперує з елементами автоматами (із специфічними властивостями) двох класів:

1) *логічними елементами* - автоматами без пам’яті, або, як їх іще називають, автоматами з одним внутрішнім станом;

2) *запам’ятовуючими елементами* - автоматами з пам’яттю, або, ще говорять, автоматами, що мають більше одного внутрішнього стану.

Говорять, що автомат з пам’яттю володіє *повною системою переходів*, якщо рівняння *в = δ(a,x)* (де *а, в* – деякі стани, *x* – вхідний сигнал, *δ* – функція переходів) має рішення відносно *х* для будь-якої пари станів *(а,в)*.

Говорять, що автомат за пам’яттю володіє *повною системою виходів*, якщо для будь-якого *у* має рішення відносно *х* рівняння *y = l(a,x).*

Теорема В.М. Глушкова про структурну повноту:

система елементарних автоматів, що включає в себе деяку функціонально повну систему логічних елементів і автомат Мура з нетривіальною пам’яттю (з кількістю внутрішніх станів > 1), що володіє повною системою переходів та виходів, являється структурно повною. Більше того, у цьому випадку існує алгоритм (його називають канонічним методом структурного синтезу автоматів з пам’яттю), який дає рішення основні задачі структурної теорії автоматів і (як ми побачимо далі) дозволяє в звести задачу структурного синтезу довільних скінченних автоматів з пам’яттю до задачі структурного синтезу комбінаційних схем.

[без доведення]

**Тема** (лабораторна робота №1)**:** **ПРОЕКТУВАННЯ КОМБІНАЦІЙНИХ СХЕМ НА МІКРОСХЕМАХ РІЗНОГО СТУПЕНЮ ІНТЕГРАЦІЇ**

Надалі під *логічним елементом* (ЛЕ) будемо розуміти електронну схему, що реалізує певну булеву функцію від n змінних (див., наприклад, додаток 1 цих методичних вказівок). Слово “реалізує” означає, що при подачі на вхід ЛЕ деякого двійкового кортежа (тобто електричних сигналів, що кодують 0 і 1) довжини n, на виході ЛЕ ми отримаємо значення, що співпадає із значенням, яке приймає задана булева функція на цьому ж кортежі у її таблиці істинності. У результаті композиції ЛЕ утворюються *комбінаційні схеми* (КС), взагалі говорячи, з *п* входами і *т* виходами. *Характеристична властивість* КС: сукупність вихідних сигналів КС у даний момент часу цілком визначається сукупністю вхідних сигналів, що діють у даний момент часу, і не залежить від вхідних сигналів, що діяли у попередні моменти часу. Говорять, що така схема має один стан. Отже, поводження комбінаційної схеми може бути описано системою булевих функцій. Розрізняють задачі аналізу і синтезу КС.

*Задача аналізу* КС зводиться до знаходження по заданій КС системи булевих функцій, що відбивають логіку її роботи.

*Задача синтезу* КС є оберненою до задачі аналізу і формулюється наступним чином. Дана булева функція і дана система логічних елементів; необхідно побудувати КС, що реалізує задану булеву функцію на заданих логічних елементах.

Процес синтезу КС можна умовно розбити на *три етапи*.

*На першому етапі* за таблицею істинності заданої булевої функції знаходять мінімальну диз'юнктивну нормальну форму (МДНФ) цієї функції та її заперечення.

Зазначимо, якщо булева функція, що описує роботу КС, є частковою (визначеною не на всіх 2n наборах), то попередньо роблять оптимальне її довизначення (при якому функція буде мати більш просту МДНФ).

На цьому перший етап закінчується.

*На другому етапі* функцію записують у так званій *операторній формі*, тобто у вигляді суперпозиції операторів логічних елементів. *Оператором логічного елемента* називають функцію, що реалізується цим елементом. Якщо число входів в операторах досить велике, то отримання операторного запису функції зводиться до її представлення в одній зі *стандартних канонічних нормальних форм. Таких форм вісім*. Продемонструємо процес їх побудови.

Нехай булева функція задана діаграмою Вейча (рис. 1.1):

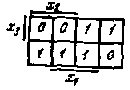


Рисунок 1.1

Легко бачити, що МДНФ даної булевої функції і її заперечення мають, відповідно, наступний вигляд:

*f*(x3, *x*2, *x*1*)* =  і (*x*3, *x*2, *x*1*)* = *,*

Позначати нормальні форми будемо шляхом указівки внутрішньої і зовнішньої функції розкладання. Так, у ДНФ внутрішньою функцією є функція І, а зовнішньої - АБО, тобто ДНФ є формою типу І/АБО*.*

Узявши подвійне заперечення МДНФ функції і застосувавши кілька разів правило де Моргана, послідовно отримаємо наступні нормальні форми:

|  |  |
| --- | --- |
| *f*(*x3, x2, x*1*)* = = | форма І/АБО |
| *=**=* *=* | І-НІ/І-НІ |
| *=* *=* | АБО/І-НІ |
| *=* | АБО-НІ/АБО |

Поставимо над МДНФ заперечення заданої функції

(*x*3, *x*2, *x*1*)* = *,*

ще одне заперечення (тим самим ми знаходимо ще одне аналітичне представлення для функції *f*(*x*3, *x*2, *x*1)) і знайдемо ще чотири нормальні форми:

|  |  |
| --- | --- |
| *f*(*x3, x2, x*1*)* = = | форма І/АБО-НІ |
| = = | форма І-НІ/І |
| = = | форма АБО/І |
| = | АБО-НІ/АБО-НІ |

Нормальні форми дозволяють отримати КС з двома рівнями (каскадами) ЛЕ, якщо елементи мають необхідне число входів.

Якщо ж число входів *р* елементів менше,ніж потрібно для реалізації отриманої нормальної форми, то змінні поєднують у групи, що містять не більше *р* елементів, використовуючи співвідношення виду

x1x2…xm = (x1…xq)…(xs…xm); - асоциативність кон’юнкції.

x1 v x2 v…v xm = (x1v…v xq) v…v (xsv…v xm); - асоциативність диз’юнкції.

; - правило подвійного заперечення.

; - правило подвійного заперечення.

Число груп змінних також не повинне перевищувати *р.* У протилежному випадку зазначені перетворення виконують стосовно груп змінних. Такі перетворення дозволяють представити задану функцію в операторній формі з урахуванням числа входів елементів. Отримана в цьому випадку форма не буде нормальною, тому що за рахунок додаткового каскадування елементів комбінаційна схема буде містити більш двох рівнів.

На *третьому етапі* (заключному) по операторному представленню булевої функції будується комбінаційна схема.

Задана система ЛЕ дозволяє реалізувати одне чи більше операторних представлень функції. Наприклад, при наявності елементів І, АБО і І-НІ можна використовувати в якості вихідної одну з п'яти нормальних форм (І/АБО, І-НІ/І-НІ, АБО/І-НІ, І-НІ/І, АБО/І) для отримання відповідних операторнх представлень з урахуванням числа входів елементів. Для вибору однієї схеми з декількох можливих необхідно порівнювати їх по заданих параметрах. Найбільш часто такими параметрами є *складність* і *швидкодія* КС.

Існують кілька способів *оцінки складності*:

*складність по Квайну* (К), обумовлена як сумарнечисловходів усіх логічних елементів;

складність у числі логічних елементів *М;*

складність у числі умовних корпусів мікросхем:  ,

де *r —* число типів мікросхем; *mi —* кількість мікросхем*i-го*типу; *ni —* число виводів (входів і виходів) мікросхеми *i-го* типу. У якості умовного використовується корпус мікросхеми на 14 виводів.

Параметри *К* и *М* доцільно використовувати при проектуванні інтегральних схем, тому що їхня вартість залежить від площі кристала, яка пропорційна числу логічних елементів і числу їхніх виводів. Оцінка *N* зручна при порівнянні складності пристроїв, побудованих на мікросхемах.

Швидкодія КС залежить від часових параметрів логічних елементів *t*01 і *t*10, що характеризують затримку сигналів елементом (час переходу вихідного сигналу від одного логічного рівня до іншого (час переходу з 0 в 1 або ж, навпаки, – з 1 в 0)). На практиці використовують, зазвичай, усереднене значення часу затримки *t =* (*t*01 + *t*10)/2 або ж максимальне *t\** = max (*t*01, *t*10).

Наприклад, для КС на однотипних елементах середній час затримки сигналів визначається як *T* = *Lt,* де *L* — *рівень схеми*, що дорівнює числу елементів, що входять у максимальний по довжині ланцюжок елементів. Якщо використовуються елементи з різною затримкою, то в схемі визначається шлях, що вимагає максимального часу поширення сигналів.

КС обирається з декількох можливих, що краще інших задовольняє заданим параметрам.

Проектування КС *з багатьма виходами* відрізняється тим, що система булевих функцій піддається спільній мінімізації (див., наприклад, додаток 1 цих методичних вказівок), а потім перетворюється до операторного представлення таким чином, щоб число використаних логічних елементів було мінімальним.

При реалізації булевих функцій у ряді випадків можна зменшити кількість корпусів мікросхем, використовуючи мікросхеми середнього ступеня інтеграції, наприклад, *дешифратори* і *мультиплексори*.

Оскільки дешифратор на *п* входів реалізує всі конституенти одиниці (див., наприклад, додаток 1 цих методичних вказівок), то для реалізації булевої функції від *п* змінних досить за допомогою АБО-елемента отримати диз'юнкцію конституент одиниці тих наборів, на яких функція приймає одиничні значення. Якщо дешифратор має інверсні виходи, то, відповідно до правила де Моргана, замість АБО-елемента варто використовувати елемент І-НІ.

Наприклад, функцію  (зазначимо, якщо зафіксувати змінні і їх порядок у булевій функції певним чином, то тоді 510 = 1012 = х3х2х1, тобто запис 357 – це закодована ДДНФ заданої булевої функції), представлену в ДДНФ, можна реалізувати за допомогою тривходового дешифратора з прямими (рис. 1.2, а) чи інверсними (рис. 1.2, б) виходами.

Якщо з кожної конституенти можна винести за дужки змінну, то при наявності в дешифратора (стробуючого) входу *Е* знадобиться дешифратор з меншим числом інформаційних входів. Представивши останню з розглянутих функцій у вигляді , можна реалізуватиїї з використанням двовходового дешифратора (рис. 1.2, *в*).



Рисунок 1.2

Дешифратори найбільш доцільно використовувати при реалізації систем булевих функцій. У цьому випадку для побудови схем потрібно один дешифратор і стільки логічних елементів, скільки функцій містить система.

Мультиплексор з *N* інформаційними входами і *k* = log2*N* керуючими входами дозволяє виключити *k* з *п* змінних булевої функції, представивши її через ***залишкові функції*** (див., наприклад, додаток 1 цих методичних вказівок), кожна з яких залежить не більш ніжвід *n-k* змінних.

Так, після двократного застосування леми про розкладання булевої функції по змінній, функція *y* = *f*(*x*n, …, *x*1) може бути представлена у формі



де *f*0, *f*1, *f*2, *f*3 — залишкові функції від *n*–2 змінних. Схема, що відповідає такому представленню функції, показана на рис. 1.3.

Залишкові функції можуть бути реалізовані будь-яким способом. Зокрема, кожна з них може бути представлена через свої залишкові функції з використанням мультиплексорів. У кожному конкретному випадку необхідно оцінювати складність схем при тому чи іншому способі реалізації функції. Помітимо, що *складність схем залежить від того, які змінні виключаються*. Для отримання найбільш простої схеми може знадобитися перебір усіх комбінацій виключення змінних.

При реалізації функції від чотирьох змінних з використанням мультиплексора, що має два керуючих входи (рис. 1.4), можна виключити шість різних пар змінних (x4x3, x4x2, x4x1, x3x2, x3x1, x2x1,)*.* При виключенні кожної пари змінних діаграму Вейча заданої функції можна розглядати як чотири самостійні діаграми для залишкових функцій *f*0, *f*1, *f*2, *f*3.

На рис. 1.5 показані всі шість варіантів визначення залишкових функцій. Виключені змінні зазначені біля діаграм для кожного варіанта. Схеми на елементах І-НІ, щовідповідають кожному варіанту, представлені на рис. 1.6.Найбільшпроста реалізація заданої функції за допомогою мультиплексора є при виключенні змінних *х*1 і *х*2 (рис. 1.7).





#### 

#### ІНДИВІДУАЛЬНІ ЗАВДАННЯ

1. Визначити свій варіант булевої функції. Для цього необхідно порядковий номер студента (магістранта) в групі перевести в двійкову систему числення і записати сім його молодших розрядів у вигляді слова *a*7*a*6*a*5*a*4*a*3*a*2*a*1, доповнюючи, у разі потреби, це слово зліва нулями*.* Визначивши значення *ai* підставити їх у табл. 1.1.

Таблиця 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *x*4 | *x*3 | *x*2 | *x*1 | y |
| 0 | 0 | 0 | 0 | *a*1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | *a*2 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | *a*3 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | *а*4 |
| 1 | 0 | 0 | 1 | *а*5 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | *а*6 |
| 1 | 1 | 1 | 1 | *а*7 |

Наприклад, якщо номер варіанта19 (0010011), то у таблицю необхідно підставити

*a*7 *=*0, *a*6 =0, *a*5 =1, *a*4 =0, *a*3 =0, *a*2 =1, *a*1 =1.

2. Для заданої функції і для її заперечення знайти МДНФ.Представити функцію у всіх восьми нормальних формах (див. приклад далі).

3. Отримати операторні представлення функції, що можуть бути реалізовані на елементах, заданих у табл. 1.2. Наприклад, для *а*3*а*2*а*1=000 будуються два операторні представлення функції, що можуть бути реалізовані на елементах: 1) 3І-НІ, 3І, 2) 2АБО-НІ.

4. Визначити операторну форму (із отриманих), що забезпечує отримання комбінаційної схеми з максимальною швидкодією і мінімальними затратами обладнання, тобто схему з кращим параметром *T* і схему з мінімальним значенням *N.* Усі мікросхеми в табл. 1.2 мають по 14 виводів.

Таблиця 1.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *а*3 | *а*2 | *а*1 | Тип елементів | Число елементів у корпусі мікросхеми | Час затримки сигналів |
| 0 | 0 | 0 | 3І-НІ / 3І, 2АБО-НІ | 3 / 3 | 20 / 24 |
| 0 | 0 | 1 | 4І-НІ / 2АБО, 2АБО-НІ | 2 / 4 | 20 / 22 |
| 0 | 1 | 0 | 4І / 2АБО, 2АБО-НІ | 2 / 4 | 24 / 22 |
| 0 | 1 | 1 | 3І / 2АБО, 2І-НІ | 3 / 4 | 24 / 22 |
| 1 | 0 | 0 | 2АБО-НІ / 4І, 2І-НІ | 4 / 2 | 22 / 24 |
| 1 | 0 | 1 | 2І-НІ / 2АБО, 2АБО-НІ | 4 / 4 | 22 / 20 |
| 1 | 1 | 0 | 2АБО-НІ / 3І, 2І-НІ | 4 / 3 | 20 / 24 |
| 1 | 1 | 1 | 2І-НІ, 2АБО-НІ | 4 / 4 | 20 / 20 |

5. На елементах ЗІ-НІ (параметри зазначені в табл. 1.2) побудувати перетворювач кодів відповідно до табл. 1.3. У процесі проектування використовувати методи спільної (сумісної) мінімізації системи булевих функцій. Для отриманої схеми визначити *L, Т* и *N.*

6. Побудувати схему для реалізації функції, заданої табл. 1.1, якщо крім елементів з табл. 1.2 можна використовувати мультиплексори з двома керуючими входами. Два мультиплексора розміщаються в одному корпусі мікросхеми на 16 виводів. Порівняти по складності отриману схему зі схемою, побудованою без використання мультиплексорів.

7. Побудувати перетворювач кодів по табл. 1.3 з використанням елементів ЗІ-НІ і чотиривходового дешифратора з інверсними виходами. Дешифратор розміщений у корпусі з 24 виводами. Порівняти значення *N* для схеми з дешифратором і без дешифратора.

Таблиця 1.3

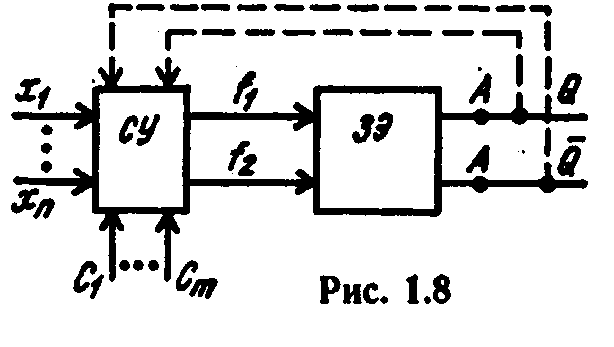
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Інформація | | | | | | | |
| на вході | | | | на виході | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | *а*1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | *а*2 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | *а*3 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | *а*4 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | *а*5 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | *а*6 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | *а*7 |

Тема(лабораторна робота №2):ПРОЕКТУВАННЯ І ДОСЛІДЖЕННЯ ТРИГЕРІВ

*Мета роботи*: вивчення схемних різновидів тригерів і оволодіння методами проектування тригерів.

**Основні теоретичні положення**

У схемах цифрової обчислювальної техніки в якості запам'ятовуючих елементів широко використовуються тригери. *Тригер* – це пристрій із двома стійкими станами, що містить запам'ятовуючий елемент (ЗЭ) і схему керування (СК), як показано на рис. 1.8,

де *x*1,…*xn -*інформаційні входи тригера; *С1,…,Ст -* тактуючі входи; *Q* і  - відповідно прямий і інверсний виходи тригера; *f*1 і *f*2 - функції збудження ЗЕ.

Тригери класифікуються по ряду ознак. У залежності від наявності тактуючих входів розрізняють *синхронні* (є тактуючі входи) і *асинхронні* (немає тактуючих входів) тригери. Найбільше поширення серед синхронних отримали однотактні тригери, що мають тільки один тактуючий вхід.

Функціональна класифікація характеризує стан входів і виходів тригера в момент часу до його переключення (*S*) і після його переключення (*S* + 1). При наявності *п* інформаційних входів можна отримати  типів тригерів. На практиці ж використовують невелике число типів тригерів. До них можна віднести *RS*-, *JK*-, *Т*- і *D-тригери*. *D-тригер* знаходить застосування тільки як синхронний. Інші з зазначених тригерів можуть бути як синхронними, так і асинхронними.

Спосіб функціонування тригерів може бути описаний таблицею переходів. З таблиці переходів асинхронного *RS-тригера* (табл. 1) випливає, що тригер не змінює свого стану в момент часу *S* + 1, тобто , якщо в момент *S* мало місце . При  і  тригер встановлюється в одиничний стан ( ), а при комбінації сигналів  і  - у нульовий ( ). При  стан тригера в момент часу *S* + 1 не визначений ( ). Така комбінація сигналів для *RS-тригера* є забороненою.

Існують різновидності *RS-тригерів*, для яких комбінація сигналів  не є забороненою. *R-тригер* відрізняється від *RS-тригера* тим, що при комбінації вхідних сигналів  він переходить у нульовий стан ( ) (табл. 2). *S-тригер* (табл. 3) у цьому випадку переходить в одиничний стан ( ), а *E-тригер* (табл. 4) не змінює свого стану ( ).

Синхронний *D-тригер* називають тригером затримки (табл. 5). Він здійснює затримку сигналу на один такт (період проходження синхросигналів). Як асинхронний *D-тригер* не знаходить застосування через відсутність режиму збереження інформації ( ).

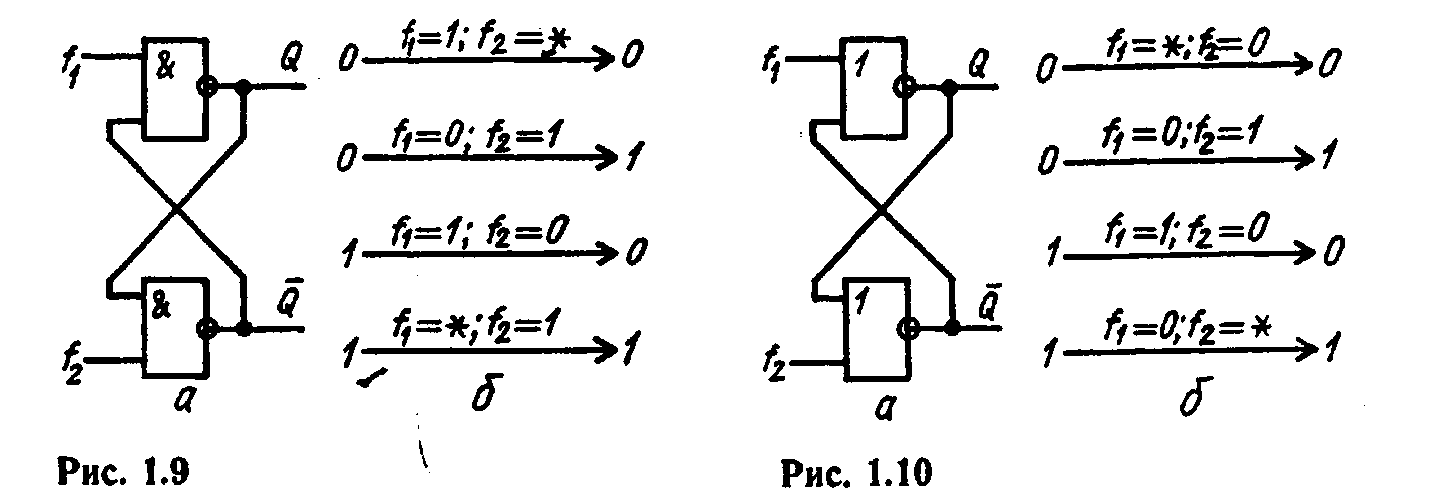
Тригери можуть мати вхід, що *дозволяє, V.* Наприклад, *D-тригер* із входом, що дозволяє, *називають* DV-тригером. При *V* = 1 він працює як *D-тригер*, а при *V =* 0 зберігає свій стан. *DV-тригер* може бути синхронним і асинхронним.

*T-тригер* називають також *рахунковим тригером*. Він здійснює підрахунок одиниць, що надходять на вхід *Т,* по модулю два, що видно з табл. 6. Як випливає з таблиці переходів *JK* -тригера (табл. 7), при комбінаціях вхідних сигналів *J* = *K* = 0, *J* = 0 і *K* = 1, *J* = 1 і *K* = 0 він працює, як *RS-тригер* (вхід *J* відповідає входові *S*, а *K* - *R*), а при *J* = *K* = 1 змінює свій стан на протилежний, тобто працює як рахунковий тригер.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблиця 1 | | | | | | |  | | Таблиця 2 | | | | | | |  | | Таблиця 3 | | | | | |
| RS | | SS | | QS+1 | | |  | | RS | | SS | | | QS+1 | |  | | RS | | | SS | | QS+1 |
| 0 | | 0 | | QS | | |  | | 0 | | 0 | | | QS | |  | | 0 | | | 0 | | QS |
| 0 | | 1 | | 1 | | |  | | 0 | | 1 | | | 1 | |  | | 0 | | | 1 | | 1 |
| 1 | | 0 | | 0 | | |  | | 1 | | 0 | | | 0 | |  | | 1 | | | 0 | | 0 |
| 1 | | 1 | | \* | | |  | | 1 | | 1 | | | 0 | |  | | 1 | | | 1 | | 1 |
| Таблиця 4 | | | | |  | Таблиця 5 | | | | | |  | Таблиця 6 | | | |  | | Таблиця 7 | | | | |
| RS | SS | | QS+1 | |  | CS | | DS | | QS+1 | |  | TS | | QS+1 | |  | | JS | KS | | QS+1 | |
| 0 | 0 | | QS | |  | 0 | | 0 | | QS | |  | 0 | | QS | |  | | 0 | 0 | | QS | |
| 0 | 1 | | 1 | |  | 0 | | 1 | | QS | |  | 1 | |  | |  | | 0 | 1 | | 0 | |
| 1 | 0 | | 0 | |  | 1 | | 0 | | 0 | |  |  | | 1 | 0 | | 1 | |
| 1 | 1 | | QS | |  | 1 | | 1 | | 1 | |  |  | |  | |  | | 1 | 1 | |  | |

Класифікація тригерів по способу запису інформації характеризує процес переключення тригера. Розрізняють наступні типи тригерів:

1. тригери, керовані рівнем сигналів (*без внутрішньої затримки*);
2. тригери, керовані перепадом сигналів (*із внутрішньою затримкою*).



*Активним* назвемо такий вектор (набір) вхідних сигналів, що може викликати зміна стану тригера. У цьому випадку мають місце співвідношення ,  і . Вектор вхідних сигналів, що відповідає режимові збереження інформації ( ), будемо вважати *пасивним*. Наприклад, для тригера, що працює відповідно до табл. 5, вектор *СD* є активним при значеннях 10 і 11, а при 00 і 01 - пасивним. Для *T-тригера* (табл. 6) значення 1 відповідає активному сигналові *Т*, а 0 - пасивному. Для правильної роботи тригерів необхідно, як правило, чергувати активні набори сигналів з пасивними.

Тригери, керовані рівнем сигналів, змінюють свій стан безпосередньо після надходження на їхні входи активного вектора вхідних сигналів. У тригерах, керованих перепадом, сигнали на виході змінюються тільки після зняття активного вектора вхідних сигналів.

У синхронних тригерах при зміні вектора вхідних сигналів необхідно забезпечити зміну тільки тактуючого сигналу. Якщо тригер змінює свій стан при переході тактуючого сигналу з 0 у 1, то говорять, що тригер спрацьовує по позитивному перепаді, а при переході з 1 у 0 - по негативному перепаді тактуючого сигналу.

Характерна риса синхронних тригерів, керованих рівнем сигналу С, полягає в тому що при сталому активному рівні сигналу Свони можуть переключатися стільки разів, скільки разів змінюються інформаційні сигнали. У тригерах, керованих перепадом, у цьому випадку вихідні сигнали *Q.* і  змінюються тільки після зняття активного рівня тактуючого сигналу. У деяких реальних схемах тригерів із внутрішньою затримкою зміна інформаційних сигналів при сталому рівні тактуючого сигналу може спричинити переключення ряду логічних елементів, однак на виходах тригера сигнали при цьому не змінюють свого значення.

1. Формулювання задачі синтезу тригерів.

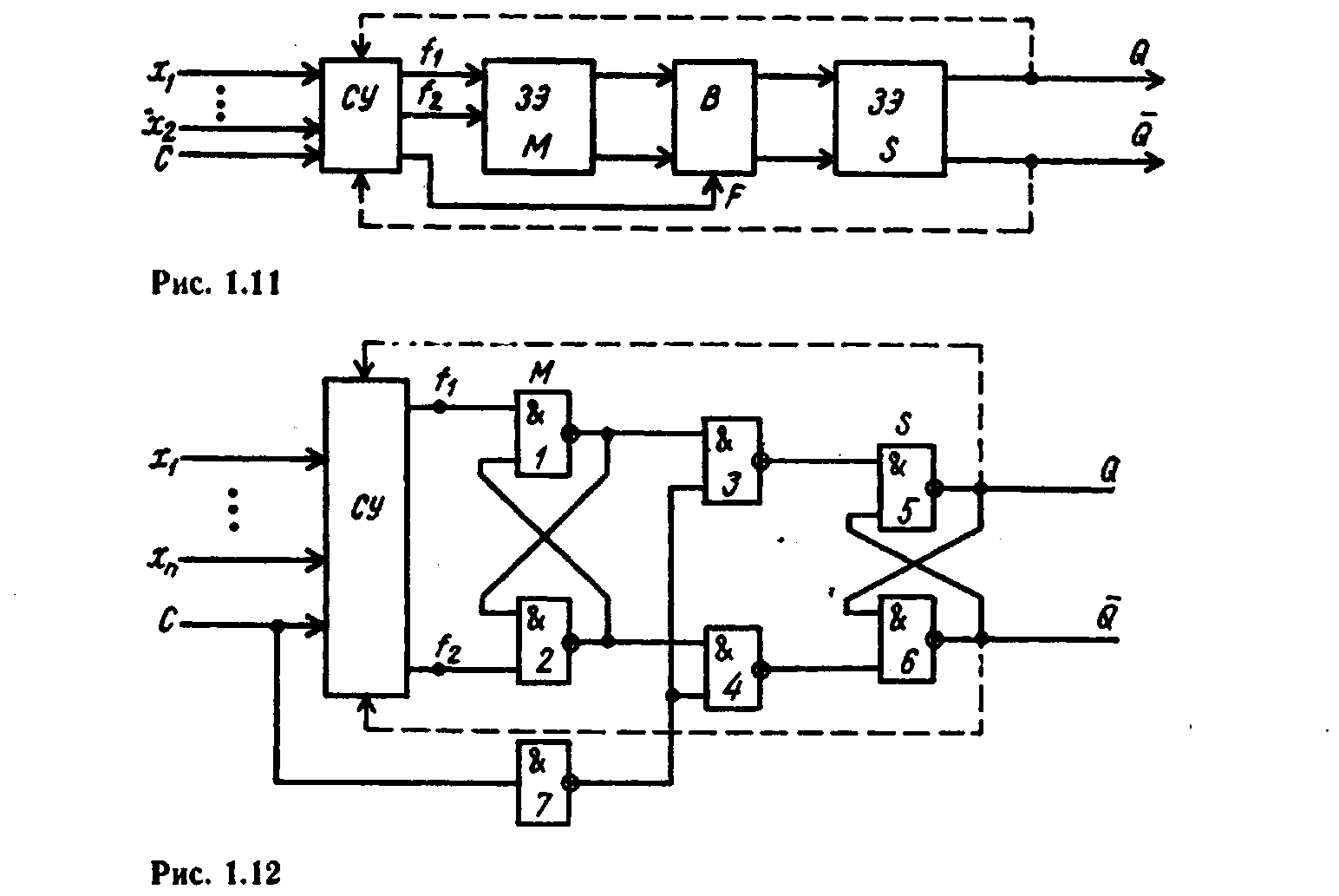
2. Вибір запам’ятовуючого елемента тригера (без внутрішньої затримки і з внутрішньою затримкою).

3. Побудова розширеної (повної) таблиці переходів тригера. Знаходження функцій збудження елементів пам’яті у вигляді мднф.

4. Побудова структурної схеми тригера.

***Проектування (синтез) тригерних пристроїв складається у виборі ЗЕ і синтезі СК в заданому елементному базисі***.

У тригерах, керованих рівнем сигналу, використовуються елементарні запам'ятовуючі елементи на вентилях І-НІ (рис. 1.9, а),

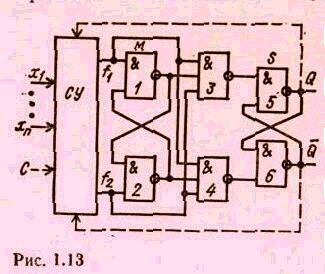


а також АБО-НІ (рис. 1.10,а). Спосіб їхнього переключення описується системами підграфів переходів на рис. 1.9, *б* і рис. 1.10, *б.*

Якщо в стовпці QS+1 таблиці переходів проектованого тригера мається значення  (табл. 6 і 7), то *Q* є аргументом функцій *f*1 і *f*2. Для забезпечення правильного переключення тригера в точках *А* (рис. 1.8) у цьому випадку необхідно включити елементи затримки. Аналогічна ситуація виникає в тому випадку, коли аргументами функцій *f*1 і *f*2 є сигнали *Q* інших тригерів, що переключаються в процесі роботи одночасно з даним тригером. Наприклад, такими «залежними» є тригери в регістрах, що зсувають, лічильниках і т.д.

Однак при побудові тригерів на потенційних елементах не можуть бути використані елементи затримки, що містять реактивні компоненти, тому що це накладало б обмеження на тривалості вхідних сигналів тригера. У потенційній системі елементів повинна бути забезпечена правильна робота тригера при будь-якій тривалості вхідних сигналів, якщо вона перевищує деяке мінімально припустиме значення. У цьому випадку на елементах І-НІ і АБО-НІ тригери будують по *МS-схемі* або за *схемою трьох елементарних тригерів*.

Перший спосіб полягає у використанні двох елементарних ЗЭ: основного М-тригера і допоміжного S-тригера. Структура однотактного МS-тригера пояснюється рис. 1.11. Запис у *М-тригер* тактується сигналом *С*, а в S-тригер - сигналом *F.* Передача інформації з М-тригера в S-тригер здійснюється через вентилі *В.* Найбільше поширення отримали МS-тригери з інвертором у ланцюзі *С* і МS-тригери із забороненими зв'язками. Схеми зазначених тригерів на елементах І-НІ приведені відповідно на рис.1.12 і 1.13, де елементи *1, 2* утворять *М-тригер*, а елементи *5, 6* — *S-тригер*. Вентилями *3, 4* у схемі на рис. 1.12 керують сигнали з виходу інвертора 7, а в схемі на рис. 1.13 - сигнали *f*1 і *f*2. За схемою на рис. 1.13 можуть бути побудовані як синхронні, так і асинхронні тригери. У схемах на рис. 1.12 і 1.13 елементи І-НІ можуть бути замінені на елементи АБО-НІ.

Тригери, виконані по МS-схемі, називають тригерами з потенційним записом інформації. Запис інформації в *М-тригери* здійснюється так само, як і в елементарні ЗЕ (рис. 1.9 і 1.10). Перепис інформації з *М-тригерів* у *S-тригери* здійснюється при знятті активного вектора вхідних сигналів тригера.

Тригер із внутрішньою затримкою, на елементах І-НІ, виконаний за схемою трьох елементарних тригерів, зображений на рис. 1.14. У даному тригері сигнали, що відповідають новому станові, установлюються при переході тактуючого сигналу з 0 у 1. Спосіб запису інформації в тригерах такого типу називають динамічним. При *С* = 0 на виходах вентилів *2* і *3* є присутнім одиничний сигнал, тобто тригер на вентилях *5* і *6* не змінює свого стану, а вентилі 1 і 4виконують функцію інверторів. Переключення тригера здійснюється відповідно до рис. 1.9, *б.*

Наприклад, якщо *f*1 = 0 і *f*2 = 1, то при переході сигналу *С* з 0 у 1 на виході вентиля *2* установлюється нульовий сигнал, що переключає в одиничний стан тригер на вентилях 5, *6* і підтверджує одиничний сигнал на виході вентиля 1. Після цього сигнали *f*1 і *f*2. можуть змінюватися що не вплине на стан основного тригера (вентилі 5 і *6)* доти, поки не здійсниться черговий перехід сигналу *С* з 0 у 1. Розглянутий тригер можна побудувати на вентилях АБО-НІ. У цьому випадку він змінює свої стани відповідно до рис. 1.10, *б.*

Для побудови ЗЕ крім вентилів І-НІ і АБО-НІ використовують і інші вентилі. Наприклад, синхронний тригер із внутрішньою затримкою і динамічним записом інформації можна побудувати за схемою на рис. 1.15, де запам'ятовуючий елемент виконаний на вентилях І-АБО-НІ. Такий тригер переключається по негативному перепаді синхросигнала *С* в відповідності з рис. 1.9, *б.* Для правильної роботи тригера необхідно, щоб функції *f*1 і *f*2. зберігали своє значення після негативного перепаду сигналу *C* не менш 2*t* (*t* - час затримки сигналів на вентилі І-АБО-НІ). Якщо ця умова при синтезі СК автоматично не виконується, то в схему вводяться додаткові елементи (інвертори, повторювачі і т.д.) для забезпечення необхідної затримки.

При синтезі СК для будь-якого типу тригера спочатку на підставі скороченої таблиці переходів будується повна таблиця переходів тригера, у якій відбивають значення *QS* для кожного вектора вхідних сигналів. Потім отримують вираження для *f*1 і *f*2. і реалізують ці функції в заданому елементному базисі.

**Приклад синтезу тригера**

В якості прикладу розглянемо процес проектування синхронного *JK-тригера* на елементах І-НІ.

*Оскільки в табл. 7 присутнє значення* *, то тригер повинен мати внутрішню затримку*. Вибираємо структуру тригера, представлену на рис. 1.12, як одну з можливих. Порядок переходів тригера *М* на вентилях І-НІ визначається системою підграфів на рис. 1.9, *б.*

Для розглянутого тригера отримуємо повну таблицю переходів (табл. 8), побудовану відповідно до табл. 7.

Таблиця 8.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *CS* | *JS* | *KS* | *QS* | *QS*+1 | *f*1 | *f*2 |
| 0 | 0 | 0 | 0 | 0 | 1 | \* |
| 0 | 0 | 0 | 1 | 1 | \* | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | \* |
| 0 | 0 | 1 | 1 | 1 | \* | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | \* |
| 0 | 1 | 0 | 1 | 1 | \* | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | \* |
| 0 | 1 | 1 | 1 | 1 | \* | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | \* |
| 1 | 0 | 0 | 1 | 1 | \* | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | \* |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | \* | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

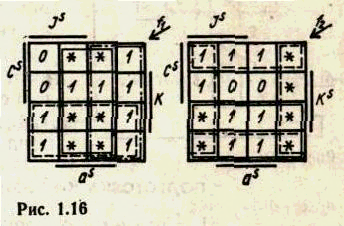
На підставі визначеного вище порядку переходів М-тригера заповнюємо в табл. 8 графи для *f*1 і *f*2, аналізуючи переходи *QS*  *QS*+1 у кожнім рядку таблиці.

За допомогою діаграм Вейча (рис. 1.16) знаходимо мінімальну диз'юнктивну нормальну форму (МДНФ) функцій *f*1 і *f*2,(індекси *S* при цьому опускаємо):

;

.

Приводимо отримані функції до виду, зручному для реалізації на елементах І-НІ:

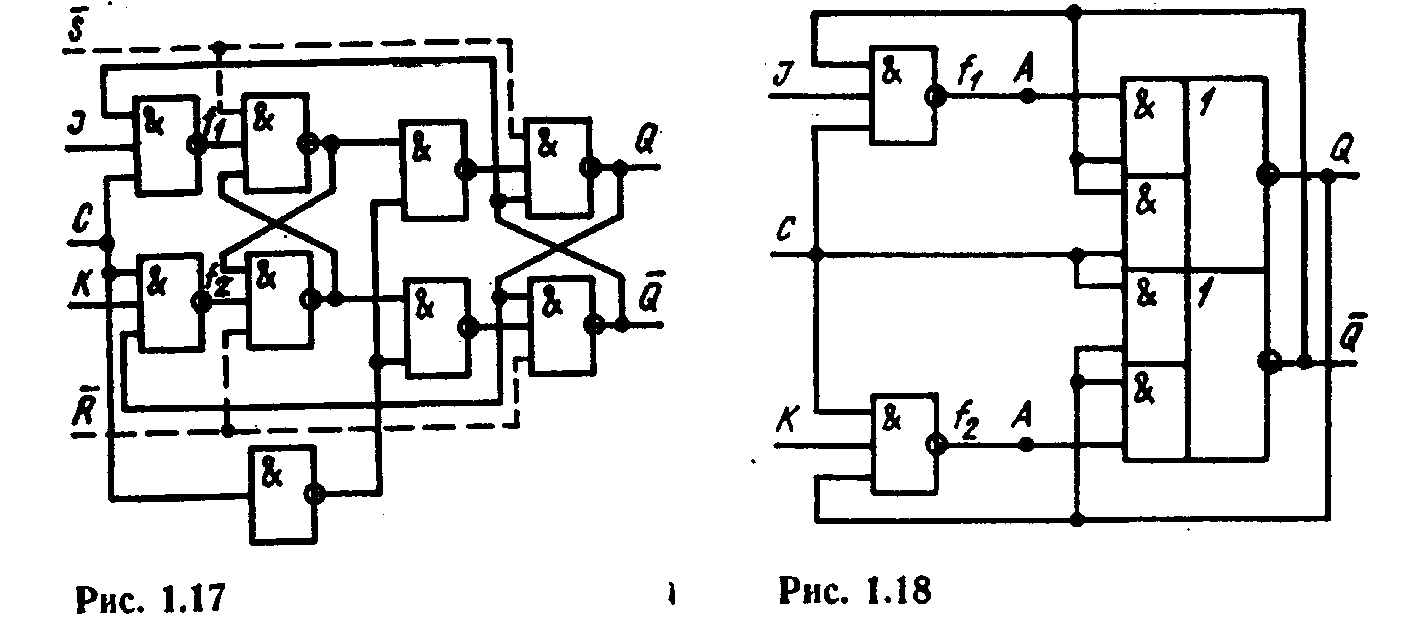
;

.

Отримана схема тригера приведена на рис. 1.17.

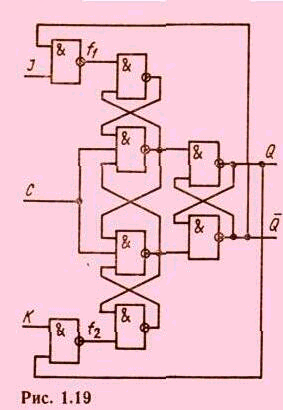
Синтез здійснюється аналогічно, якщо в якості вихідної прийняти схему, приведену на рис. 1.15. Тригер у цьому випадку буде мати вигляд, показаний на рис. 1.18. Для правильної роботи схеми необхідно, щоб затримка сигналів на вентилях І-НІ була, принаймні, удвічі більше затримки сигналів на вентилі І-АБО-НІ. Якщо ця умова не виконується, то в крапки *А* необхідно включити додаткові елементи затримки (наприклад, повторювачі або інвертори).

Якщо в якості базової вибирається схема трьох тригерів (рис. 1.14), то для синтезу СК варто скористатися табл. 9, де відсутнє значення С, тому що відповідний сигнал на СК в тригерах такого типу не надходить. З табл. 9 отримуємо  і . Схема тригера показана на рис. 1.19.



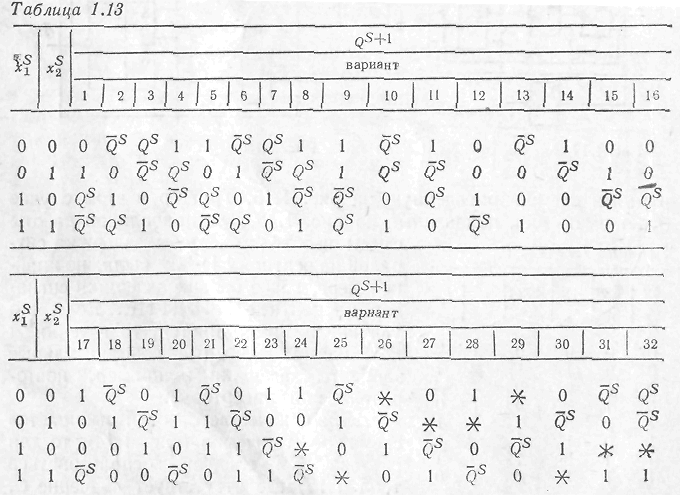
Таблиця 9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *JS* | *KS* | *QS* | *QS*+1 | *f*1 | *f*2 |
| 0 | 0 | 0 | 0 | 1 | \* |
| 0 | 0 | 1 | 1 | \* | 1 |
| 0 | 1 | 0 | 0 | 1 | \* |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | \* | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |

Синхронні тригери можуть мати асинхронні входи попередньої установки тригера в 0 (вхід *R)* і в 1 (вхід *S).* Сигнали, що надходять на ці входи, незалежно від стану інших входів тригера переключають тригер у новий стан, тобто мають пріоритет стосовно інших сигналів. Як приклад організація асинхронних входів показана пунктиром на рис. 1.17.

**ПІДГОТОВКА ДО РОБОТИ**

1. Відповідно до таблиці варіантів (табл. 1.13) побудувати синхронні тригери з використанням для СУ елементів І-НІ.



**ПОРЯДОК ВИКОНАННЯ РОБОТИ**

Синтезувати (згідно з варіантом) схему тригера, налагодити її у відповідній системі та дослідити.

### Зміст звіту

Звіт повинен містити всі викладки, необхідні для виконання лабораторної роботи (синтезу тригера), а також скриншот синтезованої схеми, що демонструє її працездатність (правильність).

Тема (лабораторна робота №3): ПРОЕКТУВАННЯ І ДОСЛІДЖЕННЯ РЕГІСТРІВ

*Мета роботи*: вивчення схемних різновидів регістрів, мікрооперацій, що виконуються на них, і оволодіння методами проектування регістрів.

**Основні теоретичні положення**

*Регістром* називається упорядкована послідовність тригерів, призначена для збереження слів інформації і виконання мікрооперацій над ними.

*Мікрооперація* - це елементарна машинна дія, у результаті якої змінюється значення слова чи здійснюється його пересилання.

Узагальнена структура регістра представлена на рис. 1, де КС - комбінаційна схема; Qi - виходи i-х тригерів (i = 1, …, n); fi - функції збудження тригерів (наприклад, Ji, Ki, Ri, Si і т.д.); С і Т - тактуючі входи, відповідно, тригерів і регістра (у конкретних схемах можуть бути відсутніми). Число розрядів (тригерів) називають *довжиною регістра*. Регістр може знаходитися в 2n станах, тобто в нього можна записати 2n різних слів.

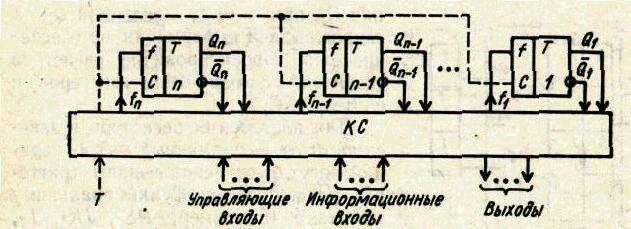


Рис.1 Узагальнена структура регістра.

Розрізняють *синхронні й асинхронні регістри*. У синхронних регістрах мікрооперації виконуються по тактуючому сигналу Т. Настроювання регістра на відповідну мікрооперацію здійснюється попередньою установкою на керуючих входах коду мікрооперації S1...Sm. Число розрядів у коді визначається зі співвідношення m >= |log2k| (або k <= 2m), де k - кількість мікрооперацій, m обирається так, щоб було мінімальним.

В *асинхронних регістрах* тактуючий вхід Т відсутній. Кожна i-а мікрооперація виконується під дією власного керуючого сигналу уi. Кількість таких сигналів дорівнює числу мікрооперацій k, тобто, у загальному випадку, більше, ніж розрядів у коді мікрооперації S1...Sm синхронного регістра.

*При виконанні мікрооперацій у кожному розряді регістра здійснюється однакове перетворення інформації*.

Характер мікрооперації можна описати через стани сигналів у момент часу, що передує виконанню мікрооперації (S), і в момент завершення мікрооперації (S+1).

Найбільше часто на регістрах виконують мікрооперації занесення (прийому, запису) слова паралельним кодом, зсуву слова, а також установки вхідного (як правило, нульового) стану. Занесення слова здійснюється через інформаційні входи Di(i = 1, …, n). Для i-го розряду регістра можна записати QiS+1 = DiS, тобто при виконанні цієї мікрооперації в i-й розряд регістра записується значення сигналу на вході Di.

Регістри, на яких виконуються мікрооперації зсуву, називаються *зсуваючими*. Зсув слова може бути здійснений уліво (у бік старших розрядів) чи вправо (у бік молодших розрядів) на *i* розрядів одночасно, де *i* = 1, …, n-1. Регістри, що мають ланцюги як лівого, так і правого зсуву, називаються *реверсивними*. Зсув слова вліво і вправо, наприклад, на один розряд можна описати відповідно як QiS+1 = QSi-1 і QiS+1 = QSi+1. Установка нульового стану регістра ( QiS+1 = 0) здійснюється, як правило, через асинхронні установчі входи тригерів.

За допомогою регістрів можна виконувати й інші перетворення інформації, зокрема, порозрядні логічні операції: кон’юнкцію (QiS+1 = QiS ⋀DSi); диз'юнкцію (QiS+1 = QiS ˅ DSi); нерівнозначність (QiS+1 = QiS ⊕ DSi); інвертування розрядів і т.д.

Виходами регістра бувають безпосередньо виходи тригерів, але в ряді випадків КС (рис. 1) включає елементи, що здійснюють видачу інформації. Видача слова може бути здійснена в прямому, оберненому чи парафазному коді. Крім того, регістр може мати виходи з трьома станами (стан логічного нуля, логічної одиниці і високоомний стан). Для видачі слова в прямому коді до виходів регістра підключають прямі виходи тригерів Qі, в оберненому коді - інверсні виходи Qі, а в парафазному коді - прямі й інверсні.

Як приклад на рис. 2, а показана організація ланцюгів видачі прямого і оберненого коду на елементах І/АБО-НІ. Для видачі прямого коду подається керуючий сигнал , а для видачі оберненого - . Якщо  =  = 0, то на усіх виходах регістра будуть присутні одиниці. Ланцюга видачі прямого і оберненого коду можна реалізувати і на елементах ВИКЛЮЧАЮЧЕ АБО (рис. 2, б). У цьому випадку при  = 0 з регістра видаються значення Qі, при  = 1 - значення Qі, a заборона видачі інформації не забезпечується. Проектування регістрів зводиться до вибору типу тригерів і синтезу КС.

Для побудови регістрів у залежності від виконуваних мікрооперацій можуть бути використані тригери, різні за функціональною ознакою (наприклад, RS-, JK.-, Т-, D- типу) і по організації (синхронні й асинхронні, із внутрішньою затримкою і без внутрішньої затримки).

# Рис.2

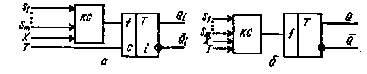
*Зазвичай в регістрах усі розряди синтезуються однаково*. Розряд синхронного регістра (без урахування ланцюгів видачі інформації) може бути побудований як на синхронному (рис. 3, а), так і на асинхронному (рис.3, б) тригері

Рис. 3

Аргументами функцій збудження f є розряди Si коду мікрооперації і множина сигналів X, що містить значення Dі, Qі, Qj (j  i) і т.д., які визначається системою мікрооперацій. Якщо використовується асинхронний тригер, то тактуючий сигнал Т подається на вхід КС.

*Асинхронні регістри* також можуть бути реалізовані з використанням і синхронних, і асинхронних тригерів. *Синхронні тригери найбільш зручні, коли на регістрі виконується одна мікрооперація (наприклад, мікрооперація зсуву). При цьому (рис. 4, а) сигнал мікрооперації Y підключається безпосередньо до входу* С *тригера*. При використанні асинхронних тригерів розряд регістра будується відповідно до рис. 4, б. У цьому випадку сигнали Yi надходять на входи КС.

Синтез функцій збудження тригерів виконують у наступній послідовності:

- складають таблицю переходів i-гo розряду регістра, у якій відображають стан тригера Qі і значення сигналів з множині XS (рис. 3 і 4) у момент часу S, а для моменту часу S+1 – новий стан тригера QіS+1;

## 

## Рис. 4

- відповідно до системи підграфів переходів використовуваного тригера для кожного рядка отриманої таблиці (для кожного переходу QіS  QiS+1) записують необхідні значення функцій збудження тригера;

- виконують синтез КС у заданому елементному базисі.

Якщо аргументами функцій збудження тригерів є значення Qі, то тригери повинні мати внутрішню затримку.

Складність і швидкодія КС при заданому наборі мікрооперацій визначається функціональним типом тригерів. Якщо для побудови регістра можна використовувати тригери різного типу, доцільно одержати кілька варіантів схем, а потім вибрати з них ту, котра має необхідні параметри (функції збудження мінімальні).

Системи підграфів переходів для JK-, RS-, Т- і D-тригерів представлені на рис. 5, де знаком \* відмічені довільні (або 0, або 1) значення функцій збудження.

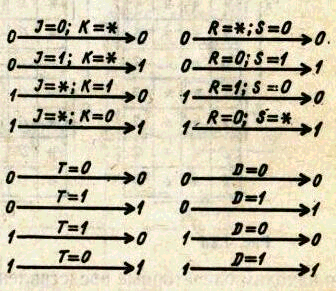


Рис. 5

Нехай необхідно побудувати синхронний регістр для виконання мікрооперацій збереження інформації, паралельного занесення інформації і зсуву слів вправо на один розряд з використанням елементів І-НІ, а також синхронних RS-, JK-, Т-, чи D-тригерів.

Будемо вважати, що значення 00, 01 і 10 коду мікрооперації S2S1 відповідають режиму збереження інформації, паралельного занесення і зсуву. Значення слова S2S1 = 11 є забороненим.

У таблиці переходів i-го розряду регістра (табл. 1) значення QіS+1 при S2S1 = 00 повторюють значення QiS, що відповідає режиму збереження інформації. Для S2S1 = 01 справедлива рівність QiS+1=Di , що відповідає мікрооперації занесення, а при S2S1 = 10 використовується співвідношення QiS+1 = QSi+1, що характеризує зсув інформації в регістрі на один розряд. Значення функцій збудження тригерів Ti, Ji, Ki, Ri, Si, і Di’ (на відміну від інформаційного входу Di регістра, функція збудження D-тригера відмічена штрихом) для кожного переходу QiS  QiS+1 занесені в таблицю відповідно до рис. 5. На підставі табл. 1 одержуємо діаграми Вейча для функцій збудження тригерів (рис. 6) і знаходимо операторні представлення функцій у формі І-НІ/І-НІ (опускаючи індекси S):

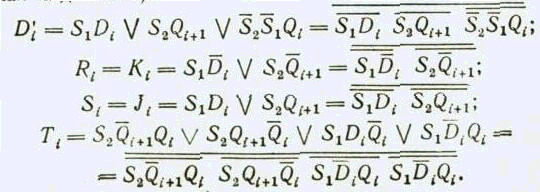




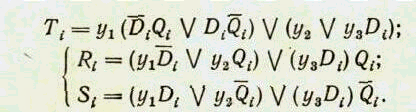
Рис. 6.

Очевидно, що найбільш складною є КС T-тригерів. Для JK- і RS-тригерів складність КС однакова. Найбільш простий є КС при використанні D-тригерів. В усіх випадках аргументами функцій збудження є Qi. Отже, для побудови регістра необхідно використовувати тригери, керовані перепадом тактуючого сигналу (із внутрішньою затримкою).

Схема 4-розрядного регістрана D-тригерах представленанарис. 7, де DR - вхід занесення інформації послідовним кодом при виконанні мікрооперації зсуву.

Аналогічно виконується синтез асинхронних регістрів. Нехай необхідно побудувати асинхронний регістр, використовуючи асинхронні RS- чи T-тригери, двoхвходові елементи І і АБО, а також інвертори. Регістр повинен забезпечувати виконання мікрооперацій y1, y2, y3 де y1 - занесення інформації (QiS+1=Dі); y2 - інвертування розрядів регістра (QiS+1= ) і y3 - нерівнозначність (QiS+1 = Qі  DSi).

Відповідно до табл. 2 і рис. 8 одержуємо операторні представлення функцій:



При цьому складність КС менше при використанні T-тригерів. Помітимо, що при виборі варіанта побудови регістра в разі потреби оцінюється складність не тільки КС, а всієї апаратури, включаючи тригери. Схема одного розряду на T-тригері представлена на рис. 9. Усі розряди регістра будуються аналогічно. Як приклад показана організація виходів регістра з трьома станами. Керування станами здійснюється сигналом а.



Рис. 7.

Часові характеристики регістрів визначають за допомогою часових діаграм. До основних з них відносять мінімальний час виконання мікрооперацій t і максимальну частоту переключення регістра f.

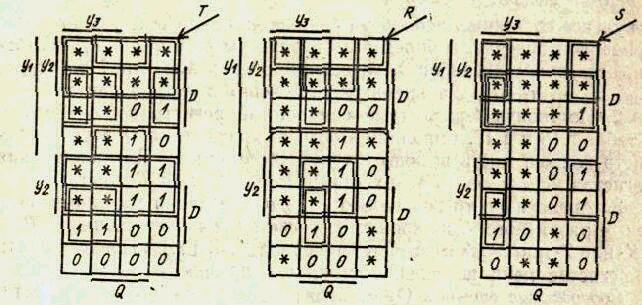


Рис. 8.

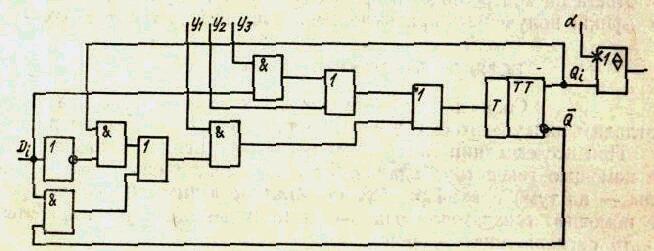


Рис. 9

Таблиця 1.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1S | S2S | DiS | QSi+1 | QiS | QiS+1 | Dі’ | Rі | Sі | Jі | Kі |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | \* | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | \* | \* | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | \* | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | \* | \* | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | \* | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | \* | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | \* |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | \* | \* | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | \* |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | \* | \* | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | \* | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | \* |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | \* | \* | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | \* | 0 | 0 | \* |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | \* | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | \* |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | \* | \* | 0 |
| 1 | 1 | 0 | 0 | 0 | \* | \* | \* | \* | \* | \* |
| 1 | 1 | 0 | 0 | 1 | \* | \* | \* | \* | \* | \* |
| 1 | 1 | 0 | 1 | 0 | \* | \* | \* | \* | \* | \* |
| 1 | 1 | 0 | 1 | 1 | \* | \* | \* | \* | \* | \* |
| 1 | 1 | 1 | 0 | 0 | \* | \* | \* | \* | \* | \* |
| 1 | 1 | 1 | 0 | 1 | \* | \* | \* | \* | \* | \* |
| 1 | 1 | 1 | 1 | 0 | \* | \* | \* | \* | \* | \* |
| 1 | 1 | 1 | 1 | 1 | \* | \* | \* | \* | \* | \* |

Таблиця 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| y1S | y2S | y3S | DSi | QiS | QіS+1 | Tі | Rі | Sі |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | \* | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | \* |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | \* | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | \* |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | \* | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | \* |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | \* | \* | \* | \* |
| 0 | 1 | 1 | 0 | 1 | \* | \* | \* | \* |
| 0 | 1 | 1 | 1 | 0 | \* | \* | \* | \* |
| 0 | 1 | 1 | 1 | 1 | \* | \* | \* | \* |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | \* | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | \* |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | \* | \* | \* | \* |
| 1 | 0 | 1 | 0 | 1 | \* | \* | \* | \* |
| 1 | 0 | 1 | 1 | 0 | \* | \* | \* | \* |
| 1 | 0 | 1 | 1 | 1 | \* | \* | \* | \* |
| 1 | 1 | 0 | 0 | 0 | \* | \* | \* | \* |
| 1 | 1 | 0 | 0 | 1 | \* | \* | \* | \* |
| 1 | 1 | 0 | 1 | 0 | \* | \* | \* | \* |
| 1 | 1 | 0 | 1 | 1 | \* | \* | \* | \* |
| 1 | 1 | 1 | 0 | 0 | \* | \* | \* | \* |
| 1 | 1 | 1 | 0 | 1 | \* | \* | \* | \* |
| 1 | 1 | 1 | 1 | 1 | \* | \* | \* | \* |
| 1 | 1 | 1 | 1 | 1 | \* | \* | \* | \* |

**ПІДГОТОВКА ДО РОБОТИ**

1. Побудувати чотирирозрядні синхронні і асинхронні регістри для виконання мікрооперацій, заданих у табл. 3, де y1 - зсув вліво на один розряд; y2 - зсуву вправо на один розряд; y3 - прийом слова паралельним кодом; y4 - диз'юнкція; y5 - кон’юнкція; y6 - нерівнозначність ( - додавання по модулю два); y7 - рівнозначність ( - еквівалентність); y8 - інвертування розрядів.

Для побудови регістра використовувати логічні елементи, зазначені в табл. 4, а також RS-, JK-, Т- чи D-тригери.

Визначити необхідну організацію тригерів (із внутрішньою чи затримкою без внутрішньої затримки) і побудувати схему регістра з ланцюгами видачі прямого коду. Передбачити установку регістра в кульовий стан, використовуючи асинхронні входи .

Значення i в табл 3 і 4 визначають варіант завдання. Для вибору варіанта необхідно його номер представити в двійковій системі числення і позначити шість молодших розрядіву виглядіслова 654321.

У процесі синтезу попередньо отримати операторні представлення функцій збудження для усіх функціональних типів тригерів, а потім вибрати тригер, що забезпечує мінімальну складність КС.

**Зміст звіту**

Звіт повинен містити короткі теоретичні відомості, необхідні для виконання лабораторної роботи і відповіді на контрольні питання, усі схеми, формули, таблиці і графіки, отримані при виконанні завдання, а також висновки.

#### Таблиця 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 2 | 1 | Мікрооперації | | |
| Номер пункту завдання | | |
| 1 (СС) | 2 (СА) | 3 (АА) |
| 0 | 0 | 0 | y1, y3, y4 | y3, y8 | Y2, y3 |
| 0 | 0 | 1 | y1, y3, y5 | y3, y7 | Y2, y4 |
| 0 | 1 | 0 | y1, y3, y6 | y3, y6 | Y2, y5 |
| 0 | 1 | 1 | y1, y3, y7 | y3, y5 | Y2, y6 |
| 1 | 0 | 0 | y2, y3, y8 | y3, y4 | Y1, y7 |
| 1 | 0 | 1 | y2, y3, y4 | y1, y8 | Y1, y8 |
| 1 | 1 | 0 | y2, y3, y5 | y2, y8 | Y1, y6 |
| 1 | 1 | 1 | y2, y3, y6 | y1, y7 | Y1, y5 |

#### Таблиця 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 5 | 4 | Логічні елементи | | |
| Номер пункту завдання | | |
| 1 | 2 | 3 |
| 0 | 0 | 0 | 2І-НІ, 3І | 3АБО-НІ | 3І, 2АБО, НІ |
| 0 | 0 | 1 | 3І, 2АБО, НІ | 2І-НІ | 3АБО-НІ |
| 0 | 1 | 0 | 3АБО-НІ | 3І, 2АБО, НІ | 2І-НІ |
| 0 | 1 | 1 | 3І-НІ | 2АБО-НІ | 2І, 3АБО, НІ |
| 1 | 0 | 0 | 2І, 3АБО, НІ | 3І-НІ | 2АБО-НІ |
| 1 | 0 | 1 | 2АБО-НІ | 2І, 3АБО-НІ | 3І-НІ |
| 1 | 1 | 0 | 2І-НІ | 2АБО-НІ | 3І, 3АБО, НІ |
| 1 | 1 | 1 | 2І, 2АБО, НІ | 3І-НІ | 3АБО-НІ |

**КОНТРОЛЬНІ ПИТАННЯ**

1. Скільки різних слів можна записати в регістр довжиною 1?

2. Які мікрооперації можна виконувати на регістрах?

3. Намалюйте узагальнену логічну структуру регістра на синхронних і асинхронних тригерах.

4. Від чого залежить складність комбінаційної схеми тригера?

6. Які регістри називаються зсуваючими?

6. Охарактеризуйте, які перетворення інформації здійснюються при виконанні різних мікрооперацій?

7. У яких випадках можна здійснити прийом слова на регістр в однофазному коді й у яких випадках - у паралельному коді?

8. Охарактеризуйте етапи синтезу комбінаційної схеми регістра.

9. Чим відрізняється процес синтезу комбінаційної схеми при використанні для побудови регістра синхронних і асинхронних тригерів?

10. Побудувати регістр на тригерах заданого типу для виконання визначеної мікрооперації.

11. Складіть таблицю переходів для *RS*-, *JK-, Т- i* D-тригерів;.

12. У яких випадках для побудови регістра необхідно використовувати тригери з внутрішньою затримкою?

13. У чому відмінність тригерів із внутрішньою затримкою від тригерів без внутрішньої затримки?

# Тема (лабораторна робота №4): ПРОЕКТУВАННЯ І ДОСЛІДЖЕННЯ ЛІЧИЛЬНИКІВ

Мета роботи: вивчення різних типів лічильників у потенційній елементній базі, оволодіння методами їхнього проектування і налагодження.

##### ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

*Лічильником* називають схему, що призначена для виконання мікрооперації рахунку і збереження слів.

Число дозволених станів лічильників називають його *періодом*, *модулем* чи *коефіцієнтом перерахунку К.*

Лічильники *можуть бути побудовані* на основі рахункових тригерів зі спеціальними міжрозрядними зв'язками, на основі регістрів, що зсувають (кільцеві лічильники) і на основі багатостійких елементів. У даній роботі розглядаються лічильники двох перших типів.

Основними часовими характеристиками лічильників є:

*f -* максимальна частота надходження рахункових сигналів;

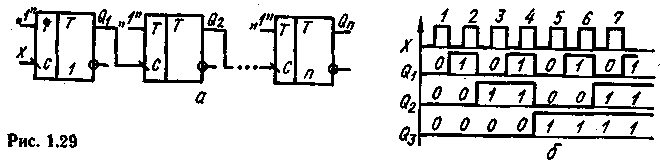
*t -* час переходу лічильника з одного стану в інший.

Лічильники зі спеціальними міжрозрядними зв'язками класифікуються по різних ознаках.

По *характеру мікрооперації рахунку* лічильники підрозділяються на *збільшуючі, зменшуючі і реверсивні*.

При надходженні чергового рахункового сигналу *X* вміст збільшуючого лічильника збільшується на 1, а зменшуючого лічильника - зменшується на 1. Реверсивний лічильник може виконувати як мікрооперацію додавання, так і мікрооперацію віднімання, у залежності від значення сигналу на керуючому вході *Y* (наприклад, при *Y =* 1 виконується додавання, а при *Y=*0 - віднімання).

У *залежності від основи системи числення*, у якій здійснюється мікрооперація рахунку, розрізняють *двійкові* лічильники, *двійково-п’ятиркові, двійково-десяткові* і так далі.



Лічильники класифікуються і по *схемних ознаках*. Для побудови лічильників у потенційній елементній базі застосовуються переважно синхронні тригери з внутрішньою затримкою, що дозволяє використовувати на один розряд двійкового лічильника один тригер.

*По способу організації ланцюгів переносу* (позики) *між розрядами* лічильники підрозділяються на наступні типи: *з послідовним переносом; з наскрізним переносом; з паралельним переносом; із груповим переносом*.

У лічильниках з *послідовним переносом* перенос (позика) у сусідній старший розряд формується тільки після переключення тригера в попередньому розряді, тобто тригери переключаються не одночасно. При проектуванні таких лічильників виникають труднощі, пов'язані з необхідністю аналізу не тільки логічного рівня сигналів, що формуються в схемі, але і моментів зміни рівнів сигналів. На рис. 1.29, *а* представлена функціональна схема n-розрядного збільшуючого лічильника з послідовним переносом, побудованого на синхронних T-тригерах, що переключаються по від’ємному перепаду тактуючого сигналу. Часова діаграма зміни сигналів на виходах Qi (i = без обліку часу переключення тригерів) показана на рис. 1.29, *б.*

У лічильниках з *паралельним переносом* аргументами функцій переносів для кожного розряду є тільки сигнали на виходах тригерів відповідних розрядів. Переноси для всіх розрядів лічильника формуються одночасно (за умови, що всі логічні елементи в схемі мають однаковий час переключення).

Ланцюги *наскрізного переносу* організуються таким чином, щоб функція переносу i-го розряду лічильника була аргументом функції переносу (i + 1)-го розряду. У цьому випадку сигнали переносів для кожного розряду формуються по черзі, починаючи з молодших розрядів лічильника. Лічильники з наскрізним переносом вимагають меншого числа входів логічних елементів для організації ланцюгів переносу, але уступають лічильникам з паралельним переносом у швидкодії.

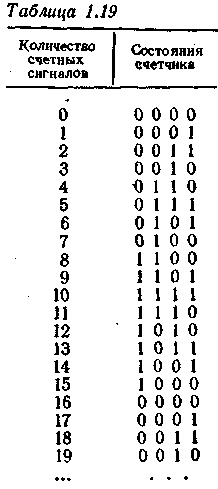
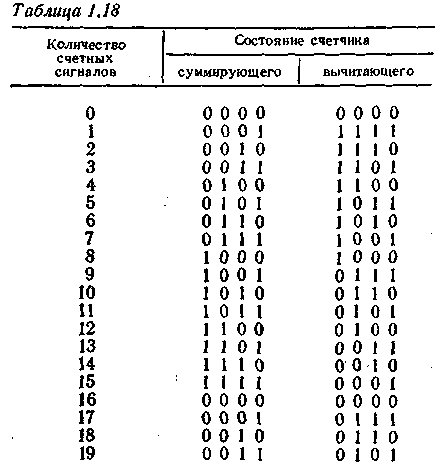
У лічильниках *із груповим переносом* розряди розбиваються на групи (наприклад, *n* розрядів розбиваються на *т* груп). У межах однієї групи, як правило, організується паралельний перенос, а між групами - послідовний чи наскрізний. По такому принципу будуються і лічильники для систем числення з основою K> 2. У цьому випадку роль груп виконують K-ічні розряди.

Якщо мікрооперація рахунку виконується в двійково-кодованій системі числення (двійково-пятірковій, двійково-десятковій і т. д.), то для побудови одного розряду лічильника необхідно не менш *n =* двійкових тригерів. Наприклад, для побудови одного розряду десяткового лічильника потрібно не менш чотирьох двійкових тригерів. Таким чином, один розряд K-ічного лічильника являє собою двійковий лічильник з коефіцієнтом перерахунку *К,* що виконує мікрооперацію рахунку у відповідному коді.

Якщо мікрооперація рахунку виконується в канонічній двійковій системі числення (в однорідній позиційній двійковій системі числення з природним порядком ваг), то такий лічильник називають лічильником *із природним порядком рахунку*. Стани чотирирозрядного лічильника з природним порядком рахунку ілюструються табл. 1.18.

Якщо мікрооперація рахунку виконується в неканонічних системах (наприклад, символічних, зі штучним порядком ваг), то *порядок рахунку вважається штучним*. Стани чотирьохрозрядного лічильника зі штучним порядком рахунку по модулю 2n, що виконує мікрооперацію рахунку в коді Грея, наведені в табл. 1.19.

З табл. 1.18 і 1.19 випливає, що стани лічильників повторюються з періодом 2n. Для скорочення таблиць станів у них заносяться тільки стани для одного періоду.



Лічильники з природним і штучним порядком рахунку можуть мати коефіцієнт перерахунку

Найбільш простими є схеми лічильників із природним порядком рахунку, побудовані на основі тригерів з рахунковим входом *(T-* і JK-тригерів).

На рис. 1.30 представлена узагальнена структура лічильника на T-тригерах (КС - комбінаційна схема, що формує функції збудження Д., що надходять на рахункові входи i-хтригерів). У JK-тригерах рахунковий вхід організується шляхом з'єднання входів *J і К.*

З табл. 1.18 випливає, що переключення тригера молодшого розряду здійснюється з приходом кожного рахункового сигналу, а інших тригерів - тільки в тому випадку, коли всі тригери молодших розрядів встановлені в 1 (збільшуючий) чи в 0 (що віднімає) лічильник.

Отже, для збільшуючих лічильників із природним порядком рахунку по модулю 2n, що має ланцюги паралельного переносу, перемикальні функції fi мають вигляд

 (1)

для лічильників, що віднімають

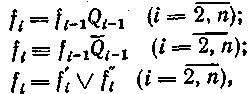
 (2)

а для реверсивних

 (3)

Для всіх типів лічильників f1 = 1.

Для лічильників з наскрізним переносом функції (1), (2) і (3) можуть бути представлені відповідно:

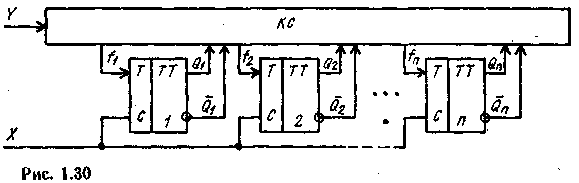


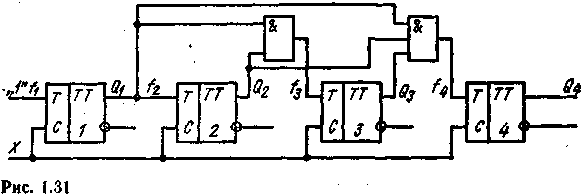
де ; ; ;  ; 

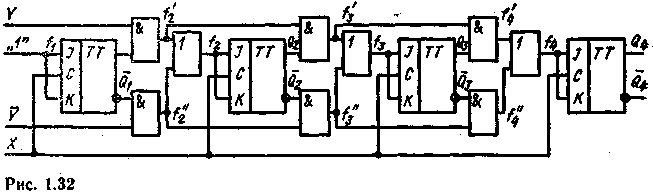
На рис. 1.31 показана функціональна схема збільшуючого лічильника з паралельним переносом на T-тригерах, а на рис. 1.32 - реверсивного лічильника, з наскрізним переносом на JK-тригерах (для *n =* 4).

*Лічильники з будь-яким коефіцієнтом перерахунку К, і будь-яким порядком рахунку можна побудувати в такий спосіб*:

скласти таблицю переходів лічильника (за формою табл. 1.20), записавши у відповідні стовпці таблиці в кожнім рядку коди станів лічильника до надходження чергового рахункового сигналу (S-й момент часу) і після його надходження ((S + 1)-й момент часу). Наприклад, у і-му рядку для S-го моменту часу записати значення сигналів на виходах тригерів у вихідному стані, а для (S *+* 1)-го моменту часу - значення після надходження першого рахункового сигналу;







уміст стовпців QiS+1 0-го рядка перенести в стовпці QiS 2-го рядка, а в стовпці QiS+1 цього рядка записати значення виходів тригерів після надходження чергового (у даному випадку другого) рахункового сигналу і т.д.;

для кожного i-го розряду лічильника записати в кожному j-му рядку таблиці значення сигналів на інформаційних входах fi тригерів (функції №*збудження* тригерів), що забезпечують переключення тригерів зі стану QiS в стан QiS+1;

отримати операторні представлення функцій fiу заданому елементному базисі для всіх типів тригерів, розглядаючи в якості аргументів значення ;

вибрати тип тригера і побудувати схему лічильника.

Нехай необхідно побудувати лічильник, що змінює свій стан у наступній послідовності 000, 001, 010, 011, 110, 111, використовуючи для цього двовходові елементи І та АБО, а також *T-* і JK-тригери.

Складаємотаблицю переходів лічильника (табл. 1.20) і відповідно до рис. 1.24 записуємо для кожного рядка таблиці необхідні значення функцій збудження тригерів. Наприклад, для першого рядка переходи ,  і  мають відповідно вид 01, 00 і 00. Для T-тригерів згідно рис. 1.24 отримуємо Т1=1, Т2=0, Т3=0, а для *JK-тригерів* записуємо в даному рядку *J1* = 1, K*1* = \*, *J2* = 0,K*2* = \*, *J3* = 0 і K*3* = \*.

Розглядаючи як аргументи значення ,  і , отримуємо операторні представлення функцій (опускаючи індекси 5):

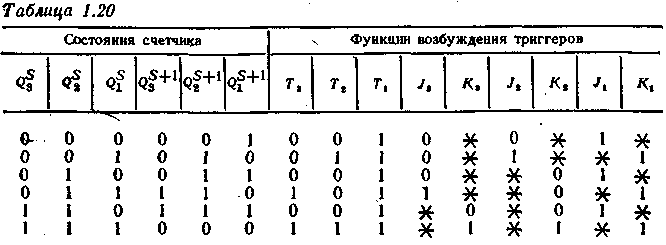


Схема лічильника на T-тригерах ілюструється рис. 1.33, а на JK-тригерах - рис. 1.34. Очевидно, що при використанні JK-тригерів для побудови лічильника потрібно менше логічних елементів.

При великій розрядності лічильників розглянутий метод стає дуже трудомістким, зокрема при мінімізації функцій збудження тригерів.

Для побудови лічильників з коефіцієнтом перерахунку можна скористатися методом виключення надлишкових станів, що рисо критичний до величини *n.*

Найбільше просто виключаються стани, що випливають підряд.



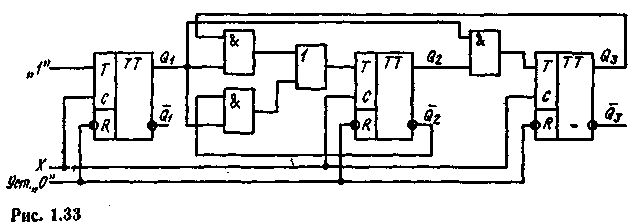
При побудові лічильника необхідно, виконати наступні етапи:

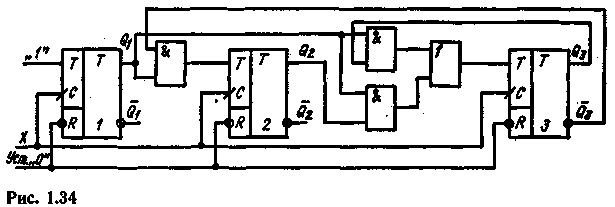
визначити число розрядів n лічильника по формулі *n =* ;

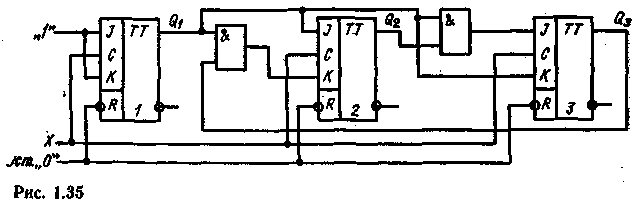
де - функція округлення числа до найближчого більшого цілого, якщо число не ціле;

скласти таблицю станів і отримати функції збудження fi тригерів заданого типу для всіх розрядів лічильника з природним порядком рахунку по модулі 2n;

виключити з таблиці надлишкові стани і позначити через *А* стан, що передує першому з групи виключених,







через *В -* перший в групі виключених, через С - наступний за останнім із групи виключених станів (стан *В* відповідає природному, а стан *С -* штучному переходу зі стану А);

визначити скоректовані функції збудження  для проектованого лічильника за наступним правилом:

а) якщо i-e розряди для станів В и С збігаються, те (корекція функцій збудження не потрібно); 

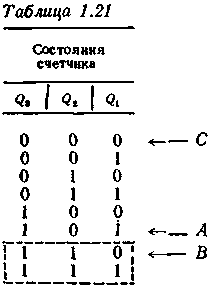
б) якщо i-ий розряд при природному переході в стан *В* переключається, а для штучного переходу в стан *С* переключення не повинне відбуватися, те , де - функція, що приймає одиничне значення на наборі *А* и, можливо, на виключених наборах (здійснюється заборона переключення тригера);

в) якщо необхідно викликати переключення i-го розряду при штучному переході в стан *С,* що не переключається при природному переході в стан *У,* те (Забезпечується примусове переключення тригера);

отримати операторні форми функцій у заданому елементному базисі і побудувати схему лічильника.

Якщо в лічильнику виключаються останні 2n-кстанів (наприклад, у збільшуючому лічильнику рахунок починається з 0 і закінчується числом *К*-1), то такі лічильники є лічильниками з природним порядком рахунку по модулю *К.* При виключенні інших станів природний порядок рахунку порушується. У цьому випадку (наприклад, для збільшуючого лічильника) число рахункових сигналів, що надійшли, не відповідає чисельному еквіваленту коду суми по *модулю* К даної кількості одиниць.

Нехай необхідно побудувати лічильник на T-тригерах з коефіцієнтом перерахунку K = 6. Визначаємо розрядність лічильника по формулі *n =* *=*=3 і будуємо таблицю переходів 3-розрядного лічильника з природним порядком рахунку по модулі 2n = 23 = 8 (табл. 1.21).



Вибираємо паралельний спосіб формування переносів і записуємо функції *Ti* (z = 1, 2, 3) для збільшуючого лічильника з природним порядком рахунку



Виключаємо стани 110 і 111, відмічені в табл. 1.21 пунктиром, і позначимо три стани через *А, В* и С. Відповідно до приведеного вище правилом знаходимо



Оскільки функція на наборах 110 і 111 не визначена, отримуємо  = Q3Q1. З обліком цього знаходимо

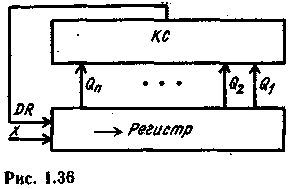
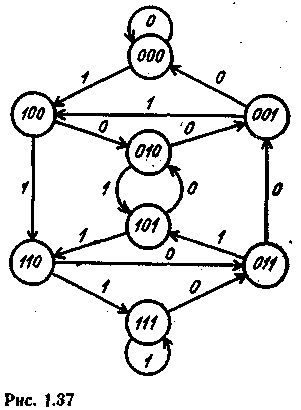


Схема лічильника показана на рис. 1.35. Установка лічильника в нульовий стан здійснюється за допомогою асинхронних входів тригерів.

Кільцеві лічильники будуються на базі зсуваючих регістрів (рис. 1.36). Виходи Qi регістра підключаються до входів комбінаційної схеми (КС), а вихід останньої - до входу *DR* занесення в регістр інформації послідовним кодом. Рахункові сигнали *X* керують зсувом у регістрі.

Кільцеві лічильники можуть забезпечити будь-який заданий коефіцієнт K перерахунку, але не можуть, у загальному випадку, забезпечити довільний порядок зміни станів.

Основна задача проектування таких лічильників складається в синтезі КС, що забезпечує необхідний коефіцієнт перерахунку. У процесі проектування необхідно виконати наступне:

визначити розрядність регістра по формулі *n =*і побудувати граф переходів зсуваючого регістра;

виділити на графі всі циклічні шляхи, що проходять через K вершин (можливі періоди лічильника з коефіцієнтом перерахунку K);

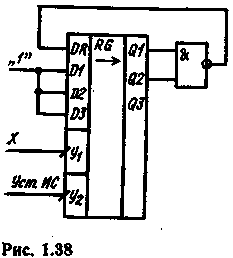
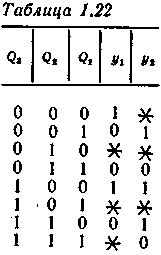
отримати операторні форми перемикальних функцій, реалізованих КС, відповідно до заданого елементного базису для кожного періоду лічильника;

оцінити складність і швидкодія КС для кожної з перемикальних функцій, вибрати КС, що відповідає заданим параметрам, і побудувати схему лічильника.

Розглянемо синтез кільцевого лічильника по модулю 5. Для побудови такого лічильника досить трьохрозрядного зсуваючого регістра, тому що *n* = = =3. Нехай КС повинна бути реалізована на елементах 2І-НІ. Припустимо, що регістр знаходиться в стані 000, тоді в залежності від сигналу на вході *DR* (0 чи 1) після зсуву регістр перейде в стан 100 або залишиться в попередньому стані. Зазначений і наступний переходи можна зобразити у виді графа (рис. 1.37), вершини якого відповідають станам регістра, а дуги - переходам, здійснюваним при 0 чи 1 на вході *DR.*

На графі можна виділити два циклічних шляхи, що проходять через п'ять вершин. Один з них проходить через вершини 000, 100, 110, 011 і 001, а інший = через вершини 111, 011, 001, 100 і 110.

У першому випадку на вході *DR* регістра повинні по черзі бути присутнім значення 1, 1, 0, 0 і 0, а в другому - 0, 0, 1, 1 і 1. Перемикальну функцію, що повинна реалізувати КС, у першому випадку позначимо через f1, а в другому - через f2. У відповідності е таблицею істинності функцій (табл. 1.22) отримуємо операторні форми  і . Функція f2при реалізації на елементах І-НІ забезпечує меншу складність і більшу швидкодію КС. Таким чином, остаточно вибираємо *DR =* f2 = . Схема лічильника на основі асинхронного зсуваючого регістра представлена на рис.1.38.

При позитивному перепаді сигналу на вході *в1* здійснюється зсування вмісту регістра (цей вхід використовується в якості рахункового), а на вході *в2* - паралельне занесення інформації. За допомогою мікрооперації занесення здійснюється установка регістра у початковий стан 111.

**ПІДГОТОВКА ДО РОБОТИ**

1. Використовуючи *JK-, T-* і D-тригери, а також елементи ЗІ-НІ, побудувати лічильник з періодом, зазначеним у табл. 1.23, де  - шість молодших двійкових розрядів номера варіанту завдання. Забезпечити мінімальну складність КС вибором відповідного типу тригера.

2. На T-тригерах і елементах І, АБО і НІ побудувати лічильник по модулю К=31. Період проектованого лічильника отримати, склавши таблицю станів збільшуючого лічильника з природним порядком рахунку по модулю 32 і виключивши з неї стан



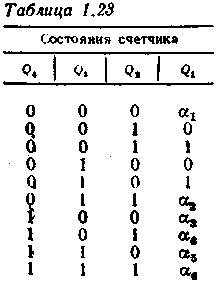
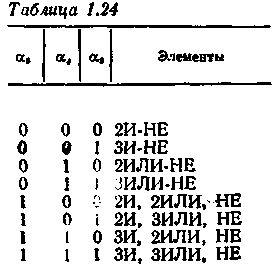
3. Побудувати кільцевий лічильник з коефіцієнтом перерахунку  на базі регістра, показаного на рис. 1.38. Для КС використовувати елементи, зазначені в табл. 1. 24.

### ПОРЯДОК ВИКОНАННЯ РОБОТИ

Зібрати, налагодити і дослідити схем лічильників, отриманих при виконанні завдання.

**Зміст звіту**

Звіт повинен містити короткі теоретичні відомості, необхідні для виконання завдання і відповіді на контрольні питання, усі схеми, формули, таблиці і діаграми, отримані при виконанні лабораторної роботи, а також висновки по роботі.

### КОНТРОЛЬНІ ПИТАННЯ

1. По яких ознаках можна класифікувати лічильники?

2. Напишіть вирази для функцій формування переносів (позик) збільшуючого, що віднімає, реверсивного лічильників з наскрізним і паралельним переносом і природним порядком рахунку по модулі 2n.

3. Як отримати T-тригер на основі RS-, D- і JK-тригерів?

4. Як враховуються набори, що відповідають забороненим станам лічильників, при мінімізації функцій збудження тригерів?

5. Складіть таблицю переходів RS-, JK-, D- і T-тригерів.

6. Як можна установити лічильник у вихідний стан?

7. Охарактеризуйте основні етапи проектування лічильників з довільною зміною станів і з виключенням надлишкових станів, що випливають підряд. Укажіть переваги і недоліки зазначених методів.

8. Який порядок рахунку називається природним, а який штучним?

9. Побудуйте лічильник на RS-, JK-, D- і T-тригерах по заданій таблиці станів.

10. Назвіть основні часові характеристики лічильників.

11. Як забезпечити заданий коефіцієнт перерахунку кільцевого лічильника?

12. Як побудувати часову діаграму роботи лічильника з урахуванням часу затримки сигналів логічними елементами і часу переключення тригерів?

13. Як визначити час переходу лічильника з одного стану в інший і максимальну частоту надходження рахункових сигналів?

14. Яка внутрішня організація тригерів дозволяє використовувати на один розряд двійкового лічильника один тригер?

15. У яких випадках доцільна групова організація ланцюгів переносів у лічильниках?

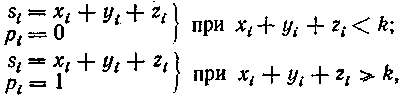
16. Укажіть переваги і недоліки кільцевих лічильників.

# Тема (лабораторна робота №5): ПРОЕКТУВАННЯ І ДОСЛІДЖЕННЯ СУМАТОРІВ

Мета роботи: вивчення методів проектування суматорів, отримання навичок у зборці і налагодженні однорозрядних суматорів на потенційних елементах, оцінка якості досліджуваних суматорів і їхня оптимізація.

**Основні ТЕОРЕТИЧНІ положення**

*Суматор* - операційний вузол, що виконує мікрооперацію арифметичного додавання (підсумовування) двох чисел (слів). Додавання n-розрядних чисел зводиться до виконання порозряднх операцій:



де si - значення суми в i-му розряді; zi - перенос з молодшого розряду; pi - перенос у старший розряд; k - основа системи числення;  порозрядні значення доданків



У *залежності від основи системи числення* і прийнятого кодування k-їчних цифр розрізняють *двійкові, трійкові, десяткові, двійково-десяткові* й інші суматори.

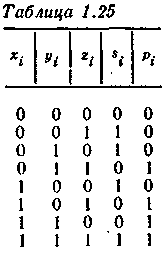
По *способу організації процесу додавання* однорозрядних доданків розрізняють *комбінаційні, накопичуючі і комбінаційно-накопичуючі суматори*.

По *способу обробки багаторозрядних чисел суматори* розділяють *на паралельні, послідовні і послідовно-паралельні*.

*Організація ланцюгів переносу між розрядами* визначає структуру суматора з *послідовним, наскрізним, груповим і одночасним (паралельним) переносом*.

Схемне рішення суматора при його проектуванні залежить від використовуваної системи елементів (серії мікросхем), вимог до тривалості операції додавання і припустимих витрат.

Комбінаційні однорозрядні суматори.Однорозрядним суматором називають перемикальну схему, що за розрядним значенням x*i* і *уi* доданків і за значенням переносу z*i* з молодшого розряду формує значення розрядної суми s*i* і перенос у старший розряд *pi.* Робота такого суматора може бути описана в табл. 1.25 (при *k* = 2).



Мінімальні диз'юнктивні нормальні форми (МДНФ) функцій *si* і *pi* мають вигляд





а їх мінімальні кон’юнктивні нормальні форми (МКНФ) -





Нормальні форми (4)-(7) функцій s*i* і *pi* можуть бути перетворені до операторного вигляду, зручного для реалізації суматора на елементах типу І-НІ, І-АБО-НІ і т.п. Наприклад, при використанні елементів І-АБО-НІ однорозрядний комбінаційний суматор можна побудувати по виразах

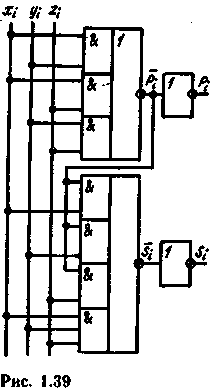




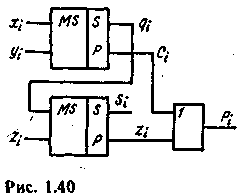
Схема такого суматора приведена на рис. 1.39.

*Напівсуматором* називають комбінаційну схему, що реалізує функції qi суми по mod *k* і переносу *c*iпри додаванні двох змінних, тобто *(k* = 2).





Повний однорозрядний суматор може бути побудований із двох напівсуматорів у відповідності зі схемою на рис. 1.40.



Комбінаційні двійково-десяткові суматори.При побудові двійково-десяткових суматорів (ДДС) десяткові цифри, як правило, кодують за допомогою чотирьохрозрядних двійкових чисел - тетрад.

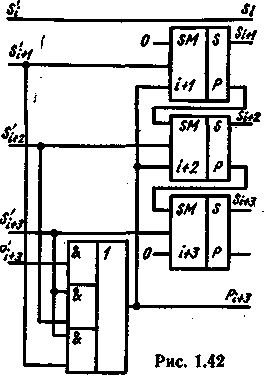
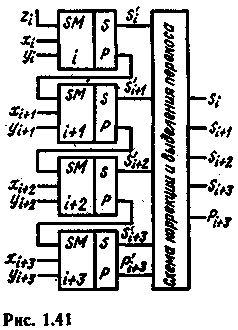
У цьому випадку ДДС повинен реалізувати п'ять перемикальних функцій: чотири з них відповідають двійково-кодованій десятковій сумі - si, si+1*,* si+2, *s*i*+3* і одна - переносу в старший десятковий розряд pi+3*.* Ці функції залежать від десяткових цифр *(x*i*, x*i+1*, x*i+2, *x*i*+3) і* *(y*i*, y*i+1*, y*i+2, yi*+3)* і переносу з молодшої тетради zi*.* Нормальні форми функцій, що реалізуються ДДС, дуже громіздкі і важко мінімізуються. Тому додавання двійково-десяткових кодів (ДДК) виконують у відповідності зі схемою на рис.1.41.

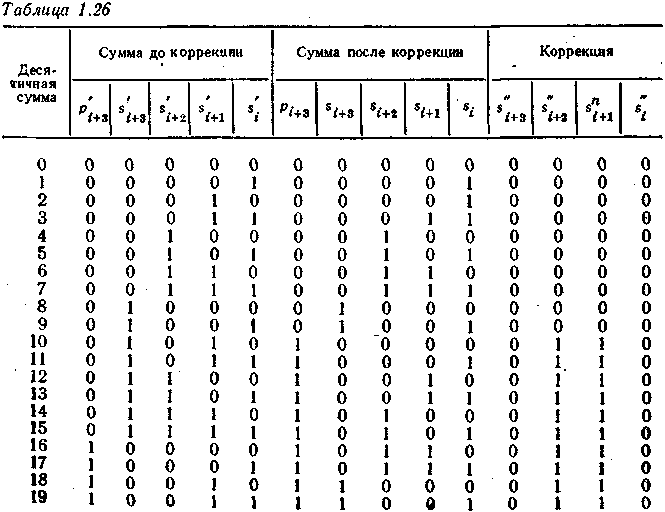
Тут на першому етапі складають ДДК десяткових цифр за правилами двійкової арифметики. Потім на другому етапі за допомогою суматорів роблять корекцію отриманого результату шляхом додавання чи віднімання деякого виправлення, величина якого залежить від некоректованого результату і виду ДДК, а також виділяють десятковий перенос у старшу тетраду.

Використовуваний ДДК повинен мати властивість аддитивності, тобто ДДК суми десяткових цифр повинен дорівнювати сумі ДДК доданків. Такою властивістю володіють, наприклад, ДДК 8, 4, 2, 1 і 8, 4, 2, 1 + , де  - ціле число, що називають *надлишком*. Якщо ж використовуваний ДДК не має властивість аддитивності, то цифри доданків необхідно попередньо (до додавання) перетворити в аддитивний ДДК. Структура схеми корекції і виділення переносу може бути визначена шляхом порівняння слова  , отриманого при додаванні цифр доданків і необхідного результату на виходах схеми корекції (рис. 1.41). Нехай, наприклад, у якості ДДК використовується код 8, 4, 2, 1. Тоді стани виходів суматорів за схемою на рис.1.41 можна описати табл.1.26.

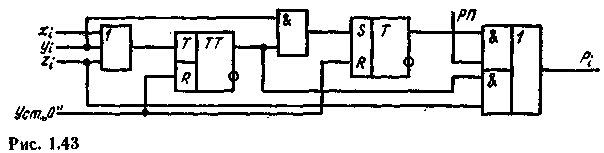
З табл. 1.26 видно, що у залежності від суми, отриманої на першому етапі, корекція результату для ДДК 8, 4, 2, 1 складається в додатку 0 чи 6. Вважаючи функції  і  частково визначеними функціями аргументів , , , , неважко переконатися, що  а 

Схема виділення переносу і корекції показана на рис. 1.42.



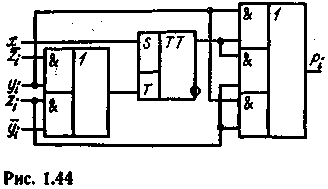


Накопичуючі однорозрядні суматори.Накопичуючим однорозрядним суматором називають схему з пам'яттю, що здійснює арифметичне додавання цифр, що складаються з *x*i*, y*iі переносу ziпри подачі їх на суматор послідовно в часі, незалежно від того, яким кодом (послідовним чи паралельним) представлені слова, що *додаються,* X *і Y*, а також запам'ятовується результат додавання. Такі суматори будують на основі тригерів T-типу. Одна зі схем однорозрядного накопичуючого суматора представлена на рис.1.43. Тут на тригері 1 спочатку формується сума , а потім .



Перенос при додаванні *x*iі *у*i формується елементом 4 і запам'ятовується тригером 2 до подачі зовнішнього сигналу дозволу переносу (РП), синхронного із сигналом zi (відмінність між zi і РП у тім, що, коли РП = 1, zi може бути дорівнює і 0, і 1). Перенос при додаванні *q*iі zi формується елементом 7. Час додавання на такому суматорі дорівнює трьом тактам. Вхід xi може бути відсутнім. При цьому цифри доданків надходять послідовно на вхід *у*i

У комбінаційно-накопичуючому суматорі (рис.1.44) сигнал переносу *p*iформується по МДНФ цієї функції, а сума si - на тригері T-типу. На вхід цього тригера подають сигнал, дорівнює сумі по mod 2 *y*iі *z*iі формований комбінаційною схемою. Час додавання тут складає два такти.



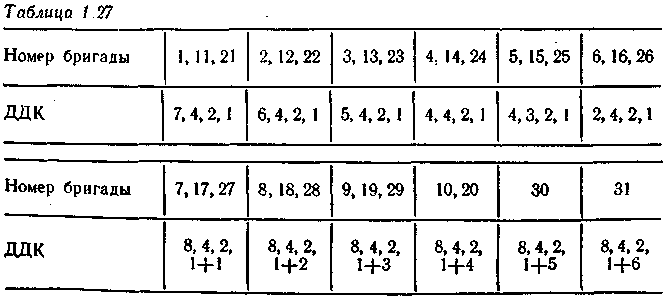
### ПІДГОТОВКА ДО РОБОТИ

1. Побудувати операторні форми функцій, що реалізуються однорозрядним повним суматором на елементах І-НІ, АБО-НІ, АБО-І-НІ, АБО-І-НІ. Відповідні схеми замалювати в протоколі, а також оцінити швидкодію і складність (у числі елементів і по Квайну) отриманих схем.

2. Використовуючи елементи, що є в макеті, побудувати і замалювати до протоколу схему одного розряду двійково-десяткового суматора. Використовуваний ДДК визначається номером бригади і зазначений у табл.1.27.

3. Якщо ДДК не має властивість аддитивності, то необхідно також побудувати схеми перетворювачів заданого ДДК в аддитивний ДДК.

4. У відповідності знаявними в завданні тригерами і логічними елементами побудувати і замалювати до протоколу схеми однорозрядного накопичуючого і комбінаційно-накопичуючого суматорів..



### ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Зібрати і налагодити однорозрядний комбінаційний суматор і напівсуматор на елементах І-НІ по схемах, отриманим при виконанні п. 1 теоретичного завдання.

2. Зібрати і налагодити заданий варіант двоїчно-десяткового суматора по схемах, отриманим при виконанні п. 2.

3. Зібрати і налагодити однорозрядний накопичуючий і комбінаційно-накопичуючий суматори по схемах, що були отримані при виконанні п. 3.

**Зміст звіту**

Звіт повинен містити короткі теоретичні відомості, необхідні для виконання завдання і відповіді на контрольні питання, усі схеми, формули, таблиці і діаграми, отримані при підготовці і виконанні лабораторної роботи, а також висновки по роботі.

### КОНТРОЛЬНІ ПИТАННЯ

1. По яких ознаках класифікують суматори?

2. Чому дорівнює мінімальна складність (у числі елементів і по Квайну) однорозрядного комбінаційного суматора і напівсуматора?

3. Яким вимогам повинні задовольняти ДДК, використовувані в ДДС?

4. У чому сутність властивості аддитивності ДДК і до чого може привести відсутність такої властивості в ДДК?

5. Приведіть приклади ДДК, що володіють і не володіють властивістю аддитивності.

6. У скільки разів число заборонених вхідних наборів для ДДК більше числа дозволених наборів?

7. Скільки розрядів повинен мати накопичуючий суматор, якщо на ньому послідовно підсумовують *М* чисел довжиною в n розрядів?

8. Чи можна змінювати місцями позначення входів (тобто xi і *y*i*,* xiі zi і т.д.) у комбінаційному суматорі? У накопичуючому? У комбінаційно-накопичуючому?

# Тема (лабораторна робота №6): ПРОЕКТУВАННЯ І ДОСЛІДЖЕННЯ КЕРУЮЧИХ АВТОМАТІВ

Мета роботи: вивчення методів синтезу керуючих автоматів із твердою логікою, отримання навичок у їхньому налагодженні й експериментальному дослідженні.

**ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ**

Арифметико-логические устройства (АЛУ) предназначены для выполнения операций над машинными словами.

Машинные слова задаются перечислением их разрядных значений (разрядов) в двоичном представлении или описанием, которое состоит из идентификатора (последовательности букв и цифр, начинающейся с буквы), справа от которого в скобках указываются номера старшего и младшего разрядов слова. Например, слова можно задать следующим образом:  и т. д.

Если разрядность (длина ) слова предварительно оговорена, то оно может быть представлено одним идентификатором, т. е. в виде  и т. д.

Поле (часть слова) обозначается идентификатором слова с указанием номера старшего и младшего из разрядов, принадлежащих полю. Например, словоможно разделить на поля



Слово может быть определено последовательностью описаний полей и разрядов, отделенных друг от друга точками. При этом любое поле может быть представлено двоичными цифрами. Возможны, например, следующие записи: и т. д. Такие слова называются составными.

Выполнение какой-либо операции в АЛУ сводится к выполнению последовательности микроопераций.

Микрооперацией называется элементарная операция, соответствующая элементарному машинному действию, в результате которого могут изменяться значения слов

Наиболее часто при выполнении арифметических операций используются микрооперации пересылки, инвертирования, сдвига суммирования и счета.

Микрооперация пересылки применяется для передачи слова (операнда) из одного узла (регистра, счетчика и т. д.) в другой и записывается с помощью оператора присваивания. Например, микрооперации пересылки могут быть представлены так:

 - передача слова из регистра *Р2* в регистр *Р1*

 - запись в регистр *РА* составного слова:

 - запись в счетчик *СТ* двоичного числа 1011;

 - передача 8-разрядного поля регистра  в старшие разряды регистра  Микрооперация инвертирования заключается в замене единичных значений разрядов слова на нулевые, а нулевых - на единичные. Например, могут иметь место такие микрооперации:  - инвертирование всех разрядов регистра *Р;*

 - образование обратного кода основных разрядов регистра *РА* (знаковые разряды не инвертируются).

Инвертирование может осуществляться в процессе пересылки слов либо с помощью специальных цепей, входящих в состав регистра.

Микрооперация сдвига слова выполняется на регистре (счетчике), имеющем цепи сдвига. Слово *X,* сдвинутое на *i* разрядов вправо, обозначается как а влево - как'.Для уточнения порядка заполнения освобождающихся разрядов используются составные слова. Микрооперации сдвига можно записать, например, в виде

 - сдвиг слова в регистре *Р1* на один разряд вправо;  - циклический сдвиг на один разряд вправо;

 - сдвиг на два разряда влево с записью единиц в освобождающиеся разряды. Сдвиг на различное число разрядов можно также выполнять в процессе пересылки слов с использованием сдвигателя.

Микпоопеоаиия суммирования записывается в виде операторов: - суммирование прямых кодов слов, находящихся  в регистрах *I*(результат, формирующийся на сумматоре, помещается в *Р1);*

 -суммирование прямого и обратного кодов *Р1* и *HJ,* дополненных знаковыми разрядами:

 - суммирование прямого кода слова, находящегося в *РЛ* с дополнительным кодом слова, записанного *PY* (единица подается на вход переноса младшего разряда сумматора).

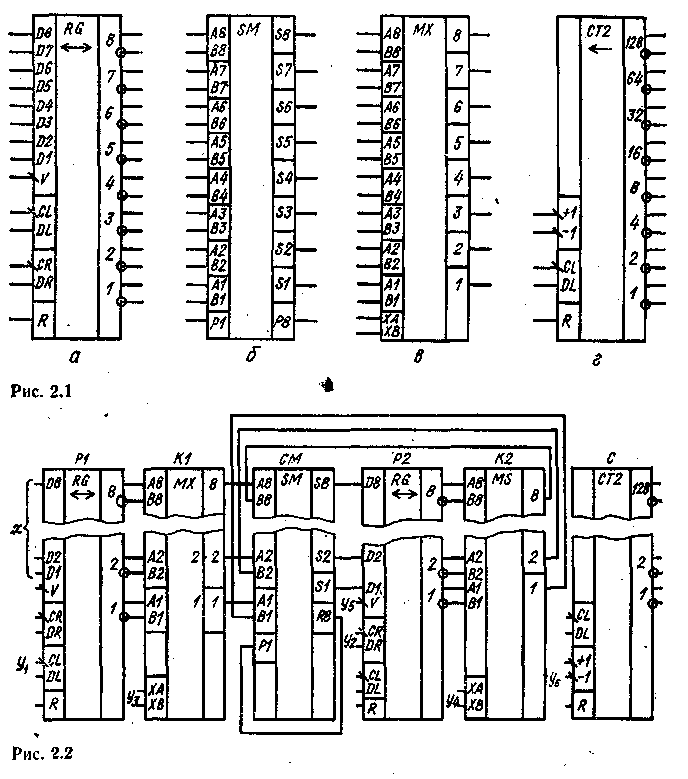
Микрооперация счета состоит в увеличении или уменьшении содержимого счетчика на единицу. Для этой микрооперации используются записи вида

Слева от микроопераций могут присутствовать метки (идентификаторы микроопераций). Метки и микрооперации разделяются двоеточием, например,и т. д.

Способ выполнения микроопераций существенно зависит от структуры операционного устройства и используемой элементной базы.

Лабораторный макет содержит четыре реверсивных сдвигающих регистра (рис. 2.1, *а),* два комбинационных сумматора (рис. 2.1, *б),* четыре коммутатора (рис. 2.1, *в)* и два счетчика (рис. 2.1, *е),* имеющих цепи сдвига слова на один разряд влево (в сторону старших разрядов).

Запись слова на регистр с информационных входов осуществляется по заднему фронту (отрицательному перепаду) сигнала на входе. Сдвиг влево и вправо также производится по заднему фронту сигналов соответственно на входах *CL* и *CR.* При сдвиге слова освободившиеся разряды регистра заполняются цифрами, поступающими последовательным кодом на входы *DL* или *DR* (в зависимости от направления сдвига). Установка нулевого состояния регистра осуществляется единичным уровнем сигнала на входе *R.*



Прибавление и вычитание единицы на счетчике осуществляется по заднему фронту сигналов соответственно на входах +1 и -1.

Сдвиг и установка нулевого состояния производится так же, как и на регистре.

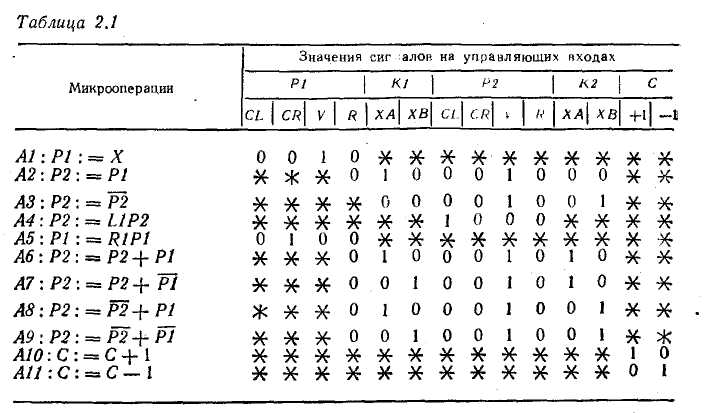
Кроме того, макет содержит четырехвходовый дешифратор, синхронные *D-* и JK-триггеры (срабатывающие по заднему фронту тактирующего сигнала) с разделенными ивходами, логические элементы И-НЕ (четырех- и восьмивходовые), а также элементы индикации (сигнальные ячейки) и управления (кнопки, переключатели).

Сборка схемы осуществляется с помощью внешних проводников. В макете предусмотрена возможность установки регистров и счетчиков в исходное состояние, не используя внешнюю коммутацию.

Выполнение микроопераций рассмотрим на примере конкретного операционного устройства (рис. 2.2).

Значения сигналов, которые необходимо подать на управляющие входы узловпри выполнении некоторых микроопераций, показаны в табл. 2.1. Регистры и счетчик изменяют свое состояние после снятия единичных управляющих сигналов со входов(в процессе выполнения микроопераций управляющий сигнал для каждого узла может подаваться только на один из указанных входов). Знаком *\** втабл. 1.2 обозначены произвольные значения сигналов (0 или 1), которые не влияют на выполнение заданной микрооперации. Благодаря этому некоторые микрооперации можно выполнять одновременно. Например, можно совмещать микрооперацию *А1* с любой другой, кроме *А5.* Нельзя выполнять одновременно микрооперации *А10* и *А11,* но каждая из них может быть совмещена с другими микрооперациями.

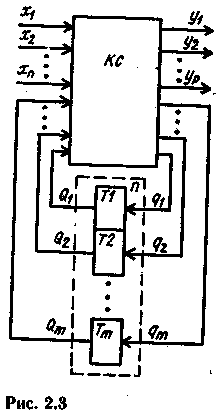
Время выполнения микроопераций зависит от пути прохождения сигналов в схеме. Требуемая длительность управляющих сигналов определяется с помощью временной диаграммы, построенной с учетом задержек сигналов в элементах и узлах операционного устройства. Обычно наиболее длительной является микрооперация суммирования.



##### ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Послідовність мікрооперацій, виконання яких приводить до виконання заданої операції, називають *мікроалгоритмом* *цієї операції*.

Мікроалгоритм може бути реалізований за допомогою *керуючого автомата*, узагальнена структура якого, показана на рис.2.3. Автомат містить комбінаційну схему *(*КС)і пам’ять (П), що складається з тригерів *T* (елементарних автоматів). Входами КС є виходи Q1,...*,* Q*m* тригерів і вхідні сигнали (логічні умови) x1,..., *xk*, щоформуються в операційному пристрої. КСвиробляє керуючі сигнали *у1,*..., *у*р для операційного пристрою і функції збудження тригерів *q1,..., qm,* що визначають перехід автомата з одного стану в інше. Кожному з множини станів {*а1,*..., *ат*}відповідає визначений набір значень QL.



Якщо вихідні сигнали залежать тільки від стану, у якому знаходиться автомат, то його називають автоматом Мура. Закон функціонування такого автомата визначається виразами

 , ,

де S = 0, 1, 2,... - моменти автоматмікроаного (дискретного) часу;  - функція переходів; - функція виходів; а{a1, …, an}, *а0* = *а1, х* = *(х1, х2,... xk)* - вектор значень вхідних сигналів;

*y =* *(y1, y2,*..., *yp)* - вектор вихідних сигналів автомата.

Автомат, вихідні сигнали якого залежать як від стану, так і від вхідних сигналів, називають автоматом Мілі. Його функціонування визначається виразами

 ;

.

Вхідними даними для синтезу автомата є схема операційного пристрою і змістовний мікроалгоритм операції, що включає опису мікрооперацій і логічних умов. Етапи побудови принципової схеми і змістовного мікроалгоритму є взаємозалежними. Процес синтезу схеми автомата включає наступні етапи:

1. Складання списку керуючих сигналів, що забезпечують виконання кожної мікрооперації.

2. Визначення тривалості кожного керуючого сигналу (у числі тактів) і періоду тактирующих сигналів автомата.

3. Отримання закодованого мікроалгоритму.

4. Відмітка станів автомата.

5. Створення графа переходів-видів автомата.

6. Кодування станів автомата.

7. Створення структурної таблиці автомата.

8. Отримання МДНФ функцій збудження тригерів і керуючих сигналів.

9. Представлення функцій збудження тригерів і керуючих сигналів в операторній формі.

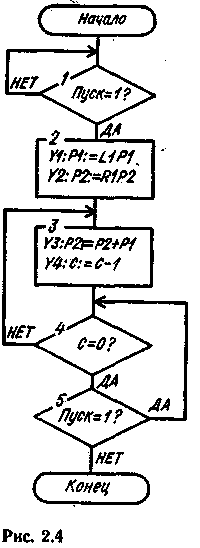
10. Побудова схеми керуючого автомата.

Розглянемо приклад синтезу автомата Мілі для керування виконанням операції

*D* = 2A2 + 0,5В на операційному пристрої (рис. 2.2).

Змістовний мікроалгоритм виконання операції представлений на рис. 2.4. У вихідному стані операнд *В* записаний у *Р2,* а операнд *А* - у *Р1* і С. У першому такті шляхом зсуву слів здійснюється подвоєння *А* и *Р1* і розподіл B на 2 у *Р2.* Далі до вмісту *Р2 А* раз додається слово, записане в *P1.* Після кожного додатка вміст *С* зменшується на 1. Обчислення закінчуються при виконанні умови *С* = 0. Відповідний інформаційний сигнал можна отримати, наприклад, дешифруванням нульового стану С. Результат операції формується в *Р2.*

Для виконання мікрооперацій *Yl, Y2, YЗ* і *Y4* (рис. 2.4) необхідно використовувати керуючі входи операційного пристрою, позначені на рис. 2.2 через *y1, y2, y3, y4, y5* і *y6.* На всі інші керуючі входи повинний подаватися нульовий потенціал.

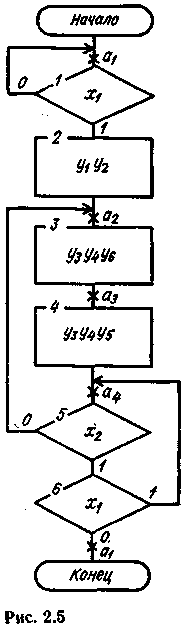


Мікрооперації *Y1, Y2* і *Y4* керуються відповідно сигналами *y1, y2* і *y6.* Для виконання мікрооперації *Y3* необхідно подати три сигнали: *y3, y4, y5.*

Необхідна тривалість керуючих сигналів визначається за допомогою часової діаграми, побудованої з урахуванням затримок сигналів в елементах операційного пристрою. Період  тактуючих сигналів, як правило, вибирається або рівним максимальній тривалості керуючих сигналів, або мінімальній. При цьому величина  повинна бути не менше часу переключення автомата з одного стану в інший. У першому випадку всі мікрооперації виконуються в синхронному режимі (за однаковий проміжок часу), а в другому - в асинхронному, причому тривалості керуючих сигналів кратні величині . Асинхронний режим можна забезпечити, наприклад, введенням у мікроалгоритм додаткових операторних вершин з керуючими сигналами, тривалості яких перевищують .

Будемо вважати, що з урахуванням швидкодії елементів для розглянутого приклада керуючі сигналів *y3* і y4 повинні мати тривалість 2, а інші - . Результати виконання перших двох етапів відбиті в табл. 2.2.

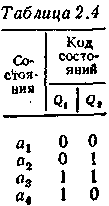
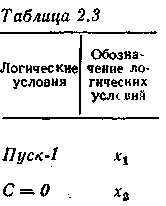
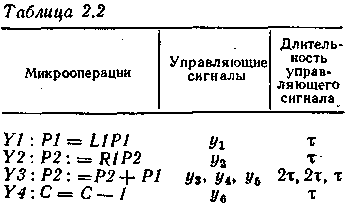
Для отримання закодованого мікроалгоритму складаємо таблицю позначень логічних умов (табл. 2.3) і заміняємо в змістовному мікроалгоритмі (рис. 2.4) опису логічних умов їх позначеннями, а опису мікрооперацій відповідними керуючими сигналами (рис. 2. 5). Оскільки керуючі сигнали *y3* і *y*4, що відповідають мікрооперації *Y3,* записаній у вершині 3 на рис. 2.4, повинні мати тривалість 2, то на закодованому мікроалгоритмі (рис. 2.5) вводимо додаткову операторну вершину (4) з цими керуючими сигналами.



Відмітка станів автомата Мілі здійснюється в такий спосіб:

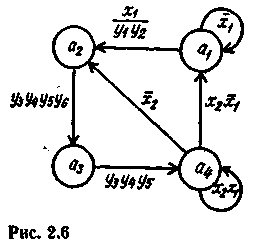
символом a1 відмічається вхід вершини (логічної чи операторної), що слідує за початковою, а також вхід кінцевої вершини;

входи усіх вершин, що слідують за операторними, повинні бути відмічені різними символами аj.



Крім станів, визначених таким чином, може виникнути необхідність уведення додаткових станів. Наприклад, якщо мікроалгоритм має петлю, що охоплює операторную вершину, із сигналом, що змінює стан вузла в момент свого перепаду, чи кілька вершин, що слідують підряд, то для забезпечення таких перепадів при кожному черговому виконанні мікрооперації необхідно вводити додаткові стани. У цих станах автомат не повинен виробляти керуючі сигнали. Додаткові стани можуть знадобитися для забезпечення протигоночного кодування (див. нижче).

Мікроалгоритм на рис. 2.5 відмічений чотирма різними станами (а1, а2, а3, а4).



Граф розглянутого автомата представлений на рис. 2.6. Число вершин графа дорівнює числу станів аj автомата. Кожному переходу автомата з одного стану в інший відповідає дуга графа. Дузі приписується набір логічних умов, при якому здійснюється перехід автомата з одного стану в інший, а також набір керуючих сигналів, що відповідають даному переходу.

Кількість m тригерів, необхідних для організації пам'яті автомата (рис. 2.3), визначається з умови , де *М* - число станів автомата. Кожному стану *аj* повинна відповідати одна визначена комбінація значень Q1, *...,* Q*m.* Для розглянутого приклада вибираємо коди станів відповідно до табл. 2.4. Для організації пам'яті будемо використовувати *JK-*тригери*.*

Помітимо, що спосіб кодування впливає на правильність формування керуючих сигналів і складність автомата. Можливість формування сигналів, не передбачених графом автомата при неоптимальному кодуванні станів, обумовлена появою «гонок», що зв'язано з розкидом часу переключення окремих тригерів автомата. Наприклад, у розглянутому випадку при переході автомата зі стану а1 (10) у стан а2 (01) під час переключення тригерів можлива поява станів 00 чи 11 (у залежності від того, який із тригерів раніше переключиться). Ці проміжні стани при використанні тригерів із внутрішньою затримкою не впливають на правильність переключення автомата, однак можуть привести до появи помилкових керуючих сигналів (наприклад, *у1* і *у*2 при виконанні умови *x1=*1). Для усунення цього недоліку можна використовувати протигоночне кодування, одним з варіантів якого є сусіднє кодування. При сусіднім кодуванні перехід автомата з одного в будь-яке інше припустиме для даного автомата стан здійснюється переключенням тільки одного тригера, унаслідок чого «гонки» не виникають. В автоматах, що не допускають сусіднього кодування, необхідно вводити додаткові стани.

Структурна таблиця автомата будується по його графу. Кожен рядок (табл. 2.5) відповідає визначеному переходу автомата з одного стану в інший. У ній записують поточний стан, стан переходу, коди цих станів, значення логічних умов, що забезпечують перехід, відповідні значення керуючих сигналів і функцій збудження тригерів. Значення функцій збудження визначаються відповідно до таблиці переходів тригера відповідного типу. У кожному рядку для i-го тригера розглядаються переходи . Довільні значення (0 чи 1) сигналів позначаються в таблиці знаком \*.

На основі структурної таблиці автомата визначаємо МДНФ функцій збудження і керуючих сигналів. Аргументами перемикальних функцій *Ji, Ki* і *уi* є значення Q1, Q2, x1 і *x2.* Використовуючи діаграми Вейча (рис. 2.7), отримуємо

;

;

;

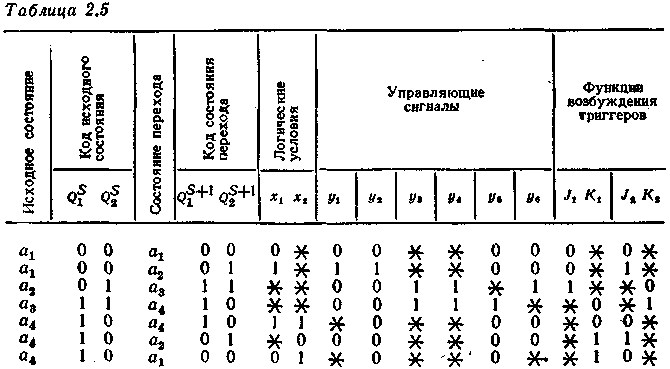




Крім того, з табл. 2.5 випливає, що



Функціональна схема автомата представлена на рис. 2.8, де *УИС -* установка вихідного стану і *ТС* - тактуючі сигнали.



Відмінність синтезу автомата Мура складається в способі отримання відміченого мікроалгоритму і графа автомата.

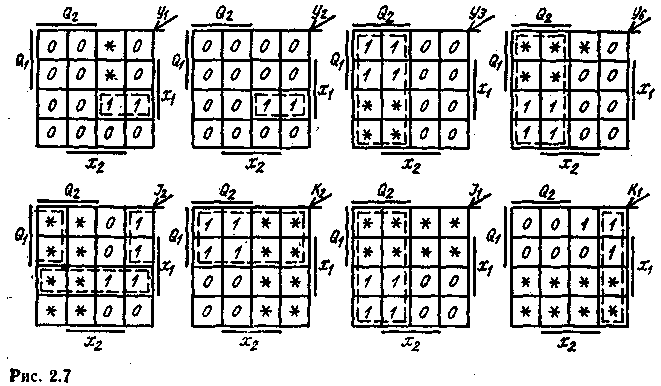
Відмітка станів автомата в цьому випадку здійснюється так:

символом a1 відмічаються початкова і кінцева вершини;

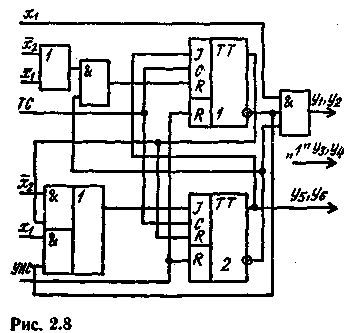
всі операторні вершини відмічаються різними символами *аj.*

У даному випадку, як і при відміткі станів автомата Мілі, може виникнути необхідність уведення додаткових станів.

На графі автомата Мура дугам приписують тільки набори логічних умов, що забезпечують відповідний перехід автомата. Керуючі сигнали записують у вершинах графа, тому що вони не залежать від логічних умов. Відмічений мікроалгоритм і граф автомата Мура для розглянутого приклада представлені відповідно на рис, 2.9 і 2.10.



Кодування станів автомата Мура можна виконувати так само, як і для автомата Мілі. Однак при відповідному кодуванні керуючі сигнали можна знімати і безпосередньо з виходів тригерів автомата Мура (комбінаційна схема для формування функцій yjне потрібна). При цьому число тригерів повинне бути не менше числа керуючих сигналів, що не повторюють один одного і не є константними. У процесі переключення автомата Мура в цьому випадку відповідний рівень керуючих сигналів установлюється швидше, ніж в автомата Мілі. Крім того, при будь-якому переході виключається можливість короткочасного формування помилкових керуючих сигналів. Приклад такого кодування станів автомата, що відповідає графу на рис. 2.8, приведений у табл. 2.6.



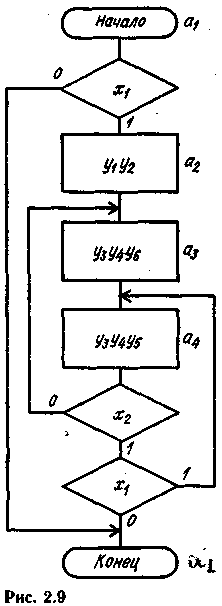
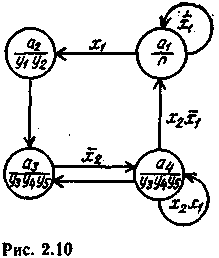
У даному випадку автомат має три тригери, причому 1



Для формування константних керуючих сигналів *у3 = y4 =* 1 тригери не вимагаються. Співпадаючі сигнали *в1* і *в*2 знімаються з одного тригера.

### ПІДГОТОВКА ДО РОБОТИ

1. Скласти змістовний мікроалгоритм обчислення функції *D* на операційному пристрої (рис. 2.2). Вид функції *D* визначається по табл. 2.7 у відповідності зі значеннями альфа5 і альфа4 двох цифр номера варіанта, записаного у виді двійкового слова, обмеженого п'ятьма молодшими розрядами .Вважати, що у вихідному стані операнды *А, В к С* записані відповідно в *P1, С* и *Р2.* Для опису мікрооперацій використовувати символіку, приведену в гл. 2.

2. Відповідно до отриманого мікроалгоритму і рис. 2.2 побудувати керуючий автомат для випадку виконання мікрооперацій у синхронному режимі. Тип тригерів визначається по табл. 2.7 і залежності, від значень  і , а тип автомата - за значенням а1. Для побудови комбінаційної схеми використовувати елементи I-НІ.

3. Побудувати часову діаграму роботи автомата для кожної комбінації значень логічних умов.

4. Проілюструвати прикладом обчислення результату *D* для одного довільного набору значень операндов *А, В я С,* представлених трьома двійковими розрядами.

### ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Зібрати схему автомата і налагодити її. Логічні умови задаються за допомогою тумблерів.

2. Зібрати схему операційного пристрою (рис.2.2) і підключити до його керуючих входів виходи автомата. Записати в *Р1, С* і *Р2* трьохрозрядні операнди, що відповідають п. 4 завдання, подати сигнал ПУСК і перевірити правильність отримання результату *D* у *Р2.*

##### ЗМІСТ ЗВІТУ

Звіт повинен містити короткі теоретичні відомості, необхідні для виконання лабораторної роботи, усі схеми, формули, таблиці і графіки, отримані при виконанні теоретичного завдання й у процесі експериментального дослідження схем, а також висновки по роботі.

### КОНТРОЛЬНІ ПИТАННЯ

1. Що таке мікроалгоритм операції?

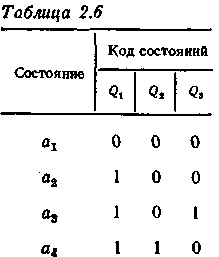
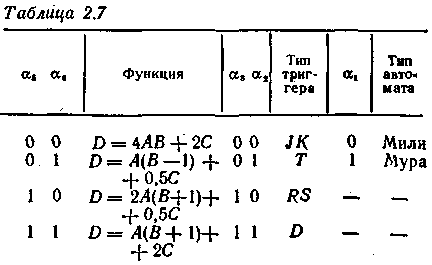
2. Намалюйте узагальнену структуру схеми керуючого автомата.

3. Напишіть вирази, що визначають закон функціонування автоматів Мілі і Мура.

4. У чому складається відмінність автоматів Мілі і Мура?

5. Охарактеризуйте основні етапи проектування керуючого автомата.

6. Як перейти від змістовного мікроалгоритму до закодованого мікроалгоритму?

7. Як побудувати граф автомата?

8. Як здійснюється оцінка станів автомата?

9. Як визначити необхідну тривалість керуючих сигналів?

10. Від чого залежить кількість тригерів, необхідне для побудови автомата?

11. У чому суть протигоночного кодування станів автомата?

12. Як скласти структурну таблицю автомата?

13. Складіть таблицю переходів для *JK-, RS-, T-* і D-тригерів.

14. Чи можливий перехід автомата в стан, не передбачене графом, при використанні тригерів із внутрішньою затримкою (тригерів, керованих рівнем сигналів)?

15. Коли можливе виникнення помилкових керуючих сигналів (не передбачених графом автомата) і чим визначається їхня тривалість?

16. Як визначити час переходу автомата з одного стану в інше?

17. Як отримати *T-* і *RS-тригери* на основі *JK-тригерів*?

18. Як побудувати часову діаграму роботи автомата?

# Тема (лабораторна робота №7):

## ПРОЕКТУВАННЯ І ДОСЛІДЖЕННЯ ОПЕРАЦІЙНИХ АВТОМАТІВ (НА “ТВЕРДІЙ” ЛОГІЦІ) ТА КЕРУЮЧИХ АВТОМАТІВ НА “М’ЯКІЙ” (“ПРОГРАМОВАНІЙ”) ЛОГІЦІ

(усно)

**Тема: ПРИНЦИП МІКРОПРОГРАМНОГО КЕРУВАННЯ І СТРУКТУРА ОПЕРАЦІЙНИХ ПРИСТРОЇВ**

**1. Принцип мікропрограмного керування**

Для виконання операцій над інформацією використовуються ***операційні пристрої*** - процесори, канали введення-виведення, пристрої керування зовнішніми пристроями і т.п. ***Функцією операційного пристрою*** є виконання заданої множини операцій F = {f1, ..., fG} над вхідними словами D = {d1; ..., dH} з метою обчислення слів R = {r1, ..., rQ}, що представляють результати операцій R=fg(D), g = 1, ..., G. Функціональна і структурна організація операційних пристроїв, що визначає порядок функціонування і структуру пристроїв, базується на ***принципі* *мікропрограмного керування****,* що полягає в наступному.

1. Будь-яка операція *fg, g =* 1, ..., *G,* якареалізується операційним пристроєм, розглядається як складна дія, що розділяється на послідовність елементарних дій над словами інформації, які називають ***мікроопераціями****.*

2. Для керування порядком проходження мікрооперацій використовуються ***логічні умови****,* що у залежності від значень слів, які перетворюються мікроопераціями, приймають значення «істина» чи «хибність» (1 чи 0).

3. Процес виконання операцій у пристрої описується у формі алгоритму, що представляється в термінах мікрооперацій і логічних умов і називається ***мікропрограмою****.* Мікропрограма визначає порядок перевірки значень логічних умов і проходження мікрооперацій у процесі отримання необхідних результатів.

4. ***Мікропрограма використовується як форма представлення функції операційного пристрою, на основі якої визначається структура і порядок функціонування пристрою у часі***.

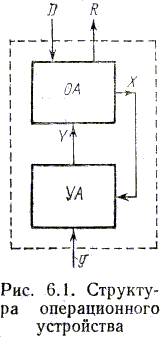
Сказане можна розглядати як ***змістовний опис принципу мікропрограмного керування***, з якого випливає, що структура і порядок функціонування операційних пристроїв визначаються алгоритмом виконання операцій F = {f1,..., fG}.Схемні рішення операційного пристрою можуть бути різні, але у всіх випадках точка зору на процес функціонування пристрою як процес реалізації мікрооперацій і перевірки логічних умов, що обумовлений мікропрограмою, є ефективною (результативною), оскільки дозволяє упорядкувати і формалізувати проектування операційних пристроїв.

**2. Концепція операційного і керуючого автоматів**

У функціональному і структурному відношенні операційний пристрій складається з двох частин: ***операційного і керуючого автоматів*** (рис. 6.1).

***Операційний автомат*** *ОА* служить для збереження слів інформації, виконання набору мікрооперацій і обчислення значень логічних умов, тобто операційний автомат є структурою, організованою для виконання дій над інформацією. Мікрооперації, що реалізуються операційним автоматом, ініціюються (збуджуються) множиною ***керуючих сигналів***Y = {y1, ..., уM},з кожним з яких ототожнюється визначена мікрооперація. Значення логічних умов, що обчислюються в операційному автоматі, відображаються множиною ***інформаційних сигналів***X={x1, ..., хL},кожен з яких ототожнюється з визначеною логічною умовою.

***Керуючий автомат*** *КА* генерує визначену мікропрограмою послідовність керуючих сигналів, яка відповідає інформаційнимсигналам (значенням логічних умов). Інакше кажучи, керуючий автомат задає в часі порядок виконання дій в операційному автоматі, який (порядок) випливає з алгоритму виконання операцій.



***Найменування операції***, яку необхідно виконати в пристрої, визначається кодом *g* операції. Стосовно керуючого автомату сигнали g1, ..., gG,за допомогою яких кодується найменування операції, і інформаційні сигнали x1, ..., хL,які сформовані в операційному автоматі, відіграють однакову роль - вони впливають на порядок вироблення керуючих сигналів *Y.* Тому сигнали g1, ..., gh і x1 ,..., хLвідносяться до одного класу - до класу інформаційних сигналів, що надходять на вхід керуючого автомату.

Таким чином, будь-який ***операційний пристрій*** ***є композицією зі зворотнім зв’язком операційного і керуючого автоматів***. Операційний автомат, що реалізує дії над словами інформації, є виконавчою частиною пристрою, роботою якого керує керуючий автомат, що генерує необхідні послідовності керуючих сигналів.

На даному рівні розгляду ***операційний і керуючий автомати можуть бути визначені своїми функціями*** - переліком дій, що виконуються ними, виходячи з яких надалі буде визначена структура автоматів.

***Функція операційного автомату***визначається:

1. Множиною ***вхідних слів*** D = {d1, ..., dH},щовводяться в автомат у якості операндів.

2. Множиною ***вихідних слів*** R= {r1, ..., rQ}*,* щопредставляють результати операцій.

3. Множиною ***внутрішніх слів*** S = {s1, ..., sN}, що використовуються для представлення інформації в процесі виконання операцій. Надалі будемо припускати, що вхідні і вихідні слова співпадають з відповідними внутрішніми словами, тобто DS і RS.

4. Множиною ***мікрооперацій*** *Y* = {y1, ..., *у*M}*,* що реалізують перетворення S=m(S)над словами інформації, де m - обчислювана функція.

5. Множиною ***логічних умов*** X= {x1,*...,* хL}, де xi *=* i(S)і i *-* булева функція.

Таким чином, ***функція операційного автомата задана, якщо визначені множини*** D, R, S, Y, X*.*

Помітимо, що ***час не є аргументом функції операційного автомату***. Функція встановлює список дій - мікрооперацій і логічних умов, - які може виконувати автомат, але ніяк не визначає порядок проходження цих дій у часі. Інакше кажучи, функція операційного автомата характеризує засоби, що можуть бути використані для обчислень, але не сам обчислювальний процес. Порядок виконання дій у часі визначається у формі функцій керуючого автомата.

***Функція керуючого автомату***- це операторна схема алгоритму (мікропрограми), функціональними операторами якої є символи y1, ..., *у*M*,* щоототожнюються з мікроопераціями, а в якості логічних умов (предикатів) використовуються булеві змінні x1,*...,* хL*.*

***Операторна схема алгоритму*** найбільш часто представляється у вигляді ***граф-схеми алгоритму*** чи ***логічної схеми алгоритму***. Кожна з цих форм визначає обчислювальний процес у послідовному аспекті - установлює порядок перевірки логічних умов x1,*...,x*Lі порядок виконання мікрооперацій y1, *...*, *у*M*.*

**Тема: ВПЛИВ СИСТЕМ МІКРООПЕРАЦІЙ І ЛОГІЧНИХ УМОВ НА ХАРАКТЕРИСТИКИ ОПЕРАЦІЙНИХ ПРИСТРОЇВ**

**Основними характеристиками операційних пристроїв** є швидкодія і витрати устаткування. *Швидкодія* операційного *пристрою* визначається середнім часом виконання операцій

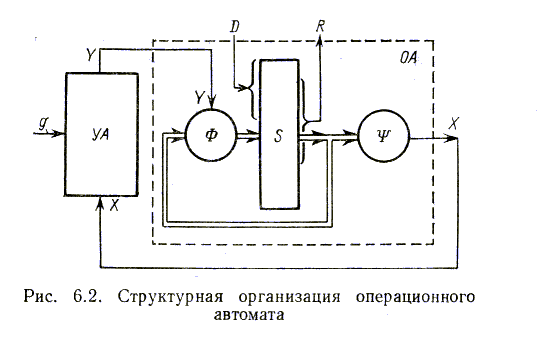
де *pg* - імовірність виконання операцій fg; - середній час виконання операції. Імовірності р1, ..., рGвизначаються класом задач, для рішення яких призначається ЕОМ і вважаються відомими. Операційний пристрій функціонує в дискретному часі *t* = 0, 1,2,... Проміжок між двома моментами *t* і *(t +* 1) дискретного часу називається *тактом.* Протягом такту формується набір керуючих сигналів, виконуються відповідні мікрооперації й обчислюються значення логічних умов. Тривалість такту *Т* залежить від складності мікрооперацій і логічних умов і швидкодії елементів, з яких побудований керуючий і операційний автомати. З урахуванням цього середній час виконання операції g = T\**,* де T - тривалість такту*;*  - середнє число тактів, за яке пристрій реалізує операцію fg. Витрати устаткування в операційному пристрої визначаються сумарною вартістю Cелементів, з яких складається пристрій.

Найбільш істотний вплив на швидкодію операційного пристрою і витрати устаткування робить набір мікрооперацій Y={уm}і логічних умов X = {хl} і, в першу чергу, - система функцій, на основі яких вони будуються. Система функцій (операцій), на основі яких описується алгоритм (мікропрограма) виконання операцій *F =*{f1, ..., fG} визначається в такий спосіб.

**Система утворюючих алгоритму.** По визначенню, введеному в п. 6.1, мікрооперація *ут*  *Y* - це присвоювання слову sa S значення sa : = - результату застосування функції до значень слів . Аналогічно, логічна умова *xl*  *X -* це булева функція , що визначає відношення між словами S. У множині *Y* = *{ym}* мікрооперації розрізняються як функціями Ф = { }, на основі яких утворяться мікрооперації, так і наборами слів , що є операндами функції. Аналогічно, множину логічних умов можна представити у вигляді набору булевих функцій , на основі яких будуються логічні умови, і наборів слів , на які поширюються булеві функції. Система , що складається з множини функцій , що утворюють мікрооперації, і множини булевих функцій , що утворюють логічні умови, називається *системою утворюючих алгоритму.* Наприклад, набір мікрооперацій *Y* = *{ym}* може будуватися на базі функцій де - знаки булевих операцій конъюнкции, диз'юнкції і заперечення (інверсії). Набір логічних умов *X =*  може будуватися на базі відносин .

Як показує досвід, склад функцій (операцій) у системі Z дуже істотно позначається на часі виконання алгоритмів (швидкодії) і витратах устаткування в операційному пристрої. Це пояснюється тим, що мікрооперації і логічні умови утворюються шляхом застосування операцій, що входять у систему утворюючих , до слів, з якими оперує алгоритм. *Тому визначення набору функцій Z є першочерговою задачею мікропрограмування*.

**Структурна реалізація мікрооперацій і логічних умов.** У загальному випадку операційний пристрій будується за схемою на рис. 6.2, що є більш детальним представленням схеми на рис. 6.1. Операційний автомат *ОА* розділяється на три частини: пам'ять S; комбінаційну схему Ф, що реалізує мікрооперації; комбінаційну схему , що обчислює значення



логічних умов. Пам'ять S забезпечує збереження слів які представляють значення операндів *D,* проміжні значення і кінцеві результати *R.* Для виконання мікрооперацій *Y = {ут}* служить комбінаційна схема Ф. Керуючі сигнали *Y, що* формуються керуючим автоматом *УА,* ініціюють виконання необхідних мікрооперацій. Так, якщо надходять сигнали і , то схема Ф виконує дві мікрооперації: ; , що зводиться до обчислення значень ; і присвоюванню їх словам . Для обчислення значень логічних умов служить комбінаційна схема *, що* реалізує систему булевих функцій , значення яких представляються інформаційними сигналами .

Керуючий автомат відповідно до коду операції *g* генерує набір керуючих сигналів, що ініціює відповідний набір мікрооперацій. Виконання мікрооперацій приводить до зміни стану пам'яті операційного автомата - значень слів . Стан пам'яті S відображається множиною інформаційних сигналів *X,* що аналізуються керуючим автоматом для визначення наступного набору мікрооперацій. Процес вироблення набору керуючих сигналів, виконання мікрооперацій і обчислення логічних умов займає один такт часу, тривалість *Т* якого визначається швидкодією логічних і запам'ятовуючих елементів. Кількість тактів залежить від операції, що реалізує пристрій, і складу системи утворюючих : чим «элементарнее», простіше функції Ф і *,* тим більше число тактів потрібно для виконання операції. Витрати устаткування в операційному пристрої, як випливає з рис. 6.2, визначаються сумою С=СS+СФ+С+СКА, де СS, СФ, С, СКА *-* витрати устаткування, що приходяться, відповідно, на пам'ять S, комбінаційні схеми Ф и , що реалізують мікрооперації і логічні умови, і керуючий автомат.

**Різноманіття операційних пристроїв**. Розглянемо різні варіанти побудови операційного пристрою, що реалізує фіксовану множина операцій над словами *D* з метою обчислення слів *R.* Кожен період роботи пристрою зв'язаний з виконанням операції *,* заданої кодом операції *g* = 1, ..., G. Функцию пристрою будемо представляти в наступній скороченій формі: , де

- перетворення, реалізоване пристроєм. Ця функція реалізується на основі принципу мікропрограмного керування структурою, приведеної на рис. 6.2.

Як перший варіант побудови пристрою можна прийняти, що перетворення - єдина мікрооперація, що реалізує пристрій. Це означає, що множина утворюючих мікрооперацій *Ф1 ={* *}* складається з єдиної функції (операції) і отсутствует необхідність у логічних умовах, тобто .Отже, якщо прийняти, що - мікрооперація, система утворюючих алгоритму складається з функції і порожньої множини функцій, зв'язаних з логічними умовами. Як показано в [54], функція реалізується комбінаційною схемою, що позначимо *.* Зіставляючи цей варіант реалізації перетворення *R* = = *(D)* зі структурою (рис. 6.2), можна бачити, що в даному випадку отсутствует необхідність у керуючому автоматі, пам'яті для збереження проміжних результатів і схемі , щообчислює значення логічних умов. З обліком цього будемо вважати, що структура (рис. 6.2) вироджується в єдину комбінаційну схему Ф, що збігається зі схемою *Ul* і реалізує перетворення за один такт. Таким чином, реалізація перетворення на основі системи утворюючих приводить до комбінаційної схеми *Ul* ,що за один такт обчислює значення *R, що* відповідають вхідним словам D і коду операції *g.*

Розглянемо інший варіант побудови операційного пристрою, що реалізує те ж перетворення *R* = *(D).* Приймемо, що елементи схеми (рис. 6.2) виконують наступні функції. Інформація, збережена в пам'яті S, розділяється на біти *.* Схема Ф виконує єдину функцію де - знак операції І-НІ, на основі якої можуть бути реалізовані мікрооперації , де - біти (однорозрядні слова). На виході схеми формується єдина логічна умова , значення якого 0 чи 1 визначається значенням результату мікрооперації. В алгоритмі значення використовується для перевірки на нуль (чи на одиницю) значення . Отже, логічні умови утворяться з єдиної функції  *=* {= 0}. Систему утворюючих позначимо як *ZP.* Система *ZP* є функціонально повної, тобто в термінах функцій можна описати будь-який алгоритм, і на основі функцій *ZP* можна побудувати операційний пристрій *Up, що* реалізує перетворення . Пристрій *UP* у каж-дом такті може виконувати одну мікрооперацію . У зв'язку з цим процес виконання перетворення повинний бути представлений у формі мікропрограми, реалізованої керуючим автоматом.

Зіставимо характеристики операційних пристроїв і *UP,* яким відповідають вищеозначені системи утворюючих і . Пристрій , являє собою комбінаційну схему і виконує перетворення за один такт часу і є найбільш швидкодіючим. Пристрій *Up* виконує перетворення у виді послідовності найпростіших операцій над бітами інформації. Тому число тактів, необхідних для реалізації перетворення , буде максимальним. Витрати устаткування в операційному пристрої: і має максимальне значення, так що сумарні витрати . Для пристрою *UP* витрати устаткування в керуючому автоматі і пам'яті операційного автомата мають максимальні значення в порівнянні з іншими можливими варіантами побудови пристроїв, а витрати устаткування Сф і gf, що приходяться "на комбінаційні схеми Ф и *,* мінімальні.

Таким чином, у загальному випадку ОП , *UP* і відповідні їм системи утворюючих Zl , *ZP* мають діаметрально протилежні властивості:

1) пристрій є найбільш швидкодіючим, а пристрій *UР* - найменш швидкодіючим;

2) у пристрої керуючий автомат відсутній, у той час як у пристрої *UP* на керуючий автомат випадає максимальна частка витрат устаткування;

3) операційний автомат у пристрої *Ul* є найбільш складним по кількості використовуваного устаткування, а в пристрої *UP* - найбільш простим.

Очевидно, що між діаметрально протилежними варіантами побудови операційних пристроїв , *UP* знаходиться ряд варіантів, показники яких (швидкодія і витрати устаткування) займають проміжне положення між екстремальними значеннями, що відповідають пристроям , *UP.* Припустимо, що існує прийом, що дозволяє для перетворення виявляти будь-які можливі системи утворюючих ісинтезувати на їхній основі ОП зі швидкодією і мінімальні по витратах устаткування.

Такого роду системам утворюючих і відповідним цим системам пристроям будемо приписувати індекси *i =* 1, . .. , *Р* в порядку зменшення швидкодії пристроїв. При цьому часи виконання перетворення пристроями будуть утворювати монотонно зростаючу послідовність . Послідовність Zl, Z2, ..., ZPбудемо називати *ієрархією систем утворюючих* перетворення , а відповідну послідовність пристроїв *U1*, *U*2, ..., *Uр* - *ієрархією операційних пристроїв, що* реалізують перетворення *.*

Ієрархія систем утворюючих містить у собі ієрархію наборів мікрооперацій і ієрархію наборів логічних умов, функціонально повних у відношенні перетворення. Можна припускати, що ієрархія операційних пристроїв має наступні властивості (рис. 6.3):

1) автоматний час виконання перетворення монотонно зростає;

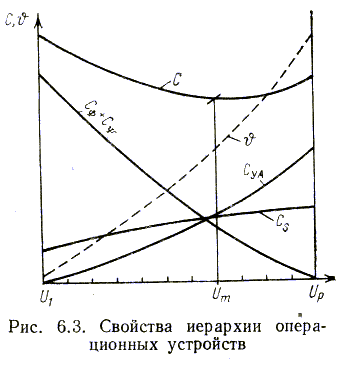
2) витрати устаткування в комбінаційній частині операційного автомата монотонно убувають;

3) витрати устаткування , щоприходяться на керуючий автомат, монотонно зростають, починаючи від нуля;

4) витрати устаткування , щоприходяться на пам'ять операційного автомата, монотонно зростають;

5) в ієрархії пристроїв мається пристрій *Um,* сумарні витрати устаткування в який мінімальні, причому в загальному випадку пристрій *Um* займає серединне положення між пристроями і ,тобто 1 < *т < Р.*

Вплив системи утворюючих на характеристики операційних пристроїв ілюструється залежностями,



приведеними на рис. 6.4, де і - витрати устаткування відповідно в операційному і керуючому автоматах і С = + - витрати устаткування в операційному пристрої. Зміна системи утворюючих Z шляхом розкладання операцій *,* реалізація яких покладається на пристрій, на усе більш і більш прості операції - мікрооперації і логічних умов - приводить до зменшення витрат устаткування в операційному автоматі , але одночасно з цим збільшуються витрати устаткування в керуючому автоматі . Таким чином, змінюючи склад мікрооперацій і логічних умов, у термінах яких описується алгоритм виконання операцій, можна устаткування з операційного автомата «перекачувати» у керуючий автомат і навпаки. З рис. 6.4 видно, що існує система утворюючих Zm, якій відповідає визначений набір мікрооперацій і логічних умов, що забезпечує мінімум витрат устаткування в операційному пристрої. Однак відповідний системі *Zm* час виконання операцій , як видно з рис. 6.3, досить великий. Щоб зменшити значення , тобто збільшити швидкодію пристрою, необхідно додаткове устаткування, що використовується для реалізації більш складних мікрооперацій і логічних умов.

Задача породження систем утворюючих (наборів мікрооперацій і логічних умов), що мінімізують витрати устаткування чи забезпечують заданий час виконання операцій у пристрої, досить складна, що ускладнює розробку формальних методів вибору оптимального складу мікрооперацій і логічних умов. Можливі підходи до рішення цієї задачі описані в [54].

**Тема: ЗАДАЧА ПРОЕКТУВАННЯ ОПЕРАЦІЙНИХ ПРИСТРОЇВ І КОНЦЕПЦІЯ ФУНКЦІОНАЛЬНОГО МІКРОПРОГРАМУВАННЯ**

На етапі системного проектування ЕОМ, коли ЕОМ розглядається як одне ціле, виявляються номенклатура пристроїв, зокрема операційних пристроїв, спосіб сполучення пристроїв (інтерфейси) і вимоги до швидкодії пристроїв. У результаті для кожного операційного пристрою стає відомим: 1) перелік входів D і виходів *R* пристрою, обумовлений складом інтерфейсів; 2) набір операцій *,* реалізація яких покладається на пристрій; 3) вимога до швидкодії пристрою, обумовлена обмеженням

\* (6.1)

на середній час виконання операцій. З урахуванням цього задача проектування операційного пристрою ставиться в такий спосіб. Задано функції операційного пристрою, обумовлені множинами D, *R* вхідних і вихідних слів і множиною операцій *F,* а також вимоги до швидкодії пристрою, обумовлені середнім часом виконання операцій. Потрібно синтезувати схему пристрою, що забезпечує реалізацію заданих функцій із заданою швидкодією і, що є мінімальною за кількостю використовуваного устаткування (вартості).

Щоб синтезувати схему операційного пристрою, необхідно прийняти деякий спосіб виконання операцій у пристрої й описати його у формі мікропрограми. Мікропрограма, що представляє функцію операційного пристрою безвідносно до засобів, що можуть бути використані для реалізації заданої функції, називається *функціональною мікропрограмою.* Функціональна мікропрограма фіксує в собі алгоритм виконання операцій, що рекомендується проектувальником, і використовується як вихідна форма представлення функцій пристрою, на основі якої синтезується структура, достатня для реалізації заданих функцій.

Для запису функціональних мікропрограм необхідна яка-небудь алгоритмічна система - *мова функціонального мікропрограмування.* Засоби мови повинні забезпечувати опис алгоритмів виконання операцій - слів, мікрооперацій, логічних умов і порядку їхнього виконання - з таким ступенем деталізації, що забезпечить синтез структури пристрою. При проектуванні операційних пристроїв і ЕОМ використовуються різні мови мікропрограмування, серед яких найбільш відомі мови LOTIS, АЛГОРИТМИК, CDL, ФОРОС і ПРОЕКТ. Ці мови містять у собі засоби, достатні для опису порядку функціонування пристроїв і перевірки коректності алгоритмів шляхом їхнього моделювання на ЕОМ.

**Мова функціонального мікропрограмування (Ф-мова)**

Ф-мова є засобом опису функцій операційних пристроїв на рівні мікропрограм безвідносно до структури (апаратури), що може бути використана для реалізації цих функцій. Ф-мова будемо розглядати як інженерна (не машинна) мова. З цієї причини використовуємо в мові загальноприйняту математичну символіку, таблиці в їхній звичайній формі і графічне представлення схем алгоритмів. Машинно-орієнтована форма Ф-мови описана в [46].

Об'єктами, що описуються засобами Ф-мови, є функціональні мікропрограми, що визначають алгоритми виконання операцій у пристроях. Функціональна мікропрограма містить у собі два роди зведень, дві частини: 1) опис слів і масивів, що встановлює типи і формати слів, з якими оперує мікропрограма; 2) змістовний граф мікропрограми, що визначає алгоритм виконання операцій у змістовній формі - у виді описів мікрооперацій і логічних умов.

**Опис слів і масивів.** Основним елементом інформації, з яким оперує функціональна мікропрограма, є *слово.* Найменування, формат і тип слова, що характеризує спосіб присвоювання і використання значень слова, визначаються описом слова. Найменування і формат слова задаються в наступному виді:

(6.2)

де С - ідентифікатор слова; і *п2* - номера старшого і молодшого двійкових розрядів слова. Приймемо, що розряди слова нумеруються ліворуч праворуч ненегативними цілими числами, тобто .Відповідно до опису (6.2) розряди слова одержують номера . Так, опис *А* (0:31) визначає 32-розрядне слово *А* и опис *В* (1:8) - 8-розрядне слово *В.* Опис однорозрядного слова складається тільки з ідентифікатора С. Так, опису Z, *ПП, ЗНАК* визначають три однорозрядних слова.

Сукупність слів, що мають однакову довжину, може поєднуватися в *масив.* Масив описується в наступному виді:

(6.3)

де *М -* ідентифікатор масиву; - границі номерів слів, що складають масив, причому ; - номера старшого і молодшого розрядів слова. Наприклад, опис *РП* [0:15] (0:31) представляє масив з 16 32-розрядних слів, а опис *ОП* [0 : 4095] (0:63) - масив з 4096 64-розрядних слів.

Деякі слова в семантичному відношенні розділяються на частині, називані *полями,* і дії в мікропрограмі можуть вироблятися як над цілими словами, так і над частинами слів - полями. Поля представляються наступними конструкціями:

(6.4)

де *С* - ідентифікатор слова ; *-* номера старшого і молодшого розрядів поля, причому ; - номер розряду слова . Перша конструкція в (6.4) виділяє -розрядне поле слова *С,* а друга - однорозрядне поле, р-й розряд слова *С.* Щоб акцентувати особливий зміст деяких полів, полям приписуються власні найменування. Найменування полів уводяться конструкціями наступного виду:

(6.5)

Тут G і H - слова, що іменують відповідні поля слова С. Довжини слів, що іменують, повинні збігатися з довжинами полів:

Щоб спроектувати структуру операційного пристрою, необхідно у функціональній мікропрограмі визначити спосіб використання значень слів і масивів, що обумовлюється типом елементів інформації. У залежності від способу використання значень виділяються наступні основні типи слів: 1) вхідні - значення приписуються поза мікропрограмою і використовуються усередині мікропрограми; 2) внутрішні (локальні) - значення приписуються і використовуються тільки усередині мікропрограми; 3) допоміжні - значення приписуються і використовуються тільки усередині мікропрограми, але існують не постійно, а тільки протягом обмежених інтервалів часу - у межах такту автоматного часу; 4) вихідні - значення приписуються мікропрограмі і використовуються поза нею. Типи слів позначаються буквами: I - вхідні, L - внутрішні, А - допоміжні, О - вихідні. Деякі слова можуть використовуватися як елементи інформації декількох типів. Наприклад, слово може вживатися як вхідне, внутрішнє і вихідне. Тому деякі слова можуть мати наступні типи IL, LO і ILO. Останній тип відповідає слову, значення якому може приписуватися і використовуватися до входу в мікропрограму, у мікропрограмі і по виходу з неї.

З використанням прийнятих позначень, множина слів, з якою оперує мікропрограма, описується у вигляді табл. 6.1. Перший стовпець таблиці визначає типи слів за допомогою символів I, L, О и другий - найменування і формати слів. У даному випадку за допомогою опису вводиться три 16-розрядних слова D, А, В, одне 17-розрядне слово *С* и одне 4-розрядне слово *СЧ.* Старші розряди слів *А, В, С* використовуються для представлення знаків і цих розрядів приписуються найменування *ЗНА, ЗНВ, ЗНС,* яким відповідають однорозрядні двійкові слова. Функціональна мікропрограма, які відповідає табл. 6.1, може оперувати тільки з тими словами, що перераховані в таблиці.

**Двійкові вирази.** Двійковий вираз визначає правило обчислення двійкового значення шляхом виконання операцій над первинними двійковими значеннями. Як первинні значення використовуються константи і значення слів, елементів масивів і полів.

Константи записуються в наступному виді:

1100 148 1210 С16

Таблиця 6.1

**Опис слів**

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Найменування і формат | Поля | Пояснення |
| I | *D*(0:15) |  | Вхідне слово |
| L | *А* (0:15) | *ЗНА* = *А*(0) | 1-й операнд |
| L | *В* (0:15) | *ЗНВ = В*(0) | 2-й операнд |
| LO | *С* (0:15) | *ЗНС* = С(0) | Результат |
| L | *СЧ* (1:4) |  | Лічильник |

Тут визначені чотири константи: двійкова, вісімкова, десяткова і Шістнадцяткова, що представляють те саме значення в різних формах. шістнадцяткові константи складаються із символів, перерахованих у табл. 6.2. Кожний із шістнадцяткових символів відповідає визначеному 4-розрядному двійковому коду. Константі з *п* розрядами відповідає двійковий набір з визначеним числом розрядів: двійковій константі - n-розрядний набір; вісімковій константі - Зn-розрядний двійковий набір; число розрядів двійкового значення, що відповідає десятковій константі, визначається кількістю значущих цифр у двійковому значенні.

Таблиця 6.2

**Шістнадцятиричні символи**

|  |  |  |  |
| --- | --- | --- | --- |
| Двійковий код | Символ | Двійковий код | Символ |
| 0 0 0 0 | 0 | 1 0 0 0 | 8 |
| 0 0 0 1 | 1 | 1 0 0 1 | 9 |
| 0 0 1 0 | 2 | 1 0 1 0 | А |
| 0 0 1 1 | 3 | 1 0 1 1 | B |
| 0 1 0 0 | 4 | 1 1 0 0 | С |
| 0 1 0 1 | 5 | 1 1 0 1 | D |
| 0 1 1 0 | 6 | 1 1 1 0 | Е |
| 0 1 1 1 | 7 | 1 1 1 1 | F |

*Слова* в двійкових виразах представляються своїми ідентифікаторами. Так, *D, А, В,* С, *СЧ, ЗНА, ЗНВ, ЗНС -* слова, розрядність яких визначена в табл. 6.1. Слово, що є елементом масиву *М,* записується у виді *М [* *],* де *М* - ідентифікатор масиву і - номер слова в масиві, значення якого повинно відповідати опису (6.3), тобто .

*Поля* записуються у виді (6.4) чи представляються ідентифікаторами слів, що іменують поля. Наприклад, у слові A, описаному в табл. 6.1, можуть виділятися поля

*А* (0), *А (1:4*), *А* (14:15), причому поле *А* (0) може представлятися словом *ЗНА.* У двійкових виразах елементи масивів і полів використовуються нарівні зі словами, представляючи в кожнім випадку відповідне двійковое значення.

Первинні двійкові значення - константи, слова і поля - поєднуються в двійкові вираз за допомогою двійкових операцій, список яких приведений у табл. 6.3. Операція інверсії є унарною і забезпечує зміну значення розрядів на зворотне - 0 на 1 і 1 на 0. Так, якщо *А* (1:4) і *В* (1:4) - слова зі значеннями 1010 і 0110 відповідно, то двійкові вирази *А* = 0101 і *В =* 1001. Операція складання «.» служить для приєднання значення, зазначеного праворуч від знака операції, до значення, зазначеному ліворуч. Так, *А. В =* 01011001 і *В. А =* 10010101. Операції конъюнкции і диз'юнкції у відмінності від булевих операцій застосовуються до наборів значень, що відповідають однойменним розрядам операндів.

Таблиця 6.3

|  |  |  |
| --- | --- | --- |
| Знак операції | Найменування операції | Порядок старшинства |
|  | Інверсія | 1 |
| . | Складання | 2 |
|  | Кон’юнкція | 3 |
|  | Диз'юнкція | 4 |
|  | Додавання по модулю 2 (нерівнозначність) | 4 |
| + | Додавання | 5 |
|  | Циклічне додавання | 5 |
| - | Віднімання | 5 |

Наприклад, і . Аналогічним образом інтерпретується операція додавання по модулю 2: . Результат додавання по модулю 2 дорівнює нулю тільки в тому випадку, коли значення однойменних розрядів операндов рівні між собою, тобто якщо *А* = *В.* Операція додавання виконується за правилами двійковой арифметики над операндами *А, В* як над цілими двійковими числами без знака

Результат додавання n-розрядних значень може складатися з *(п* + 1) значущого розряду. Операція циклічного додавання виконується так само, як операція додавання, але перенос зі старшого розряду суми додається до молодшого розряду

Операція віднімання поширюється на цілі двійкові значення без знака:

При відніманні меншого значення з більшого результат представляється двійковим значенням з числом розрядів, що дорівнює числу розрядів операндів. При відніманні більшого значення з меншого результат представляється в додатковому коді і містить нескінченне число розрядів, причому старші розряди дорівнюють 1.

При виконанні бінарних операцій операнди сполучаються по молодших розрядах. Операнд із меншим числом розрядів доповнюється нулями з боку старших розрядів до довжини операнда з великим числом'разрядов.

Двійковий вираз будується з первинних двійкових значень з використанням знаків двійкових операцій і дужок (,). Порядок обчислення двійкового виразу визначається дужками, старшинством операцій, зазначеним у табл. 6.3, і порядком проходження операцій однакового старшинства в записі двійкового вираз.

Прикладами двійкових виразів є наступні конструкції:

Останній вираз обчислюється в наступному порядку: 1) інвертування ; 2) формування складеного значення ; 3) додавання С с складеним значенням; 4) додавання суми зі значенням 1.

**Мікрооперації.** У синтаксичному змісті (за формою запису) *мікрооперація* - це оператор присвоювання, за допомогою якого слову присвоюється значення двійкового виразу. Мікрооперація складається з лівої частини, знака присвоювання := і двійкового виразу. У лівій частині оператора вказується слово, поле чи складене слово виду *А.В.* Наступні оператори є мікроопераціями:

- присвоювання слову A нульового значення;

- присвоювання полю слова В значення 1;

- присвоювання слову A значення слова В;

С = С + 1 - збільшення значення слова С на 1;

С = С - 1 - зменшення значення слова С на 1;

С = С + A - збільшення значення С на значення слова A;

- додавання С с додатковим кодом слова А, тобто віднімання А з С;

А.В:=X - присвоювання слову *А* старшим розрядам слова X і слову В молодших розрядів слова X.

Операцію присвоювання := будемо розуміти наступним чином. Якщо в лівій частині оператора зазначене -розрядне значення, а двійковий вираз містить розрядів, то при слову присвоюється молодших розрядів двійкового вираз, а при старшим розрядам приписуються значення 0 і , молодшим розряд-значення двійкового вираз.

**Класифікація мікрооперацій.** Випливаючи інженерним традиціям, мікрооперації підрозділяються на наступні класи: 1) установки; 2) інвертування; 3) передачі; 4) зсуву; 5) рахунку; 6) додавання; 7) бінарні логічні; 8) комбіновані.

*Мікрооперація установки* - присвоювання слову значення константи. Наприклад, .

*Мікрооперація інвертування* забезпечує зміну значення слова на інверсне. Наприклад, .

*Мікрооперація передачі* - присвоювання слову значення іншого слова, у тому числі - інверсії чи складеного слова. Наприклад, А : = В; А (0) : = В (0); С : = А; С: = 11.B(1:15).

*Мікрооперація зcуву* служить для зміни положення розрядів слова стосовно початкового шляхом переміщення кожного розряду на *k* позицій вліво чи вправо. Будь-яка мікрооперація зсуву може бути представлена у формі оператора присвоювання. Наприклад, зсуви слова *А* (1:32) можуть бути описані в такий спосіб:

А: = А(21:32).0 - зсув на один розряд уліво з уведенням нуля в вивільнюваний при зсуві розряд;

А: = А (2:32).А(1) - циклічний зсув на один розряд уліво;

А: = 00.A(1:30) - зсув на два розряди вправо з уведенням нулів у вивільнювані розряди;

А: = А (31:32).А (1:30) - циклічний зсув на два розряди вправо.

У цілях скорочення запису мікрооперацій зсуву і додання семантичного відтінку відповідним операторам використовуються дві стандартні процедури:

Rк(А) - видалення з двійкового виразу A k молодших правих розрядів, тобто зсув значення на k розрядів вправо;

Lк(А) - видалення з двійкового виразу A k старших лівих розрядів, тобто зсув значення на k розрядів уліво.

Якщо - n-розрядне двійковое значення, то Rk (A) = а1а2 ... an-k і Lk(A)=. З використанням зазначених процедур перераховані вище мікрооперації зсуву представляються наступними операторами: А:=Ll (A.0); А:=Ll(А.А(1)); А:=R2(00.A); А:=R2(A(31:32).А). Додатково відзначимо, що якщо А(1:32) і В(1:32) - 32-розрядні слова, то оператор А:=0.В(1:31) представляє не мікрооперацію зсуву, а передачу слова В зі зсувом на один розряд вправо, інакше - передачу поля В(1:31).

*Мікрооперація рахунку* забезпечує зміну значення слова на одиницю. Наприклад, А:=А+1 ; В:=В-1; С:=A+1; С(1:4):=С(1:4)+1.

*Мікрооперація додавання* служить для присвоювання слову значення суми доданків. Наприклад, С : = С + А; С : = .

*Бінарні логічні мікрооперації* присвоюють слову значення, одержуване поразрядним застосуванням операцій до пар відповідних розрядів доданків. Наприклад, ; .

*Комбіновані мікрооперації* - це мікрооперації, що не належать жодному з перерахованих вище класів. Комбінована мікрооперація містить у собі кілька дій, властиві мікроопераціям різних класів. Наприклад, мікрооперація породжує зсув вліво з одночасним інвертуванням слова *А* і є комбінованою.

**Сумісність мікрооперацій.** Деякі з використовуваних у програмі мікрооперацій можуть виконуватися паралельно в часі, у той час як інші - тільки послідовно. Властивість сукупності мікрооперацій, що гарантує можливість їхнього паралельного виконання, називається *сумісністю.* Мікрооперації, що не володіють зазначеними властивостями, називаються *несумісними.* Сумісність мікрооперацій обумовлена, по-перше, змістом операторів, що представляють мікрооперації, - так звана *функціональна сумісність* - і, по-друге, структурою операційного пристрою, що допускає чи виключає можливість паралельного виконання декількох мікрооперацій, - так звана *структурна сумісність.* У функціональних мікропрограмах, що описують алгоритм виконання операцій безвідносно до структури пристрою, паралельно можуть виконуватися тільки ті мікрооперації, що мають властивість функціональної сумісності. Структура може внести обмеження на кількість паралельно виконуваних мікрооперацій. Тому можливість паралельного виконання мікрооперацій повинна визначатися виходячи зі структурної сумісності. Таким чином, якщо структура операційного пристрою не визначена, то сумісними називаються функціонально сумісні мікрооперації. Якщо структура задана, то сумісними називаються структурно сумісні мікрооперації.

Мікрооперації і , де - підмножини слів з S, називаються *функціонально сумісними,* якщо тобто якщо мікрооперації присвоюють значення різним словам. Так, мікрооперації - функціонально сумісні, але мікрооперації - функціонально несумісні і, отже, не можуть виконуватися паралельно в часі, тобто одночасно. Можливість паралельного виконання мікрооперацій, функціонально сумісних у смислі даного визначення, забезпечується за рахунок наступного принципу виконання мікрооперацій: спільне виконання мікрооперацій зводиться до одночасного обчислення значень двійкових виразів , після чого ці значення одночасно приписуються словам , зазначеним у лівих частинах операторів присвоювання. Умовою сумісності *т* мікрооперацій є сумісність кожної пари мікрооперацій.

Зазначене визначення функціональної сумісності мікрооперацій не є єдиним. Інше розуміння функціональної сумісності використовується в [46].

Мікрооперації називаються *структурно несумісними,* якщо через обмеження, породжуваних структурою операційного автомата, вони не можуть бути виконані спільно - в одному такті автоматного часу. У протилежному випадку мікрооперації є структурно сумісними. Структурна несумісність мікрооперацій пов'язана з використанням мікроопераціями спільного устаткування, одиничність якого виключає можливість спільного виконання мікрооперацій. Існують структури, що забезпечують сумісність усіх функціонально сумісних мікрооперацій, але всі інші структури вносять обмеження на сумісність окремих груп чи усіх без винятку мікрооперацій.

**Логічні умови.** Логічна умова - це булева функція , де - слова з множини S. Булева функція складається з первинних булевих виразів, що зв'язуються між собою знаками булевих операцій: - заперечення; - коньюнкция, - диз'юнкція; - заперечення рівнозначності. У якості первинних булевих виразів використовуються однорозрядні слова і поля, а також відносини. Відносини мають вид , де \* - знак операції відносини Прикладами логічних умов є наступні вираз:

*А* (0) - однорозрядне поле, значенням 0 і 1 якого відповідають такого ж значення логічної умови;

*А* (0) *А* (1) - умова приймає значення 0, якщо перемінні *А* (0), *А* (1) одночасно рівні 0, і значення 1, якщо хоча б одне перемінна має значення 1;

*А* (0) *А* (1) - умова приймає значення 0, якщо *А* (0) = *А* (1), і значення 1, якщо *А* (0)  *А* (1);

*А =*0 - умова, представлена відношенням рівності, приймає значення 0, якщо *A*  0, і значення 1, якщо *А* = 0;

*С* < 12710 - умову, представлена відношенням «не більше», приймає значення 0, якщо С > 127, і значення 1, якщо *С* 127. У мікропрограмуванні більш складні конструкції логічних умов зустрічаються дуже рідко.

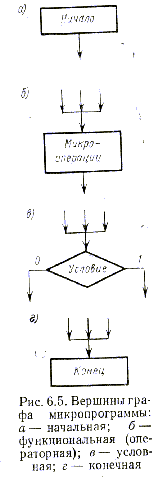
**Змістовний граф мікропрограми.** Для інженерних застосувань алгоритми виконання операцій у пристрої зручно описувати в графічній формі - змістовним графом мікропрограми. Граф будується з використанням вершин чотирьох типів (рис. 6.5) і дуг, що зв'язують вершини. *Початкова вершина* відзначає початок алгоритму і має єдиний вихід, з якого виходить дуга до першої виконуваної вершини графа Для позначення операторів алгоритму використовуються вершини двох типів: функціональні й умовні. *Функціональна (основна) вершана* визначає дія - сукупність функціонально сумісних мікрооперацій, виконуваних паралельно. Мікрооперації у вершині представляються у виді операторів присвоювання. У функціональну вершину може входити будь-яке, не менше 1 число дуг і з вершини виходить тільки одна дуга. *Умовна вершина* використовується для розгалуження обчислювального процесу в одному з двох можливих напрямків, вибір якого визначається поточним значенням логічної умови, зазначеного у вершині. Якщо умова має значення 0, обчислювальний процес розвивається по дузі, відзначеної символом 0, у противному випадку - по дузі, відзначеної символом 1. В умовну вершину може входити будь-як число дуг, але виходять завжди дві дуги. *Кінцева вершина* відзначає кінець мікропрограми. У кінцеву вершину може входити будь-як число дуг. Граф, що представляє мікропрограму, вважається коректним, якщо виконуються наступні умови:

1) граф містить тільки одну початкову і тільки одну кінцеву вершини;

2) у будь-яку вершину, крім початкової, повинна входити хоча б одна дуга, що виходить з іншої вершини графа;

3) з кожного виходу будь-якої вершини, крім кінцевої, повинна виходити одна дуга, що веде до деякої вершини графа;

4) при всіляких значеннях слів повинний існувати шлях з початкової вершини в кінцеву.



Умови 2 і 3 вимагають, щоб кожна вершина мала хоча б однієї попередню й однієї наступної вершини, причому функціональна вершина може мати тільки одну наступну вершину, а умовна - точно дві наступні вершини, що відповідають виходам 0 і 1 умовної вершини.

**Приклади функціональних мікропрограм.** Нехай операційний пристрій призначений для виконання тільки двох операцій F={\*, /} над двійковими числами з фіксованою комою, що представляються словами наступного формату:

0 1 15

|  |  |
| --- | --- |
|  | Цифрові розряди |

(6.6)

Результати операцій - добуток і частка - будемо представляти в такому ж форматі з використанням процедури округлення. Якщо ділене чи більше дорівнює дільнику, значення частки перевищує 1 і не може бути представлене у форматі (6.6). Цей випадок будемо відзначати виробленням ознаки переповнення.

Зосередимо увагу лише на процедурі побудови функціональних мікропрограм і тому будемо вважати, що можна використовувати будь-як алгоритми виконання операцій незалежно від витрат часу на їхню реалізацію.

Для множення будемо використовувати алгоритм обчислення добутку, починаючи від молодших розрядів множника, що складається з наступних дій [43]:

1. Добуток покладається рівним нулю.

2. Якщо молодший розряд множника дорівнює 1, добуток збільшується на значення модуля множеного.

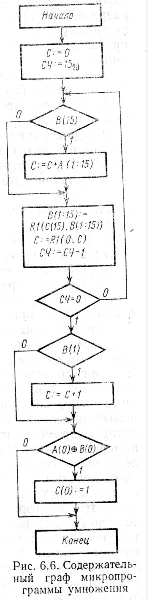
3. Добуток і множник зрушуються на один розряд вправо, у результаті чого в молодший розряд множника вводиться чергова цифра множника.

4. Дії 2, 3 повторюються для обробки всіх 15 цифр множника.

5. Добуток округляється за значенням 16-го розряду.

6. Якщо знаки співмножників однакові, добутку присвоюється знак плюс; якщо знаки різні, добутку присвоюється знак мінус.

Визначимо слова, що необхідні для виконання множення по описаному алгоритму. Операнди - множене і множник - будемо представляти словами А і В, формат яких визначається (6.6). Значення приписуються словам А і В поза алгоритмом до початку операції, тому ці слова повинні мати тип I (вхідні). Потім слова А і В використовуються в процесі виконання алгоритму, що відзначається присвоюванням їм типу L (внутрішні). Отже, слова А, В повинні мати тип IL. Значення добутку будемо представляти словом С (0:15). Щоб одержати округлений добуток, необхідно зберегти значення принаймні 16-го розряду добутку. Слово С (0:15) повинне мати тип LO, оскільки воно обробляється в мікропрограмі і по закінченні операції представляє значення результату, використовуване поза мікропрограмою. Оскільки всі цифри множника обробляються однаково, доцільно організувати цикл. Для визначення моменту завершення циклу необхідний лічильник числа повторень циклу. На початку операції лічильник встановлюється в стан 15 і після обробки кожної цифри його стан повинний зменшуватися на одиницю. Перехід лічильника в стан 0 свідчить про закінчення циклу. Щоб закодувати значення 15, необхідно 4 двійкових розряди, тобто лічильник повинний мати формат СЧ (1:4). Значення лічильника використовується тільки в мікропрограмі, тому слово СЧ повинне мати тип L (внутрішнє). Перераховані слова, що необхідні для реалізації операції множення, описані в табл. 6.4.

 Таблиця 6.4

**Опис слів у мікропрограмі множення**

|  |  |  |
| --- | --- | --- |
| Тип | Слово | Пояснення |
| IL | *А* (0:15) | Множене |
| IL | *В* (0:15) | Множник |
| LO | *С* (0:15) | Добуток |
| L | *СЧ* (1:4) | Лічильник циклів |

Функціональна мікропрограма, що описує алгоритм операції множення, складається з опису слів (табл. 6.4) п змістовного графа мікропрограми (рис. 6.6). Виконання операції починається з присвоювання добутку С нульового значення й установки лічильника *СЧ* у початковий стан 15. Ці мікрооперації є сумісними, що дозволяє об'єднати їх в один функціональний оператор. Обчислення добутку починається з аналізу значення молодшої цифри мкожителя. Якщо *В* (15) = 1, добуток *Зі* збільшується на значення модуля множеного, що представляється в розрядах 1-15 слова *А.* У наступному операторі виробляється зсув цифрових розрядів 1-15 множника *В* с метою виділення чергової цифри множника, зсув добутку С и зменшення значення лічильника на 1. Для збереження молодших розрядів добутку розряд С (15) передається в старший розряд слова *В, що* звільняється при зсуві множника. Щоб визначити необхідність повторення циклу вироблення добутку і момент закінчення формування значення добутку, перевіряється умова СЧ=0. До закінчення обробки 15 цифр множника лічильник знаходиться в стані, відмінному від 0. Тому умова має значення 0 («неправда») і в мікропрограмі виконується перехід на обробку наступної цифри. Коли *СЧ =* 0, вироблення добутку закінчується. Округлення результату виробляється додаванням 1 до 15-му розряду слова *С,* якщо розряд *В* (1) = 1. Значення логічної умови *А* (0) *В* (0) визначає знак добутку. Якщо *А* (0) В (0) = 0, знаки співмножників однакові й у знаковому розряді *С* (0) добутку зберігається значення 0, що відповідає знаку плюс. Якщо *А* (0) *В* (0) = 1, у знаковий розряд *С* (0) добутку заноситься значення 1, що відповідає знаку мінус.

Для ділення чисел з фіксованої коми будемо використовувати алгоритм без исправляюш.їхніх додавань [43]:

1. Якщо дільник дорівнює нулю, фіксується переповнення при розподілі.

2. Перевіряється можливість ділення. Для цього з діленого віднімається дільник. Якщо залишок не негативний, то частка більше 1 (переповняє розрядну сітку) і розподіл неможливий.

3. Залишок подвоюється зсувом уліво.

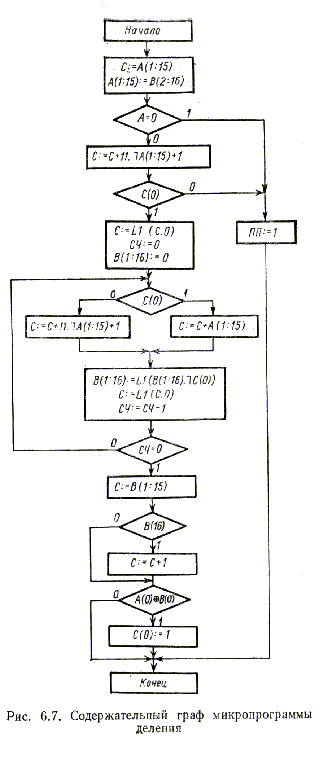
4. Якщо залишок позитивний, то дільник віднімається з залишку; якщо залишок негативний, то дільник суммируется з залишком.

5. Якщо новий залишок позитивний, чергова цифра частки дорівнює 1; якщо залишок негативний, цифра частки дорівнює 0.

6. Дії 3-5 повторюються до вироблення 16 цифр частки.

7. Частка округляється за значенням 16-го розряду.

8. Частці присвоюється знак плюс, якщо знаки операндов однакові, і знак мінус, якщо знаки операндов різні.



Визначимо набір слів, необхідний для виконання операції ділення. Як і при множенні, операнди -ділене і дільник - будемо представляти словами А и *В, а* частка - словом *С.* Для організації циклу використовується лічильник *СЧ* (1:4). При складанні функціональних мікропрограм прагнуть зменшити число мікрооперацій. Цього можна досягти, якщо при виконанні ділення використовувати в основному ті ж мікрооперації, що застосовувалися для множення. З цією метою приймемо, що в процесі виконання операції залишок представляється словом С, дільник - словом *А* и частка формується в слові *В.* У процесі ділення залишок подвоюється і може приймати значення з діапазону . Отже, для розміщення залишку потрібно 16 цифрових розрядів і знаковий розряд, тобто слово *С* повинно бути 17-розрядним. Щоб одержати округлений результат, необхідно виробити 16 цифр частки, для розміщення яких слово *В,* використовуване для представлення частки в ході операції, повинне мати 16 цифрових розрядів.

У процесі ділення може виникнути переповнення. Ознака переповнення будемо фіксувати в однорозрядному слові *ПП.* Перераховані слова, необхідні для реалізації операції ділення, описані в табл. 6.5.

Змістовний граф мікропрограми ділення приведений на рис. 6.7. Розподіл починається з переміщення операндов: ділене *А* передається в С и дільник - в А. Знаки операндов зберігають своє первісне положення і займають кульові розряди слів *А* и *В.* При виконанні передачі *С : = А* (1:15) знаковий розряд слова *С* приймає значення 0. Потім перевіряється на нуль значення дільника. Якщо цифрові розряди дільника рівні кулю, фіксується переповнення *ПП* : = 1. При ненульовому дільнику виробляється спробне віднімання, для чого до 17-розрядного слова С (діленому) додається додатковий код дільника 11. . Значення 11, що додається до оберненого коду , представляє негативний знак і інверсію нуля,

заносимо; про у «вільний» розряд С (1) слова *С.* Знак різниці представляється значенням С (0). Якщо С(0)=0, ділене не менше значення дільника і розподіл неможливий, що відзначається виробленням ознаки переповнення *ПП* : = 1. Якщо знак різниці негативний - *С* (0) = 1 , починається вироблення 16 цифр частки.

Таблица 6.5

**Опис слів у мікропрограмі ділення**

|  |  |  |
| --- | --- | --- |
| Тип | Слово | Пояснення |
| IL | *А*(0:15) | Ділене |
| IL | *В*(0:16) | Дільник |
| LO | *С*(0:16) | Залишок (частка) |
| L | *СЧ* (1:4) | Лічильник циклів |
| LO | *ПП* | Ознака переповнення |

Попередньо залишок подвоюється зсувом уліво, лічильник і слово *В,* у який буде формуватися частка, встановлюються в початковий стан. Цифра частки виробляється шляхом чи віднімання додавання залишку і дільника, що реалізується мікроопераціями *С :* = *С +* 11. і *С : = С* + *А* (1:15). Знак знову отриманого залишку визначає чергову цифру частки, значення якої протилежно коду знака, тобто цифра частки дорівнює . Цифри частки виробляються послідовно, починаючи від старших розрядів. Тому для формування частки використовується зсув слова *В* с уведенням значення в крайній правий розряд слова. Ця дія виконується мікрооперацією *В* (1:15) : = L1 *(В* (1 : : 16). ). Визначення моменту закінчення циклу вироблення 16-розрядної частки забезпечується в такий спосіб. Лічильнику *СЧ* спочатку присвоюється значення 0. Перед першою перевіркою умови *СЧ* = 0 з лічильника віднімається одиниця: *СЧ :* = *СЧ* - 1. Оскільки лічильник містить обмежене число розрядів, різниця (0000-1) буде мати значення 1111 = 1510, тобто лічильник зі стану 0 перейде в стан 15. Після вироблення 16 цифр частки, розташовуваних у розрядах 1-16 слова *В,* старших 15 розрядів передаються в слово *С,* у якому повинен розміщатися результат, виробляється округлення за значенням молодшого розряду частки В(16) і в разі потреби результату присвоюється знак мінус.

**Тема: ФУНКЦІЇ ОПЕРАЦІЙНОГО І КЕРУЮЧОГО АВТОМАТІВ**

Функціональні мікропрограми , що описують алгоритми виконання операцій  у пристрої, несуть у собі інформацію про функції операційного і керуючого автоматів. На даному етапі розгляду принципів побудови і проектування операційних пристроїв обмежимося визначенням форми, у якій представляються функції автоматів, не торкаючись техніки виконання перетворень, що дозволяє оптимізувати функції автоматів.

**Функція операційного автомата.** Як відзначалося в п. 6.2, функція операційного автомата визначається множиною слів S, з якими оперує автомат, множиною мікрооперацій Y і множиною логічних умов X, необхідних для реалізації набору операцій . Множини S, *Y, X* визначаються з функціональних мікропрограм у такий спосіб.

Кожна функціональна мікропрограма *,* g = 1, ..., G, що визначає алгоритм виконання операції *,* вводить множини слів*,* мікрооперацій і логічних умов. Множини , , характеризують функцію операційного автомата частково - у частині засобів, достатніх для реалізації тільки однієї операції . Множини *S, Y, X,* достатні для реалізації всіх операцій з набору , визначаються об'єднанням множин , , відповідно:

; ;

Найбільш складною є задача об'єднання слів з метою визначення множини слів S, необхідних операційному автомату для реалізації операцій *F.* Очевидно, витрати устаткування в операційному автоматі будуть тим меншими, чим менше слів містить множина S, а точніше, чим менше сумарне число розрядів у словах із S. Якщо число слів у множинах дорівнює відповідно *N1,* ..., *NG,* то для виконання будь-якої операції досить слів, тобто потужність множини S визначається максимальною потужністю об’єднаних множин . Отже, при об'єднанні слів необхідно ототожнювати між собою слова, що належать об’єднуваним множинам. Коли функціональні мікропрограми складаються незалежно одна від іншої, для ототожнення слів потрібно спеціальна процедура, що дозволяє ототожнювати слова з різними ідентифікаторами і довжинами таким чином, щоб мінімізувати витрати устаткування в операційному автоматі. Така процедура описана в [8]. А поки будемо припускати, що функціональні мікропрограми складаються з урахуванням об'єднання їх в операційному пристрої й у зв'язку з цим тотожні слова, що використовуються в різних мікропрограмах, ідентифікуються однаковими іменами, як у табл. 6.4, 6.5. У такому випадку процедура об'єднання множин зводиться до перерахування в множини S слів з попарно різними ідентифікаторами, причому, якщо два слова з однаковими ідентифікаторами мають різну довжину, то в множини S уводиться тільки слово з максимальною довжиною і припускається, що слово меншої довжини може бути представлене в більшому форматі. Так, з зіставлення табл. 6.4 і 6.5 видно, що множина слів, перерахована в табл. 6.5, забезпечує представлення всіх слів з табл. 6.4, тобто набір слів, використовуваний в алгоритмі ділення, достатній для реалізації множення.

Таблиця 6.6

**Мікрооперації, що використовуються в мікропрограмах множення і ділення**

|  |  |
| --- | --- |
| Позначення | Мікро операція |
|  | C : =0 |
|  | СЧ : =1510 |
|  | C : = C+A (1 : 15) |
|  | B (1 : 15) : = R1 (C (16). B (1: 15)) |
|  | C : = R1 (0.C) |
|  | СЧ : = СЧ – 1 |
|  | C : = C + 1 |
|  | C (0) : = 1 |
|  | C : = A (1 : 15) |
|  | A : = B (1 : 15) |
|  | C : = C + 11. A (1:15) + 1 |
|  | ПП : = 1 |
|  | C : = L1 (C.0) |
|  | СЧ : = 0 |
|  | B : = 0 |
| *у*16 | B (1:16) : = L1 (B (1:16 ). C (0)) |
|  | C : = B (1 : 15) |

Множина мікрооперацій ,що використовуються у мікропрограмах , виходять шляхом вибірки зі змістовних графів алгоритмів операторів присвоювання, що представляють мікрооперації. Множина *Y* визначається об'єднанням множин , тобто перерахуванням усіх попарно помітних мікрооперацій з множин . У табл. 6.6 приведений об'єднаний список мікрооперацій, що використовуються у мікропрограмах (рис. 6.5, 6.6). Мікрооперації використовуються в мікропрограмі множення (рис. 6.6) і мікрооперації  - у мікропрограмі ділення (рис. 6.7).

Таблиця 6.7

#### Логічні умови, що використовуються в

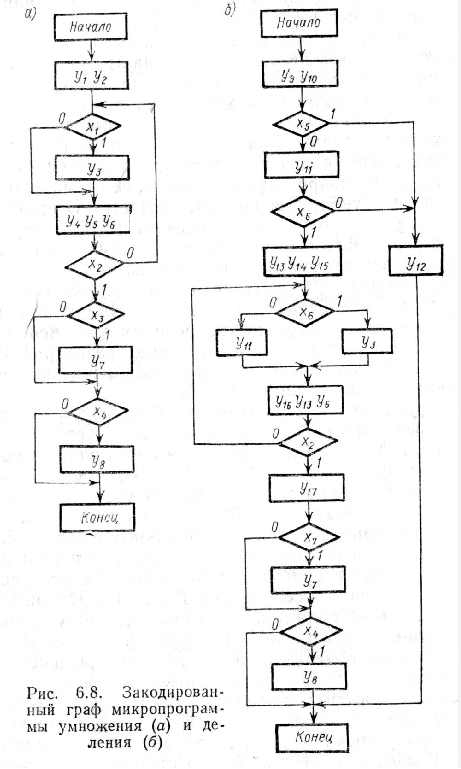
**мікропрограмах множення і ділення**

|  |  |
| --- | --- |
| Позначення | Логічна умова |
|  | B (15) |
|  | СЧ = 0 |
|  | B (1) |
|  | A (0)  B (0) |
|  | A (1 : 15 ) = 0 |
|  | C (0) |
|  | B (16) |

Аналогічно визначається множина логічних умов *X,* достатня для реалізації операцій . У табл. 6.7 приведений список логічних умов, що використовуються у мікропрограмах множення (умови ) і ділення (умови .

Таким чином, функція операційного автомата представляється таблицями, що містять опис слів, список мікрооперацій і список логічних умов, зокрема табл. 6.5-6.7.

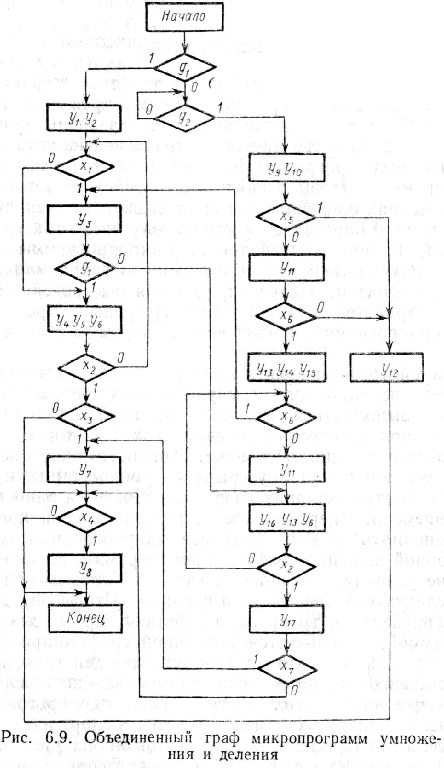
**Закодований граф мікропрограми**. Стосовно керуючого автомату мікрооперації і логічні умови варто розглядати як елементарні символи, що автомат інтерпретує у виді керуючих сигналів і сприймає як інформаційні сигнали *.* Тому функція керуючого автомата представляється у формі операторной схеми мікропрограми (алгоритму), у якій символи відіграють роль операторів, а символи  *-* роль логічних умов. Закодована граф-схема мікропрограми, що є однією з форм операторних схем, виходить зі змістовного графу мікропрограми шляхом заміни мікрооперацій, зазначених в операторних вершинах, на відповідні їм символи і заміни логічних умов, що містяться в умовних вершинах, на відповідні символи *.* Граф мікропрограми, відзначений символами і *,* що ототожнюються з керуючими й інформаційними сигналами, називається *закодованим графом мікропрограми.* На рис. 6.8 представлені закодовані графи мікропрограм множення і ділення, отримані на основі змістовних графів рис. 6.6 і 6.7 з використанням табл. 6.6, 6.7, що визначає найменування мікрооперацій і логічних умов.



**Функція керуючого автомата**. Функція керуючого автомату визначається сукупністю закодованих графів мікропрограм , що відповідають змістовним графам мікропрограм *.* У принципі на основі графів  можна синтезувати *G* керуючих автоматів, що забезпечать керування операційним автоматом. Однак таке рішення є неоптимальним з наступної причини. Графи різних мікропрограм містять однакові операторні й умовні вершини, що зіллються, якщо побудувати граф Г, що поєднує в собі графи . Наприклад, у мікропрограмах множення і ділення (рис. 6.7) округлення і формування знака результату виробляється однаковим образом - з використанням тих самих мікрооперацій і логічних умов. Мікропрограми додавання і віднімання відрізняються лише оператором, що інвертує при відніманні знак другого операнда. Отже, якщо - число вершин у графах мікропрограм , то число вершин в об'єднаному графі *.* Можна екати, що зменшення числа вершин у мікропрограмі спричинить зменшення витрат устаткування в керуючому автоматі. У зв'язку з цим витрати устаткування в автоматі, побудованому по об'єднаній мікропрограмі Г, у загальному випадку виявляються менше сумарних витрат устаткування в G автоматах, кожний з який реалізує відповідну мікропрограму . Таким чином, функцію керуючого автомата доцільно представляти у виді закодованого графа, що є об'єднанням закодованих графів мікропрограм, що описують алгоритми виконання окремих операцій.

Метод об'єднання граф-схем алгоритмів у єдину графу-схему Г, що містить мінімальне число операторних і умовних вершин, детально викладений у [6]. В об'єднаній мікропрограмі шляхи розвитку процесу обчислень, що відповідають різним операціям , задаються набором перемінних , за допомогою яких кодуються операції. Так, для кодування 10 операцій потрібно 4-розрядний код значення якого 0001, 0010, 0011, ..., 1010 відповідають операціям реалізованим операційним пристроєм. Двійкові перемінні відіграють роль логічних умов, що визначають переходи в об'єднаній мікропрограмі [6].

На рис. 6.9 приведений об'єднаний граф мікропрограм множення і ділення. Операції кодуються перемінними код = 10 відповідає множенню і код = 01 - ділення. Сумарне число операторних і умовних вершин у графах поєднуваних мікропрограм (рис. 6.8) дорівнює 25. В об'єднаній мікропрограмі їхнє число зменшилося до 23. При об'єднанні великого числа мікропрограм створюється більше можливостей для злиття фрагментів різних мікропрограм, у результаті чого число вершин в об'єднаній мікропрограмі виявляється значно менше сумарного числа вершин у поєднуваних мікропрограмах.



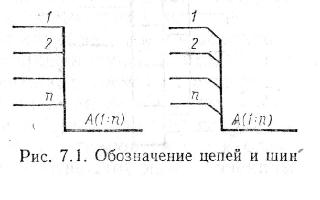
**Тема: СТРУКТУРНИЙ БАЗИС ОПЕРАЦІЙНИХ АВТОМАТІВ (OA)**

Функція ОА зводиться до введення-виведення і збереження слів інформації, виконанню мікрооперацій (МО) і обчисленню логічних умов (ЛУ). Щоб реалізувати ці дії, необхідний набір елементів, достатній для побудови структур із заданими функціями. Набір елементів, на основі яких можуть будуватися структури з визначеними властивостями, називається *структурним базисом* (елементною базою). Структурний базис OA повинен містити елементи, що забезпечують передачу і збереження слів інформації й обчислення значень функцій, на основі яких будуються МО і ЛУ. Для цих цілей найбільш широко використовується наступний набір елементів: *шини, що* забезпечують передачу слів інформації; *регістри, що* забезпечують збереження слів; *KC,* за допомогою яких обчислюються значення функцій.

**Шини**. Для передачі біта інформації (двійкової змінної) необхідний один ланцюг. Сукупність ланцюгів, що використовуються для передачі слова, називається *шиною.* Приймемо, що шина одержує найменування слова, що передається по ній, а ланцюги шини нумеруються так само, як розряди слова. Так, слово A(1:n)передається по шині А(1:n), щоскладається з nланцюгів з номерами 1, ..., n*.* Ланцюги і шини зображуються на схемах у виді рис. 7.1. Ланцюги ідентифікуються номерами чи найменуваннями розрядів слова, наприклад найменуваннями А(1), ..., А(n)*.*

Для реалізації МО передачі виду В:=Авикористовуються *керовані шини, що* позначаються на структурних схемах, як показано на рис. 7.2, *а, б.* Тут *А* - вхід шини; B- вихід шини; *yt* - керуючий сигнал, що ініціює MO передачі *В : = А.* Символ «» чи стрілка позначають точку застосування керуючого сигналу. Керована шина реалізує наступну функцію:

тобто при y1=0 передача слова інформації заборонена, а при y1=1 вихід шини приймає значення слова, що представлене на вході. Керована шина улаштована в такий спосіб. Нехай ajі bj- розряди j слів А i В*.* Сигнал bj, щопредставляє значення розряду bj*,* може прийняти значення 1 тільки в тому випадку, якщо aj=1 й одночасно yi=1, тобто значення bj є кон’юнкцією значень aj і yi: bj=ajyi*.* Отже, n-розрядна керована шина - це сукупність з *п* двовходових елементів І (рис. 7.2, в). На функціональних схемах зазначена сукупність елементів може позначатися, як показано на рис. 7.2*.* Керована шина вносить затримку  в процес передачі слова, де  - затримка сигналу на логічному елементі.

За допомогою шини слово можна передавати від одного джерела до декількох приймачів. У ряді випадків до одного приймача *В* повинні передаватися різні слова A1, …, AK*.* У кожен момент часу приймач може прийняти тільки одне слово, так що різні слова можуть передаватися, лише чергуючись у часі. Отже, кожна із шин A1, …, AK,щопідключається до загального приймача, повинна бути керованою і по сигналу yk, k=1, …, Kреалізувати передачу B:=Ak. Сукупність керованих шин із загальним виходом називається *мультиплексором.* На структурних схемах мультиплексор зображується як показано на рис. 7.3, *а.* Функція мультиплексора

тобто зводиться до *К* МО передачі B:=A1, …, B:=AK,що ініціюються керуючими сигналами y1, …, yK*.* Ці МО є несумісними: у кожен момент часу може бути реалізована тільки одна з них. Неважко бачити, що значення розряду вихідного слова є наступною функцією значень керуючих сигналів y1, …, yK і однойменних розрядів слів A1, …, AK:

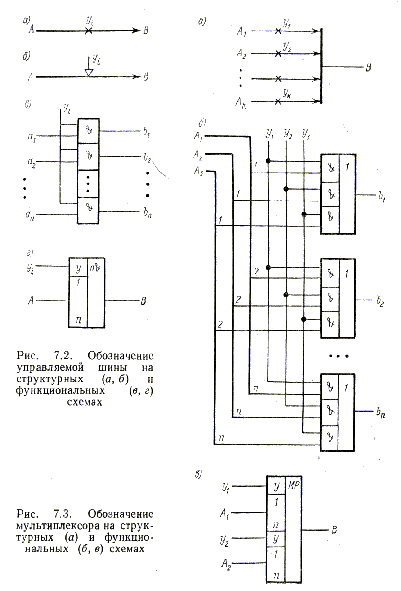
(7.1)

Функціональна схема тривходового мультиплексора, побудована по (7.1), приведена на рис.7.3, б. Схема складається з *п* елементів І-АБО, кожен з яких формує сигнал у вихідному ланцюзі мультиплексора. Затримка передачі слова в мультиплексорі дорівнює 2, де  - затримка сигналу на одному логічному елементі. Позначення мультиплексора, як елемента функціональної схеми, приведене на рис. 7.3, *в.*

*Регістром* називається сукупність запам'ятовуючих елементів, призначена для збереження слова інформації. В якості запам'ятовуючих елементів в регістрах використовуються тригери. *Тригер* - це логічна (перемикальна) схема з двома стійкими станами, що позначаються як 0 і 1. Стан тригера *Т*, як правило,представляється двома вихідними сигналами s і : у стані 0 сигнал s = 0 і = 1 і в стані 1 сигнал s = 1 і = 0. Коли стан тригера відображається парою сигналів s і *,* говорять, що стан (двійкова змінна) s представляється в *парафазному**коді.* Вихід тригера s називається *прямим*, а вихід - *інверсним*. Тригер переключається з одного стану в інший під впливом вхідних сигналів, число і дія яких на стан тригера визначаються типом тригера.

Тригери розділяються на два класи: синхронні й асинхронні. *Синхронний тригер* має синхронізуючий вхід Сдля подачі сигналу, що відзначає хід реального часу і дозволяє переключення тригера, тільки у визначені моменти часу. Якщо синхронізуючий сигнал С = 0, тригер зберігає свій стан незалежно від значень сигналів на його входах. Тригер переключається в стан, що відповідає значенням вхідних сигналів, тільки в момент надходження сигналу С*=*1. Тригер, що не має синхронізуючого входу, називається *асинхронним.* Асинхронний тригер переключається в той момент коли змінюється значення сигналів на його вході.

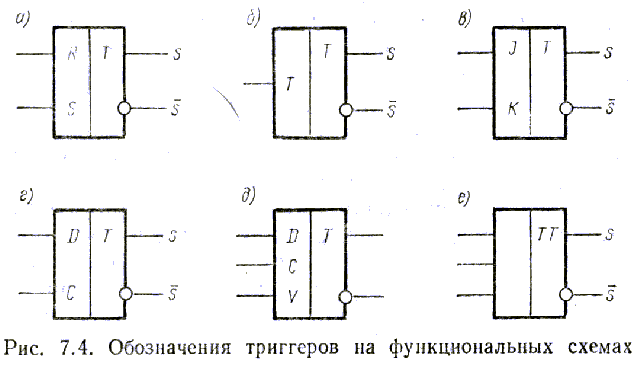
Стан тригера змінюється (переключається), коли на вхід тригера впливає сигнал зі значенням 1. У залежності від впливу на стан тригера вхідні сигнали позначаються в такий спосіб: R - установка тригера в 0 (скидання); S - установка в 1; Т - переключення в протилежний (обернений) стан (з 0 у 1 і з 1 у 0); J - установка в 1 чи протилежний стан; К - установка в 0 чи протилежний стан; D - установка в стан, що відповідає значенню D (D=0 - у стан 0, D=1 - у стан 1); V - дозвіл переключення в стан, обумовлений значенням інших вхідних сигналів. Тригери можуть бути оснащені різними наборами входів, у залежності від яких виділяють наступні основні типи тригерів: RS-тригери, Т-тригери, JK-тригери, D-тригери, DV-тригери.



*RS-тригер* (рис. 7.4, *а)* переключається під впливом вхідних сигналів *R, S* відповідно до табл. 7.1. У таблиці переходів на перетинанні рядка і стовпця зазначене стан, у яке переключається тригер під впливом набору вхідних сигналів. З табл. 7.1 видно, що при *R =* 0; S = 0 тригер зберігає свій стан, сигнал S = 1 при *R =* 0 переключає тригер у стан 1 і сигнал *R* = 1 при S = 0 - у стан 0. Сигнали S = 1 і *R =1* переводять тригер у невизначений стан, і, отже, така комбінація сигналів неприпустима. У синхронному RS-тригері переключення можливе тільки при З=1.

*Т-тригер* (рис. 7.4, б) переключається відповідно до табл. 7.2. Сигнал *Т* = 1 викликає перехід у протилежний стан. У синхронному Т-тригері переключення можливе тільки при *С =* 1.

*JK-тригер* (рис. 7.4, в) переключається сигналами J і K відповідно до табл. 7.3. При значеннях вхідних сигналів *JК =* 00; 01; 10 цей тригер функціонує так само, як RS-тригер (вхід *J* аналогічний *S,* і вхід *До* аналогічний *R),* але при одночасному надходженні сигналів *J =* 1 і K = 1 тригер переключається



у протилежний стан, тобто працює як Т-тригер. Отже, при замиканні входів *J* і *До* між собою JK-тригер перетворюється в Т-тригер.

*D-тригер* (рис. 7.4, *г)* переключається під впливом сигналу *D* відповідно до табл. 7.4. Сигнал *D* = 0 переключає тригер у стан 0 і сигнал *D* = 1 - у стан 1. Синхронізуючий сигнал *З* визначає момент переключення тригера.

*DV-тригер* (рис. 7.4, *д)* переключається під впливом сигналів *D* і *V* відповідно до табл. 7.5. Сигнал *V* є що дозволяє: при *V* = 0 переключення тригера заборонене, а при *V* = 1 тригер приводиться в стан, рівний значенню сигналу *D.*

Таблиця 7.1

Таблиця переходів RS-тригера

|  |  |  |
| --- | --- | --- |
| RS | Стан | |
| 0 | 1 |
| 00  01  10  11 | 0  1  0  - | 1  1  0  - |

Таблиця 7.2

## Таблиця переходів Т-тригера

|  |  |  |
| --- | --- | --- |
| T | Стан | |
| 0 | 1 |
| 0  1 | 0  1 | 1  0 |

Таблиця 7.3

## Таблиця переходів JK-тригера

|  |  |  |
| --- | --- | --- |
| JK | Стан | |
| 0 | 1 |
| 00  01  10  11 | 0  0  1  1 | 1  0  1  0 |

У процесі переключення тригера вхідні сигнали не повинні змінювати своїх значень. Може виявитися, що значення вхідних сигналів залежать від станів тригерів. У такому випадку в процесі переключення тригерів вхідні сигнали змінюють свої значення, у результаті чого тригери можуть знаходитися в довільних станах.

Таблиця 7.4

## Таблиця переходів D-тригера

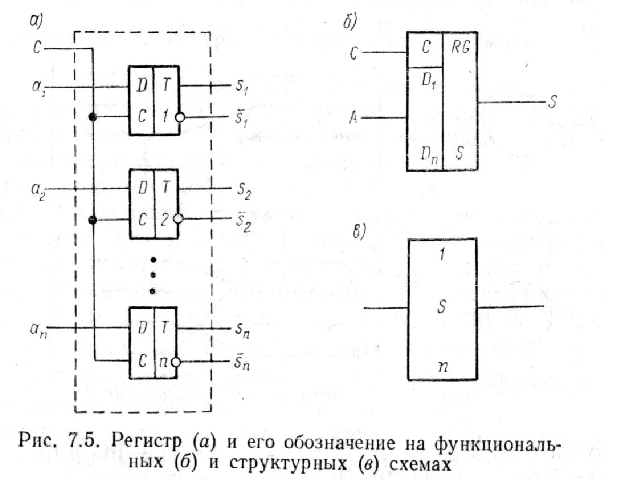
|  |  |  |
| --- | --- | --- |
| D | Стан | |
| 0 | 1 |
| 0  1 | 0  1 | 0  1 |

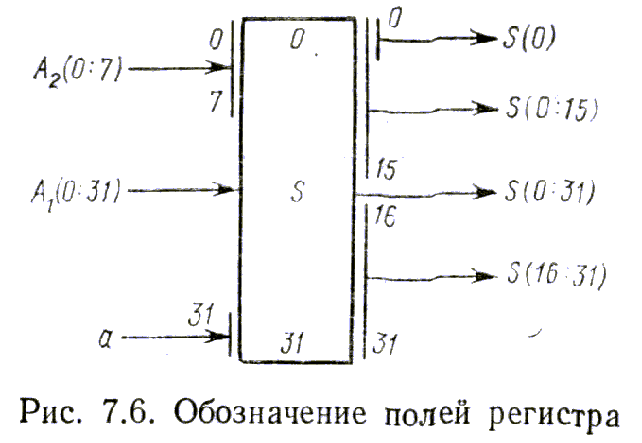
Таблиця 7.5

## Таблиця переходів DV-тригера

|  |  |  |
| --- | --- | --- |
| VD | Стан | |
| 0 | 1 |
| 00  01 | 0  0 | 1  1 |
| 10  11 | 0  1 | 0  1 |

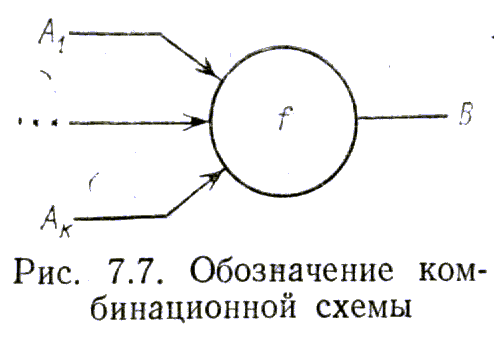
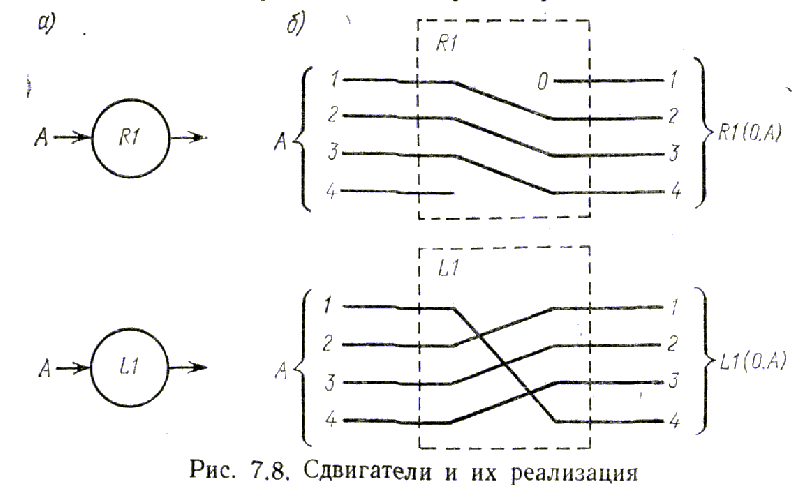
Для роботи в таких умовах використовуються подвійні тригери - тригери типу MS, що складаються, принаймні, із двох RS- чи D-тригерів, певним чином з'єднаних між собою [3, 14]. Тригер типу MS переключається в стан, що характеризується значеннями вхідних сигналів, що існували на момент надходження синхронізуючого сигналу, і не залежить від значень вхідних сигналів у період переключення тригера, обумовлений тривалістю синхронізуючого сигналу. Подвійний тригер позначається на функціональних схемах, як показано на рис. 7.4, *е.*



На рис. 7.5, *а* приведена функціональна схема регістра RG на синхронних D-тригерах. Слово завантажується в регістр із вхідних ланцюгів по сигналу С*,* щодозволяє переключення тригерів у стани, обумовлені значеннями сигналів *.* Позначення n-розрядного регістра на функціональних схемах показане на рис. 7.5, *б.* Приймається, що регістр отримує найменування слова, що зберігається в ньому, і розряди регістра (запам'ятовуючі елементи) нумеруються так само, як розряди слова. Так, слово S(1:n)зберігається в регістрі *S,* розряди якого нумеруються від 1 до n. Збережене в регістрі слово представляється на вихідній шині, найменування якої, обумовлене найменуванням переданого слова, співпадає з найменуванням регістра. На структурних схемах регістр позначається, як показано на рис. 7.5, б, де S - найменування регістра; 1 і n - номера старшого і молодшого розрядів.

Частина регістра, що співпадає з полем слова, називається *підрегістром* і може розглядатися як самостійний регістр. У регістр інформація може завантажуватися окремими полями. Порядок підключення до регістра шин, по яких передаються поля слів, позначається на структурних схемах, як показано на рис. 7.6. Аналогічним образом виділяються поля слова, збереженого на регістрі.

**KC.** Для обчислення значень слів і булевих значень використовуються KC, що складаються з логічних елементів, що реалізують БФ універсальних елементів, що реалізують функцію І-НІ (АБО-НІ). KC, що реалізує функцію над словами A1, …, AK з метою обчислення слова чи булевої змінної В,будемо позначати на структурних схемах, як показано на рис. 7.7. KC може реалізувати будь-яку обчислювану функцію f. Найбільш часто KC використовуються



для реалізації функцій , що використовуються для обчислення значень слів, і для обчислення значень ЛУ на основі наступних відносин: і т.д. З метою збільшення наочності структурних схем на додаток до перерахованих функцій , введемо фіктивні KC з функціями Rk і Lk, щозабезпечують зсув слова вправо і вліво на *k* розрядів. Їхнє позначення приведене на рис. 7.8, *а.* Як видно з рис. 7.8, *б,* зсув слова *А* на фіксоване число розрядів виробляється відповідною комутацією ланцюгів без використання логічних елементів, тобто витрати устаткування в схемах *Rk* і *Lk* дорівнюють нулю.

**Тема: КАНОНІЧНА СТРУКТУРА ОА ТА ЇЇ ВЛАСТИВОСТІ**

Структура ОА синтезується в заданому структурному базисі виходячи з функцій, реалізація яких покладається на автомат. Функція автомата визначається множинами слів, МО і ЛУ, що представляються, наприклад, у формі табл. 6.4-6.6.

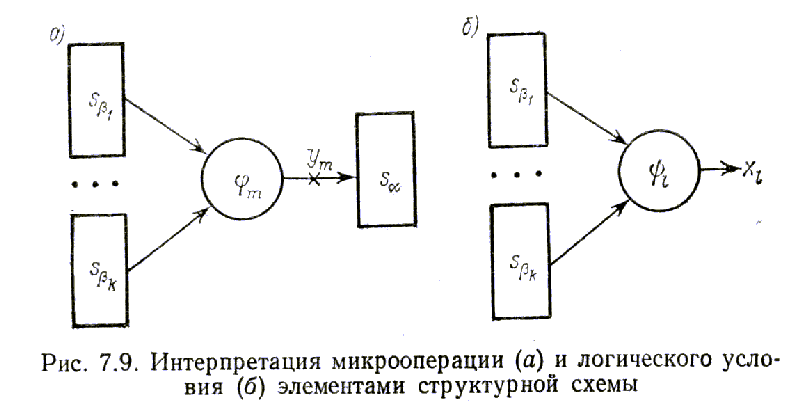
**Синтез канонічної структури.** Структуру ОА можна синтезувати безпосередньо по функції, що задається: 1) множиною слів S={s1, …, sN}, що можуть бути вхідними, вихідними і внутрішніми; 2) множиною МО , де m=1, ..., *М* i *;* 3) множиною ЛУ , де l=1,…,L.Структура автомата синтезується в такий спосіб:

1. Словам s1, …, sN, що описані в якості внутрішніх (тип L), ставляться у відповідність регістри s1, …, sNз довжинами n1, …, nN, що дорівнюють довжинам слів. Якщо слово si розділяється на поля, то в регістрі si виділяються відповідні підрегістри, що використовуються для збереження розрядів полів.

2. Словам , щоописані в якості вхідних (тип I), ставляться у відповідність вхідні полюси (входи) d1, …, dHструктурної схеми. Кожен вхід d1, …, dHз'єднується з регістром шиною, що виходить із входу.

3. Словам , щоописані в якості вихідних (тип О), ставляться у відповідність вихідні полюси (виходи) r1, …, rQ структурної схеми. Кожен регістр з'єднується з виходом r1, …, rQ шиною, що виходить з регістра.

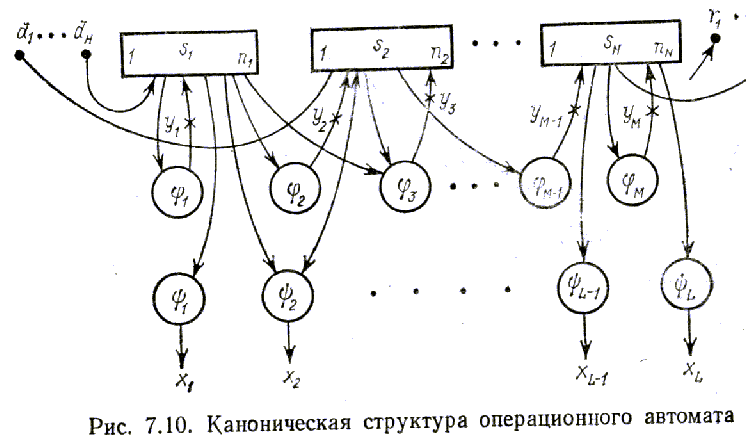
4. Кожній МО ymY, щоописується оператором присвоювання , ставиться у відповідність KC m (рис. 7.9, а)*,* входи якої підключаються до регістрів , а вихід з'єднується керованою шиною з регістром s. Керована шина відзначається сигналом ym, щоініціює MO - присвоювання слову s значення .



Для виконання МО передачі s:= s не потрібна KC, що обчислює значення двійкового виразу. Тому структурна реалізація МО передачі забезпечується керованою шиною, що з'єднує регістр s із регістром s і відзначена відповідним керуючим сигналом. Аналогічно, MO установки s:=const реалізується керованою шиною, початок якої відзначається константою const і відповідним керуючим сигналом.

5. Кожній ЛУ l=1,…,L ставиться у відповідність KC l (рис. 7.9, б), входи якої з'єднуються з регістрами ,а вихід відзначається інформаційним сигналом xl*.* Якщо l - тривіальна БФ (наприклад, *,* де p- номер розряду слова s), ЛУ інтерпретується шиною (ланцюгом), що виходить з розряду p регістра s. Вихід шини представляє значення інформаційного сигналу xl*.*

Структура ОА, що виникає в результаті заміни кожного елемента функції (слова, МО, ЛУ) відповідними елементами структурного базису (шинами, регістрами, KC), є основною для синтезу інших структур і називається *канонічною структурою.* Канонічна структура має вид, представлений на рис. 7.10. Сукупність регістрів s1, …, sN утворює пам'ять ОА, у якій зберігаються вихідні значення, проміжні і кінцеві результати. Вихідні значення завантажуються в регістри через входи d1, …, dH*.* Кінцеві результати виводяться з пам'яті ОА через виходи r1, …, rQ. Сукупність KC 1, …,M і пов’язаних з ними керованих шин, що відзначені сигналами y1, …,yM,служить для виконання МО. Значення ЛУ обчислюються KC 1, …, L, на виходах яких визначені значення інформаційних сигналів x1, …, xL*.*



На рис. 7.11 приведений варіант канонічної структури для множення і ділення 16-розрядних чисел з фіксованою комою. Схема (рис. 7.11) побудована на основі функцій ОА, що описані у табл. 6.4-6.6. Вихідні ланцюги регістрів А, В, С, СЧоб'єднані в шину S, з якої сигнали надходять на виходи KC, що реалізують МО й обчислюють значення ЛУ. Передбачається, що по шині S значення слів, збережених на регістрах, передаються в парафазному коді. При цьому

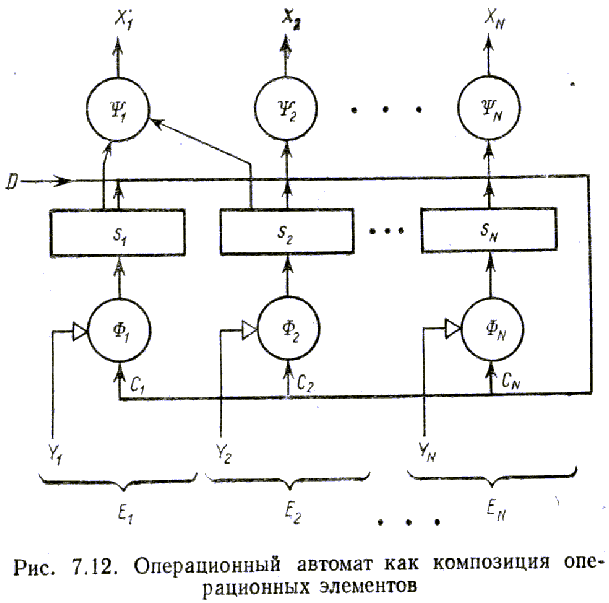
кожна двійкова змінна, збережена в тригері відповідного регістра, представляється як прямим, так і оберненим (інверсним) значенням. З цієї причини інверсії знімаються із шини, по якій передаються значення слів А, В, С, СЧ.Керовані шини y1, …, y17 підключаються до регістрів А, В, С, СЧчерез мультиплексори, що поєднують декілька шин в одну.

Відповідно до моделі (рис. 6.2) ОА розділяється на три частини: пам'ять S; KC Ф, що реалізує функції 1, …, M, пов'язані з виконанням МО Y;KC , що обчислює значення ЛУ X*.*

Канонічну структуру на рис. 7.10, 7.11 можна представити в наступному вигляді (рис. 7.12). Множина МО Y розділяється на підмножини , кожна з яких складається із сукупності МО, що обчислюють значення того самого слова s1, …, sN*.* Різні множини Yiі Yjне містять спільних МО і тому KC Ф можна розділити на незалежні підсхеми Ф1, …, ФN, що реалізують підмножини МО Y1, …, YN*.* Підсхеми Ф1, …, ФN обслуговують відповідні регістри s1, …, sN і у кожному такті можуть реалізувати по одній МО *.* Аналогічно, схема , що використовується для обчислення значень ЛУ X, може бути розділена на підсхеми 1, …, N, що обчислюють значення підмножин ЛУ Х1, …, ХN*.* Схему, що складається з регістра snі комбінаційних частин Фn і n, можна розглядати як елементарний ОА, що обчислює значення одного слова інформації і контролює його за допомогою ЛУ. Елементарний ОА називається *операційним елементом.* Таким чином, у загальному випадку ОА розділяється на сукупність операційних елементів Е1, …, ЕN,число яких визначається кількістю внутрішніх слів, що оброблюються мікропрограмою.



Трохи видозмінена модель структурної організації OA описана в [55]. У цій моделі вхідні слова D={d1, …, dH}можуть використовуватися як аргументи МО, а вихідні слова R={r1, …, rQ}можуть визначатися як результати МО.



**Функціонування OA.** ОА функціонує в часі в порядку, обумовленому KA. Процес функціонування розділяється на інтервали часу, що називаються *тактами.* Протягом такту виконуються наступні дії: 1) KA формує набір з одного, двох чи більше сигналів ya, …, ywY;2) під впливом керуючих сигналів ya, …, ywуKC-ах Ф автомату обчислюються значення двійкових виразів ; 3) відповідні регістри переключаються в нові стани, обумовлені значеннями ; 4) KC  обчислюють нові значення інформаційних сигналів X, щовідповідають новому стану пам'яті S,тобто новим значенням слів s1, …, sN*.* Тривалість такту Tвизначається структурою схем і швидкодією логічних і запам'ятовуючих елементів, що використовуються у схемах керуючого й операційного автоматів. Таким чином, такт - це проміжок часу між сусідніми моментами переключення стану пам'яті. Протягом такту пам'ять автомата переключається в новий стан, відмінний від попереднього.

ОА можна розглядати як самостійний об'єкт. У такому випадку термін «такт» умовно застосовується тільки до ОА. *Такт ОА* ОА - це проміжок часу, необхідний для виконання заданого набору МО і обчислення значень ЛУ і дорівнює інтервалу часу від моменту надходження на вхід автомата керуючих сигналів до моменту вироблення значень інформаційних сигналів, що відповідають стану пам'яті автомата.

Розподіл дій у часі виробляється на основі синхронного принципу. При цьому тривалість такту Tвизначається максимальним значенням, необхідним для виконання будь-якої МО й обчислення значення будь-якої ЛУ. Хід часу відзначається за допомогою синхронізуючих сигналів, що виробляються в ЕОМ генератором синхронізуючих імпульсів, що слідують з періодом, який дорівнює тривалості такту Т*.* Ці сигнали використовуються для синхронізації моментів переключення тригерів.

**Характеристики ОА.** Основні характеристики ОА - продуктивність, швидкодія і витрати устаткування (вартість).

*Продуктивність ОА* визначається кількістю МО, що виконуються автоматом за один такт. В одному такті можуть виконуватися тільки сумісні МО, що фігурують в одному операторі мікропрограми, і їхнє число може змінюватися від такту до такту. Отже, кількість МО, що виконуються за один такт, можна розглядати як дискретну випадкову величину. Фактор випадковості вноситься вхідними даними D,у залежності від значень яких процес виконання мікропрограми може розвиватися по різних шляхах, що обумовлені графом мікропрограми. З урахуванням цього продуктивність оцінюють або максимальним значенням, що дорівнює максимальному числу МО, які може виконати автомат протягом такту, або середнім значенням. *Середня оцінка продуктивності* визначається наступним чином. Мікропрограма складається з Коператорів, кожен з яких містить mkМО k=1, …, K,що виконуються спільно за такт часу. При одній реалізації мікропрограми кожен оператор виконується в середньому qkраз. Виходячи з цих значень, середнє число МО, що виконуються автоматом у такті, оцінюється величиною

(7.2)

Середня продуктивність Vзалежить від властивостей мікропрограми (числа сумісних МО в операторах і частоти використання операторів) і структури автомата, що може накладати обмеження на сумісність МО.

*Швидкодія ОА* характеризується тривалістю такту ОА автомата. Чим менше тривалість такту, тим вище швидкодія автомату. Швидкодія залежить, в основному, від внутрішньої структури KC Ф і  і швидкісних характеристик логічних і запам'ятовуючих елементів, що використовуються у KC і пам'яті автомата.

*Витрати устаткування* в OA визначаються сумою витрат устаткування в пам'яті автомата S*,* KC Ф, що реалізують МО, і KC , що обчислюють значення ЛУ.

Важливими характеристиками структур є регулярність і універсальність (багатофункціональність).

*Регулярною* називається структура, що складається з однотипних частин, однаковим образом пов’язаних між собою. Регулярність структури може виявлятися у використанні однакових операційних елементів чи у використанні однакових схем для обробки значень у кожному розряді слова. Структура є максимально регулярною, якщо всі слова обробляються однаковим чином (одним набором МО) і одночасно з цим однаково обробляються всі розряди слова. Чим більш регулярна структура, тим простіше процес її виробництва. Тому регуляризація структури в кінцевому рахунку приводить до збільшення надійності і зменшення вартості виробу.

*Універсальність (багатофункціональність)* структури виявляється в можливості реалізації однією і тією ж структурою досить широкого класу функцій. Ступінь універсальності структури ОА можна охарактеризувати граничною потужністю множини алгоритмів {A1, …, Aw},кожен з яких може бути реалізований структурою. Якщо структура ОА універсальна, то реалізація будь-якого алгоритму зводиться до перенастроювання структури зовнішніми засобами, наприклад, шляхом програмування. Коли система МО і ЛУ повна, максимальний ступінь універсальності досягається, якщо будь-яка MO і будь-яка ЛУ можуть бути поширені на кожен з регістрів автомата. У цьому випадку потужність класу алгоритмів, що ефективно реалізуються структурою, обмежується в основному кількістю слів (регістрів), з якими оперує автомат. Визначення ступеню універсальності абсолютним значенням - значенням потужності Wмножини алгоритмів - викликає серйозні складності. Як правило, ступінь універсальності характеризують не перерахуванням можливих алгоритмів, а описом властивостей (параметрів) класу алгоритмів, що реалізуються структурою. Хоча визначення абсолютного значення універсальності викликає складності, порівняльна оцінка універсальності двох структур знаходиться досить легко, тобто завжди можна сказати, яка з двох структур є більш універсальною. Чим більш універсальна структура, тим ширше область її застосування. Збільшення ступеню універсальності структур дозволяє скоротити номенклатуру виробів, що випускаються, і підвищити обсяг випуску кожного виробу, що приводить до зниження вартості виробництва. Універсальність структур особливо важлива для великих інтегральних схем, оскільки їхня розробка вимагає значних витрат, що окупаються тільки при великому обсязі виробництва. Отже, збільшення універсальності структури приводить до зниження її вартості навіть у тих випадках, коли універсальність досягається за рахунок введення в структуру додаткового устаткування.

Виявляється, що регулярність і універсальність структур - взаємозалежні властивості. Регулярні структури**,** як правило, більш універсальні, чим нерегулярні, і збільшення ступеня універсальності може бути досягнуте за рахунок регуляризації структури.

**Властивості канонічних структур OA**. Оцінимо продуктивність, швидкодію і витрати устаткування в автоматах з канонічною структурою. Канонічна структура має максимальну продуктивність у порівнянні з іншими варіантами структур, що реалізують той самий алгоритм А. Це пояснюється тим, що канонічна структура не вносить обмежень на сумісність МО: усі функціонально сумісні МО можуть виконуватися паралельно в одному такті. Тому витрати часу на виконання алгоритму з використанням канонічної структури мінімальні в порівнянні з іншими варіантами структур OA. Менших витрат часу можна досягти, якщо змінити алгоритм виконання операцій у пристрої. Швидкодія різних структур OA, що побудовані на одній і тій же елементній базі, розрізняється незначно. Але все-таки канонічна структура має найвищу швидкодію (їй властива найменша тривалість такту ОА) у порівнянні з іншими варіантами структур.

У більшості випадків канонічна структура не є мінімальною по кількості використовуваного устаткування, що викликано наступними причинами: 1) пам'ять автомату може бути надлишковою по відношенню до розглянутого алгоритму; 2) множина схем Z={1, …, M, 1, …, L} може містити схеми, еквівалентні по функціям, що ними реалізуються; 3) множині схем Z може відповідати інша множина Z’, еквівалентна за своїми функціями, але така, що породжує менші витрати устаткування. Зазначені причини усуваються в такий спосіб.

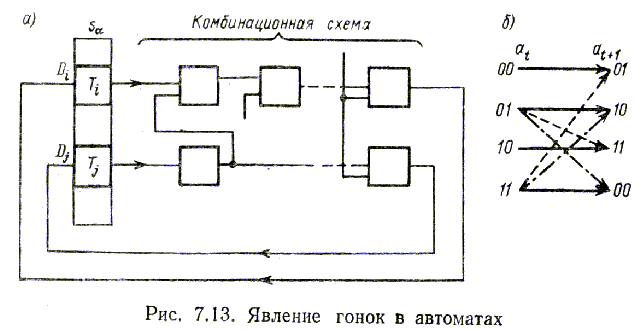
Для мінімізації пам'яті автомату розроблений ряд формальних методів, що дозволяють перетворити алгоритм таким чином, щоб мінімізувати кількість слів, що використовуються в алгоритмі для представлення даних.

Надмірність KC Z={1, …, M, 1, …, L} викликана наступним. У схемі (рис. 7.11) для виконання МО y3, y11 використовуються два суматори, еквівалентні з точки зору функцій, що реалізовуються ними. Витрати устаткування зменшаться, якщо для виконання цих МО використовувати один суматор. Більш того, цей же суматор можна використовувати і для виконання МО рахунку y7: С:=С+1. Таким чином, декілька KC a, …, , що реалізують однакові функції, можна замінити однією схемою, що дозволяє зменшити кількість устаткування в автоматі. Нарешті, шляхом глибоких перетворень алгоритму, що приводять до зміни набору МО і ЛУ, можна створити на­бір, для реалізації якого будуть потрібні KC Z’, що приводять до менших витрат устаткування в порівнянні зі схемами Z={1, …, M, 1, …, L} [23]. Техніка глибоких перетворень алгоритмів тільки починає розроблятися, тому такі перетворення з метою зміни часу реалізації алгоритмів і зменшення витрат устаткування виробляються в основному евристичними методами.

Таким чином, канонічна структура, що реалізує задану функціональну мікропрограму (алгоритм), має максимально можливу для даної мікропрограми продуктивність і максимальну швидкодію. Канонічна структура синтезується прямо по функціональній мікропрограмі без використання яких би то не було процедур мінімізації витрат устаткування. Тому в загальному випадку канонічна структура є надлишковою по кількості устаткування, що використовується в ній.

**Тема: ЗАБЕЗПЕЧЕННЯ СТІЙКОСТІ ФУНКЦІОНУВАННЯ OA**

У функціональних мікропрограмах зустрічаються МО, що присвоюють слову sзначення, що залежить від значення того ж слова s. Зокрема, такими є МО . При реалізації подібних МО у схемах можуть виникати негативні явища: гонки сигналів і проскакування станів.



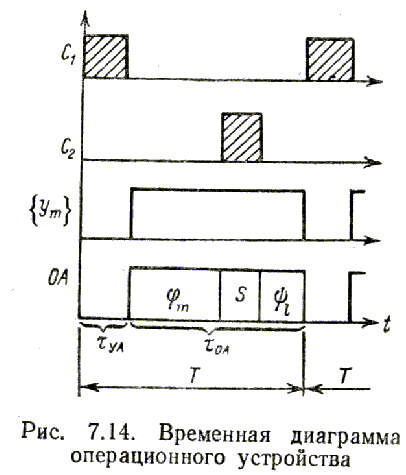
*Явище гонок* полягає в наступному. У момент надходження керуючого сигналу, що ініціює MO, у схемі починається процес передачі сигналів з виходів регістрів через KC до входів тригерів регістра, якому присвоюється нове значення, що обчислюється KC. Під впливом вхідних сигналів тригери регістра переключаються в нові стани. На рис. 7.13, *а* зображені шляхи проходження сигналів від тригерів Ti, Tjрегістра s через KC до входів цих тригерів, причому вихідні сигнали з тригерів Ti, Tjчерез логічні елементи можуть впливати на значення сигналів збудження Di, Dj,тобто в загальному випадку Di =fi(Ti, Tj, …),Dj= fj(Ti, Tj, …). Сигнали поширюються по різних ланцюгах і одні можуть досягати входів тригерів швидше, ніж інші. Це викликано, по-перше, тим, що різні елементи мають неоднакову затримку, обумовлену розкидом електричних параметрів елементів, і, по-друге, різним числом елементів, через які проходять сигнали. Якщо якийсь сигнал першим досягне входу тригера (виграє гонки), то відповідний тригер першим змінить свій стан і вихідний сигнал. Нове значення вихідного сигналу вплине на значення інших сигналів збудження, що формуються KC. У результаті цей сигнал збудження, що повинен надійти на якийсь тригер, може змінити своє значення, і тригер, що не встиг переключитися в черговий стан, буде встановлений у стан, відмінний від необхідного. Ефект гонок сигналів ілюструється рис. 7.13, *б.* Тут розглядається 2-розрядний регістр, що може знаходитися в станах 00, 01, 10, 11, обумовлених станами тригерів Ti, Tj*.* Припустимо, що на регістрі виконується MO рахунку, що збільшує код *at* стану регістра на 1. Стан регістра після виконання МО рахунку позначимо *at+*1. При нормальному функціонуванні схеми регістр повинен переключатися зі стану *at* у стан *at+*1 в порядку, визначеному на рис. 7.13, *б* суцільними лініями, тобто при *at =* 00 наступний стан - = 01, при і так далі. Однак, через гонки сигналів або тригер , або тригер може переключитися першим. Переходи *at**at+*1, при яких тригер переключається першим, позначені на рис. 7.13, *б* штриховими лініями, а переходи, при яких першим переключається тригер - штрих-пунктирними лініями. Як видно з рис. 7.13, *б,* при *at* =01 через гонки регістр може переключитися в стан *at+*1*,* що дорівнює 11 чи 00 і відмінний від необхідного.

Таким чином, через розходження часових характеристик ланцюгів, по яких поширюються сигнали, може припинитися вироблення одних сигналів збудження під впливом інших, що мають менший час поширення, у результаті чого деякі запам'ятовуючі елементи не встигнуть переключитися в необхідні стани. Таке явище називається гонками сигналів. Результат гонок - перекручування функцій, виконання яких покладено на схему.

До такого ж результату приводить явище *проскакування станів.* Проскакування станів виникає в тому випадку, якщо автомат повинен переключатися в стан *at+*1*,* відмінний від початкового *at.* Якщо автомат знаходиться в стані *at* і надходить сигнал, що ініціює MO, тобто перехід у черговий стан *at+*1,у схемі може розвитися послідовність переходів *at**at+*1…*at+*k,що закінчиться лише в момент скидання в 0 керуючого сигналу. Стан *at+k,* у який перейде автомат, залежить від швидкісних характеристик елементів і тривалості керуючого сигналу, що суперечить логіці функціонування схеми. Явище, у результаті якого автомат протягом такту багаторазово змінює свій стан, називається *проскакуванням станів*.

Явища гонок і проскакування станів можуть виникати одночасно і як завгодно спотворювати необхідний порядок функціонування автомата, як говорять, порушувати *стійкість* функціонування автомата.

Для забезпечення стійкості функціонування автомату можна використовувати синхронізацію. При цьому регістри будуються на синхронних тригерах, на відповідні входи яких від генератора синхронізуючих імпульсів одночасно подається сигнал, що дозволяє переключення тригерів. До моменту надходження синхронізуючого сигналу вироблення всіх сигналів збудження повинне бути закінчене. Тому момент надходження синхронізуючого сигналу повинен відстояти від моменту надходження керуючих сигналів на проміжок часу, що перевищує максимальну тривалість перехідних процесів у KC.

Однак синхронізація тригерів не є радикальним засобом забезпечення стійкості через те, що синхронізуючий сигнал має не нульову тривалість. Поки синхронізуючий сигнал дорівнює 1, можуть змінитися, причому не одночасно, значення сигналів на виходах тригерів, слідом за якими змінюється значення сигналів збудження. Коливання температури, старіння елементів істотно впливають на затримки сигналів в елементах і час переключення тригерів. Тому при наявності синхронізації можливі гонки і проскакування станів, хоча і з меншою імовірністю.

Ефект гонок і проскакування станів цілком виключається, якщо в регістрах використовувати подвійні синхронні тригери - тригери типу MS. Подвійні тригери реагують на значення сигналів, що існували до моменту початку переключення, і зміна вхідних сигналів у період дії синхронізуючого сигналу не позначається на стані, у якому переключається тригер.

Таким чином, у загальному випадку регістри OA повинні будуватися на основі синхронних тригерів типу MS. При цьому робота операційного пристрою в часі організується відповідно до рис. 7.14. Для синхронізації роботи KA служать сигнали С1*,* щослідують з періодом Т,а для синхронізації ОА - сигнали С2*,* щослідують з таким же періодом і зсунуті по фазі по відношенню до сигналів С1. По сигналу С1 KA формує набір керуючих сигналів {ym}досить великої тривалості. Для цього в межах такту Tвиділяється інтервал часу УА, тривалість якого визначається швидкодією KA. Для виконання МО і обчислення значень ЛУ виділяється інтервал часу ОА, протягом якого в ОА обчислюються значення функцій {m} і по сигналу С2 передаються в тригери пам'яті S, після чого якийсь час витрачається на обчислення значень ЛУ {l}. Після того, як значення ЛУ визначені, тобто сформовані інформаційні сигнали X, починається наступний такт *Т* роботи операційного пристрою.

**Тема: СТРУКТУРНА ОРГАНІЗАЦІЯ І СИНТЕЗ I-АВТОМАТІВ**

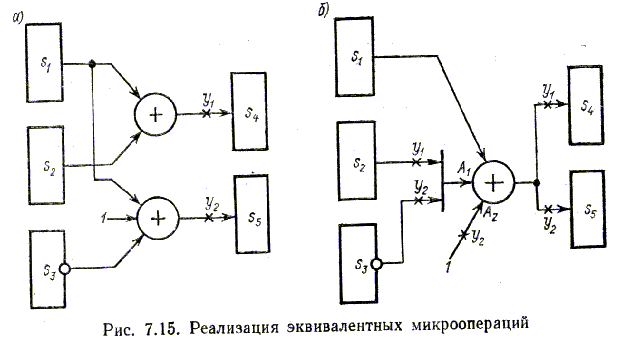
**Еквівалентні мо і узагальнені оператори.**

У мікропрограмах можуть зустрічатися МО виду і , що обчислюють значення слів  з використанням однієї і тієї ж функції m, що застосовується до різних наборів значень і .

Наприклад, МО, де y1, y2 - керуючі сигнали:

(7.3)

реалізують одну і ту ж саму функцію додавання над різними словами.



У канонічній структурі для реалізації цих МО необхідні дві KC - два суматори (рис. 7.15, *а*). Зменшити витрати устаткування в OA можна, якщо використовувати тільки один суматор (рис. 7.15, б)*.* У цьому випадку суматор підключається до регістрів s2, s3 керованими шинами, за допомогою яких вхід A1суматора з'єднується з регістром s2 чи s3. Коли виконується MO y1,то на вхід A1 надходить значення s2, а при виконанні МО y2 - значення s3*.* Аналогічно, константа 1 надходить на вхід A2 суматора тільки при виконанні МО y2. Таким чином, одна KC може використовуватися для виконання декількох МО, еквівалентних по функціям, що реалізуються ними. За рахунок цього можуть бути зменшені витрати устаткування в OA.

Використання однієї KC для виконання декількох МО виключає сумісність цих МО. Так, функціонально сумісні МО (7.3) у структурі (рис. 7.15, а)можуть виконуватися в одному такті, але стають несумісними в структурі (рис. 7.15, б), тобто можуть виконуватися тільки в різних тактах. Якщо у функціональній мікропрограмі МО y1, y2 використовувалися спільно в одній операторній вершині, то при використанні структури (рис. 7.15, б)час виконання операцій збільшиться через структурні обмеження на сумісність цієї пари МО. Таким чином, *економія устаткування може спричинити збільшення часу виконання операцій*.

Процедура мінімізації числа KC, що використовуються в OA для виконання МО, базується на поняттях еквівалентності МО і узагальненого оператора.

МО і вважаються *еквівалентними,* якщо їх оператори містять одну і ту ж саму функцію m, тобто функції в операторах мають однакові імена. При цьому кількість аргументів р і qу функціях m у загальному випадку може бути різною. Еквівалентність МО означає, що для обчислення двійкових виразів, які відповідають MO, може використовуватися та сама KC. Еквівалентність встановлюється в такий спосіб. *Двійкові вирази* і (де С, С - аргументи, що представляються словами, їхніми інверсіями і константами; \* - знаки двійкових операцій) *називаються* *еквівалентними*, якщо один з двійкових виразів може бути приведений до іншого шляхом:

1) заміни слова С словом С чи інверсією С;

2) заміни слова Сконстантою (у тому числі і нулем) і навпаки;

3) заміни одних констант іншими, у тому числі і нульовими;

4) рівносильними перетвореннями виразу .

*Еквівалентними MO* називаються МО з еквівалентними двійковими виразами.

Еквівалентними є МО:

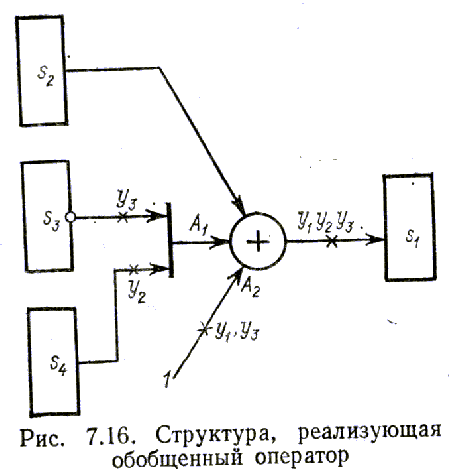
y1: s4 := s2 + 1;

y2: s1 := s2 + s4;;

; (7.4)

Для побудови структури, що реалізує сукупність еквівалентних МО у, …, yw вводиться спеціальна форма представлення таких МО - *узагальнений оператор.*

*Узагальнений оператор* представляється у вигляді , де - допоміжні змінні, що приймають різні значення при виконанні МО у, …, yw:

 (7.5)

Помітимо, що МО у, …, ywнесумісні і тому допоміжна змінна Atвизначається завжди одним із значень, що визначене у правій частині виразу (7.5). Наприклад, еквівалентним MO (7.4) відповідає узагальнений оператор

(7.6)

де

Узагальненому оператору (7.6) відповідає структура (рис. 7.16). Функція, що фігурує в узагальненому операторі, реалізується KC - у даному випадку суматором. Аргументи функції інтерпретуються входами KC. При цьому допоміжна змінна At - ім'я шини, по якій на вхід KC передається значення операнду. Значення допоміжної змінної визначається значеннями слів чи констант, для передачі яких необхідні керовані шини, що приєднуються до шини в загальному випадку через мультиплексор. При виконанні різних МО використовуються спільні засоби, зокрема, та сама керована шина може передавати слово при виконанні двох і більше МО. Такі керовані шини відзначаються декількома керуючими сигналами. На рис. 7.16 ланцюг, по якому передається константа 1*,* відзначений двома сигналами y1, y3,а шина, що зв'язує вихід суматора з регістром s1 - трьома сигналами y1, y2, y3*.* Передача по такій шині відбувається, якщо один з керуючих сигналів має значення 1. Так, при виконанні МО у3відкриваються шина s3 і шина, що зв'язує вихід суматора з регістром s1.

**Клас I-автоматів.** Визначимо структуру ОА, продуктивність якого не нижче продуктивності автомата з канонічною структурою, а витрати устаткування мінімальні. Необхідна продуктивність може бути забезпечена тільки в тому випадку, якщо структура, що синтезується, не буде вносити обмежень на сумісність МО, тобто буде забезпечувати можливість одночасного виконання усіх функціонально сумісних МО. Ця умова виконується тільки в тому випадку, якщо кожна KC використовується для виконання еквівалентних МО, пов'язаних з обчисленням значень одного слова sn (n=1, …, N), але не різних слів. Зі сказаного випливає, що для мінімізації витрат устаткування необхідно узагальнювати KC для виконання декількох еквівалентних МО, що належать одній підмножині МО Yn={sn:=m(S)} (n=1, …, N),щообчислюють значення слова sn*.*

Такий підхід до мінімізації витрат устаткування приводить до структур виду рис. 7.12, у яких KC Ф1, …, ФN будуються з урахуванням еквівалентності МО, що належать відповідним підмножинам Y1, …, YN*.* OA, структура яких забезпечує можливість одночасного виконання усіх функціонально сумісних МО при використанні мінімально можливого числа KC, виділяються в особливий клас OA - *клас I-автоматів.* Для I-автоматів характерно, що кожен з регістрів sn обслуговується своєю KC Фn, засобами якої реалізуються МО, що обчислюють значення слова sn, n=1, …, N*.* Наслідок цього - максимальна продуктивність, що при наявності *N* KC Ф1, …, ФN може, хоча б у принципі, досягати NМО за такт.

**Синтез I-автоматів.** Синтез зводиться до перетворення заданого набору МО Yу сукупність узагальнених операторів, що використовуються як форма для побудови структурної схеми I-автомата. Процес синтезу будемо ілюструвати на прикладі автомата для множення і ділення чисел, функції якого задані табл. 6.4-6.6.

Структура I-автомата синтезується в такий спосіб.

1. Множина МО Y={y1, …, yM}розбивається на підмножини Y1, …, YN*,* щовідповідають внутрішнім словам (регістрам) s1, …, sN*.* MO , що обчислює значення слова s, приписується підмножині Y.

У прикладі, що розглядається, множина МО Y={y1, …, y17}, наведена в табл. 6.5, розбивається на підмножини

YA={y10};

YB={y4, y15, y16};

YC={y1, y3, y5, y7, y8, y9, y11, y13, y17};

YСЧ={y2, y6, y14};

YПП={y12},

пов'язані з внутрішніми словами А, В, С, СЧ, ПП*.*

1. На підмножинах Yn, n=1,…,N виділяються класи еквівалентних МО .

У прикладі, що розглядається, підмножина YAскладається з однієї МО. Кожна MO еквівалентна сама собі, тому KA,1={y10}.У підмножині YB немає пар еквівалентних МО, тому KB,1={y4}, KB,2={y15}, KB,3={y16}. У підмножині YC МО додавання і рахунку y3, y7, y11 є еквівалентними й утворюють клас

Інші МО утворюють одноелементні класи KС,2={y1}, …, KC,7={y17}.Підмножина YСЧне містить пар еквівалентних МО, тому KСЧ,1={y2}, KСЧ,2={y6}, KСЧ,3={y14}. Для підмножини YППмаємо KПП,1={y12}.

3. Для кожного класу , що містить не менше двох еквівалентних МО, будується узагальнений оператор. Якщо клас містить тільки одну MO, то узагальненим оператором для класу є сама MO.

У прикладі, що розглядається, існує один клас KС,1, що містить декілька МО. Йому відповідає наступний узагальнений оператор:

С:=С+А1+А2, (7.7)

де

Тут і надалі при визначенні значень допоміжних змінних А1, А2 рядок «0 в інших випадках» опускається.

4. Виходячи з опису слів, списку узагальнених операторів і ЛУ, будується структурна схема I-автомата. При цьому компоненти функції автомата (слова, узагальнені оператори і ЛУ) інтерпретуються компонентами структурної схеми так само, як при синтезі канонічної структури.

На основі функцій (табл. 6.4 - 6.6) будується структура I-автомата, яка співпадає з канонічною структурою (рис. 7.11) за винятком частини, що реалізує МО YC*.* Структура цієї частини, побудована з урахуванням oпeратору (7.7), зображена на рис. 7.17.

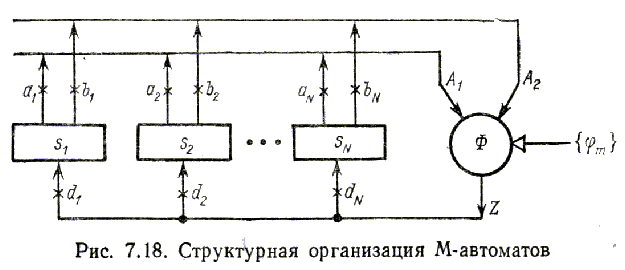
##### 

**Тема: СТРУКТУРНА ОРГАНІЗАЦІЯ І СИНТЕЗ М-АВТОМАТІВ**

У структурі I-автомата можуть міститися еквівалентні за своїми функціями KC, що використовуються для обслуговування різних регістрів. Так, у структурі (рис. 7.11) МО типу додавання y3, y7, y11,пов'язані з регістром С,і MO рахунку y6,пов’язану з регістром СЧ,можна було б реалізувати одним суматором і за рахунок цього зменшити витрати устаткування в автоматі. При заданому алгоритмі виконання операцій витрати устаткування в комбінаційній частині автомата можна мінімізувати, якщо кожну KC m узагальнити стосовно всіх регістрів s1, …, sN,тобто використовувати кожну KC m для виконання всіх еквівалентних МО з множини *Y.* OA, синтезовані на основі принципу узагальнення KC, що використовуються для виконання МО, називаються M-автоматами*.*

**Структура і властивості М-автоматів.** Принцип узагальнення KC породжує структури, подібні наведеній на рис. 7.18. У даній структурі для збереження внутрішніх слів використовуються регістри s1, …, sN (ланцюга передачі даних *D* і виведення результатів *R* на рис. 7.18 не показані). Для обчислення будь-якого двійкового виразу використовується одна KC Ф, рівнодоступна стосовно регістрів s1, …, sN*.* Операнди, що беруть участь у МО, надходять на вхід схеми Ф по шинах A1, A2. Для вибірки слів на шину A1 використовуються керуючі сигнали a1, …, aN, а для вибірки слів на шину A2 - сигнали b1, …, bN*.*

Сигнал аiініціює передачу A1:=si, а сигнал bj - передачу A2:=sj. Схема Ф налаштовується на виконання визначеного перетворення Z:=m(A1, A2) керуючим сигналом m, m=1, …, K*.* Обчислене значення Z може бути занесене в будь-який регістр s1, …, sN*.* Завантаження результату *Z* у регістр *sk* ініціюються керуючим сигналом *dk.* Щоб виконати MO sk:=m(si, sj)*,* необхідно подати набір керуючих сигналів ai, bj, m, dk,під впливом яких на вхід KC вибираються слова si , sj,над ними виконується перетворення m і результат завантажується в регістр sk*.*



Якщо виконуєтьсяунарна MO, наприклад передача sk :=  si, жоден із сигналів bj не виробляється і схема Ф реалізує операцію інвертування. При виконанні МО установки sk :=const в автомат повинні надходити два керуючих сигнали: m, що ініціює формування константи const, і dk,що керує завантаженням значення Z у регістр sk*.* Як видно з рис. 7.18, у кожному такті М-автомат може виконувати тільки одну MO sk:=m(si, sj). Отже, продуктивність М-автомата має мінімальне значення, що дорівнює одній МО за такт. Швидкодія М-автомата відрізняється від швидкодії I-автомата вкрай незначно, оскільки тривалість такту М-автомата збільшується не більше ніж на 2 за рахунок введення в схему мультиплексорів A1, A2. Можна чекати, що витрати устаткування в М-автоматі мінімальні, оскільки кожна KC mФ використовується для виконання всіх еквівалентних МО з множини Y*.*

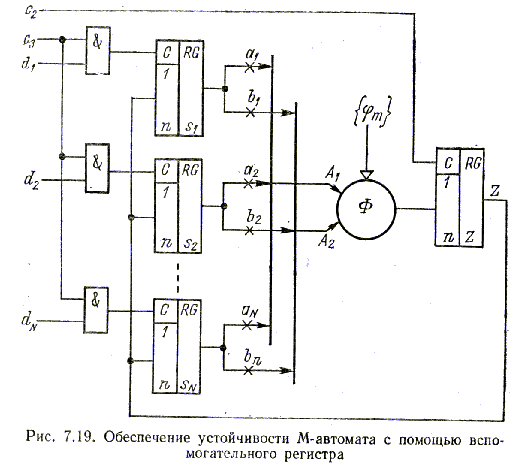
М-автомат породжує специфічний набір керуючих сигналів {ai}, {bj}, {m}, {dk},за допомогою яких ініціюються дії, що приводять до виконання МО y1, …, yM.Кожен із зазначених керуючих сигналів ініціює визначену дію, яку можна розглядати як MO. Таким чином, М-автомат породжує новий набір МО {A1:=si}{A2:=sj}{Z:=m(A1, A2)}{sk:=Z},достатній для реалізації МО функціональної мікропрограми.

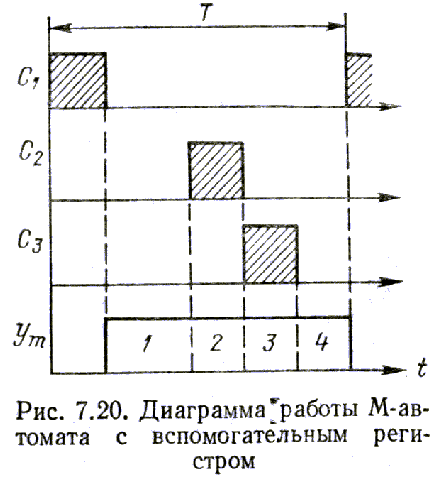
**Забезпечення стійкості М-автоматів.** Стійкість функціонування автомата може бути забезпечена за рахунок використання в регістрах s1, …, sN подвійних тригерів. Тому кількість устаткування в кожному розряді регістра принаймні подвоюється.

Більш економічним є наступне рішення. У вихідну шину KC встановлюється допоміжний регістр *Z* (рис. 7.19). Процес переключення основних регістрів s1, …, sNі допоміжного регістру Z синхронізується сигналами, часова діаграма яких приведена на рис. 7.20. Синхронізуючий сигнал С1 впливає на KA, на виході якого в кожному такті формується набір керуючих сигналів ai, bj, m, dk*.* У період дії керуючого сигналу в OA виконуються наступні дії: 1) вибірка операндів si, sj і обчислення значення двійкового виразу m(si, sj); 2) завантаження обчисленого значення в регістр Z, що синхронізується сигналом С2; 3) передача слова Z у регістр sk*;* 4) обчислення значень ЛУ X:=(S).

На цьому такт роботи операційного пристрою закінчується.

Усталена робота автомата забезпечується парою синхронізуючих сигналів С2 С3, що приймають значення 1 тільки в різні моменти часу. Коли С2 = 1, значення m(si, sj)фіксується на допоміжному регістрі Z при незмінних значеннях слів si, sj, оскільки переключення регістрів s1, …, sN заборонене сигналом С3=0. Сигнал С3 приймає значення 1 тільки після того, як сигнал С2=0. Тому стан регістра skзмінюється при фіксованих значеннях сигналів збудження, що надходять з виходів регістра Z, переключення якого заборонено сигналом С2=0. Таким чином, ланцюг оберненого зв'язку від виходів до входів регістрів S розірваний в будь-який момент часу або сигналом С1*=*0, або сигналом С2=0, що виключає можливість виникнення гонок і проскакування станів, тобто забезпечує стійкість функціонування ОА.



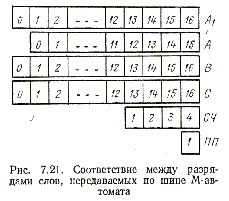
Введення в схему автомата допоміжного регістра дозволяє використовувати в регістрах пам'яті замість подвійних тригерів одинарні тригери, за рахунок чого зменшують витрати устаткування, що приходяться на пам'ять автомата. Однак економія устаткування приводить до збільшення тривалості такту *Т,* тобто зменшенню швидкодії пристрою. За рахунок регістра Z тривалість такту збільшується на величину, що дорівнює часу переключення регістра Z.

**Способи обчислення значень ЛУ.** ЛУ можуть обчислюватися у такий же спосіб, як у канонічній структурі (рис. 7.10): KC 1, …, L підключаються по входах до регістрів s1, …, sN. У цьому випадку схеми 1, …, L забезпечують обчислення значень усіх ЛУ X={x1, …, xL},що використовуються у функціональній мікропрограмі.

Деякі ЛУ є еквівалентними в смислі однаковості властивих їм функцій l і можуть бути реалізовані однієї і тією же KC. Структура М-автомата добре пристосована для узагальнення KC з метою обчислення значень декількох ЛУ, оскільки існує регістр Z, на який можна передати будь-яке слово s1, …, sN чи функцію цих слів. Підключення до регістра Z схем ={1}, щоформують значення ЛУ, дозволяє використовувати одну схему 1 і, отже, один інформаційний сигнал для обчислення і представлення значень декількох еквівалентних ЛУ, пов'язаних з різними словами. За рахунок цього зменшуються витрати устаткування в OA і кількість інформаційних сигналів. Однак при такому способі формування інформаційних сигналів у кожному такті обчислюються значення тільки тих умов, що залежать від результату поточної МО, а інші умови можуть бути визначені тільки в наступних тактах шляхом передачі на регістр Z відповідних слів s1, …, sN*.* З цієї причини, щоб виконати перехід по декількох ЛУ, що пов'язані з різними словами, у мікропрограму включають додаткові оператори, що забезпечують передачу відповідних слів на регістр Z. Через це збільшується час виконання мікропрограми, тобто зменшується швидкодія операційного пристрою, і зростають витрати устаткування у керуючому автоматі.

**Синтез М-автоматів.** Задача синтезу вирішується виходячи з функції ОА, що задається описом слів і списками МО і ЛУ. Синтез зводиться до породження на основі списку МО Yсукупності операторів, що властиві структурі М-автомата: МО передачі A1:=si, A2:=sj,МО перетворення слів Z:=m(A1, A2), МО завантаження результату sk:=Z*.*

Процедура синтезу розділяється на кілька етапів. Етапи синтезу будемо ілюструвати на прикладі автомату для множення і ділення чисел, функції якого визначені табл. 6.4-6.6.

1. *Розподіл регістрів по шинах.* У структурі на рис. 7.18 для передачі операндів використовуються дві шини A1, A2, дояких регістри s1, …, sN підключаються за допомогою *2N* керованих шин, що ініціюються керуючими сигналами a1, …, aN, b1, …, bN*.* Очікується, що в деяких випадках кожен з регістрів може бути підключений до однієї шини: або до шини A1, або до A2, тобто в загальному випадку мінімальне число керованих шин, що використовуються для передачі слів на вхід KC Ф, лежить у межах від *N* до *2N* включно. Коли використовується *2N* шин, автомат виявляється найбільш універсальним, а структура його - у максимальному ступені регулярною. Для економії устаткування регістри прагнуть підключати до шин A1, A2 за допомогою мінімального числа керованих шин. Як правило, у результаті цього зменшується ступінь універсальності і регулярності структури автомата. Мінімізація числа керованих шин, що використовуються для передачі операндів на входи KC Ф, зводиться до поділу множини слів S={s1, …, sN}на дві підмножини і , що задовольняють наступним умовам: 1) якщо слова є операндами однієї МО (одночасно використовуються у двійковому виразі), то ці слова повинні бути включені в різні підмножини A1, A2, тобто si A1, sjA2 або sjA1, siA2; 2) кожне слово повинне міститися хоча б в одній підмножині, тобто A1A2=S;3) сумарні витрати устаткування в схемах передачі операндів s1, …, sN до входів схем 1, …, K повинні бути мінімальні. Для рішення цієї задачі використовуються комбінаторні методи(, що будуть розглянуті надалі).

Для прикладу, що розглядається, приймемо, що слова А, В, С, СЧ, ППрозділяються на наступні підмножини:

A1={C}, A2={А, В, С, СЧ, ПП} (7.8)

Це означає, що кожне із слів підключається до шини A2, а до шини A1підключається тільки слово С,що, як видно з табл. 6.5, бере участь у якості операнду в MO y3, y4, y11, y16 спільно зі словами А і В*.*

2. *Визначення форматів і значень слів* A1, A2. Як правило, приймається наступна угода про порядок представлення внутрішніх слів s1, …, sN допоміжними словами A1, A2: операнди , щоскладають множину Ai, вирівнюються по молодших розрядах, тобто молодші розряди слів співставляються з молодшим розрядом слова Ai*.* Кількість розрядів у слові Ai визначається максимальним числом розрядів у слові , що включене у множину Ai.

У прикладі, що розглядається, відповідно до (7.8), слово A1співставляється зі словом С і тому повинне мати формат А1(0:16). Відповідність між розрядами допоміжного слова A2 й основних слів А, В, С, СЧ, ПП,що описані у табл. 6.4, представлена на рис. 7.21.

3. *Визначення операторів, що реалізуються* *М-автоматом.* МО вибірки слів на шини, перетворення слів і завантаження результату в регістри будемо називати *операторами* на відміну від МО y1, …, yM, що використовуються у функціональній мікропрограмі.

Визначення операторів зводиться до побудови табл. 7.6, у якій зазначені оператори, що відповідають MO, перерахованим у табл. 6.5.

При цьому використовуються наступні угоди: 1) слова A1, A2представляють тільки прямі значення операндів, але не їхні інверсії: 2) невизначуване перетворення m, що реалізується KC, породжує значення *Z* = 0:3) молодший розряд слова *Z* відповідає молодшим розрядам слів s1, …, sN (у даному випадку слів А, В, С, СЧ, ПП)*.* Поділ МО на сукупність операторів, що відповідають структурі автомата, виконується евристично. При побудові операторів необхідно враховувати відповідність між розрядами внутрішніх слів s1, …, sN і допоміжних слів A1, A2, Z, а також мінімізувати число операторів, за допомогою яких описується процес виконання МО Y*.* З метою уніфікації операторів змінена форма МО : двійкова змінна С(15) замінена змінною С(16), а поле В(1:15) - полем В(1:16). З цих же розумінь MO у17: С: =В(1:15), що забезпечує передачу слова Взі зсувом на один розряд вправо (рис. 7.21), представлена оператором зсуву Z:=R1(0.B(1:16)), еквівалентним МО .

Таблиця 7.6

Оператори, що реалізуються М-автоматом

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ym | МО | Оператори | | | |
| А1:=si | А2:=sj | Z:=m(А1,А2) | sk:=Z |
| y1 | C:=0 |  |  |  | С:=Z |
| y2 | СЧ:=1510 |  |  | Z:=1510 | СЧ:=Z |
| y3 | C:= C+A(1:15) | А1:=С | А2:=A | Z:= А1+А2(1:15) | С:=Z |
| y4 | B(1:15):=R1(C(16).B(1:15)) | А1:=С | А2:=B | Z:=R1(А1(16).А2(1:16)) | B(1:16):=Z |
| y5 | C:= R1(0.C) |  | А2:=C | Z:=R1(0.А2) | С:=Z |
| y6 | СЧ:= СЧ – 1 |  | А2:=СЧ | Z:=А2 - 1 | СЧ:=Z |
| y7 | C:= C + 1 | А1:=С |  | Z:=А1 + 1 | С:=Z |
| y8 | C(0):= 1 |  |  | Z:=100...0 | С(0):=Z(0) |
| y9 | C:= A(1:15) |  | А2:=A | Z:=А1(1:15) | С:=Z |
| y10 | A:= B(1:15) |  | А2:=B | Z:=А1(1:15) | A(1:15):=Z |
| y11 | C := C + 11.A(1:15) + 1 | А1:=С | А2:=A | Z:= А1+11.А(1:15)+1 | С:=Z |
| y12 | ПП:= 1 |  |  | Z:=100...0 | ПП:=Z |
| y13 | C:= L1(C.0) |  | А2:=С | Z:=L1(А.0) | С:=Z |
| y14 | СЧ:= 0 |  |  |  | СЧ:=Z |
| y15 | B:= 0 |  |  |  | B(1:16):=Z |
| y16 | B(1:16):=L1(B(1:16).C (0)) | А1:=С | А2:=B | Z:=L1(А2(1:16).А1(0)) | B(1:16):=Z |
| y17 | C := B (1:15) |  | А2:=B | Z:=R1(0.А2(1:16)) | С:=Z |

1) слова A1, A2представляють лише прямі значення операндів, але не їхні інверсії:

2) невизначуване перетворення m, що реалізується KC, породжує значення *Z* = 0:3) молодший розряд слова *Z* відповідає молодшим розрядам слів s1, …, sN (у даному випадку слів А, В, С, СЧ, ПП)*.*

***Поділ МО на сукупність операторів, що відповідають структурі автомата, виконується евристично***. При побудові операторів ***необхідно враховувати*** відповідність між розрядами внутрішніх слів s1, …, sN і допоміжних слів A1, A2, Z, а також мінімізувати число операторів, за допомогою яких описується процес виконання МО Y*.*

***З метою уніфікації операторів змінена форма МО*** у4: двійкова змінна С(15) замінена змінною С(16), а поле В(1:15) - полем В(1:16). ***З цих же міркувань*** MO у17: С:=В(1:15), що забезпечує передачу слова Взі зсувом на один розряд вправо (рис. 7.21), представлена оператором зсуву Z:=R1(0.B(1:16)), еквівалентним МО у4.

Табл. 7.6 містить у собі список операторів, необхідний для реалізації МО Yв М-автоматі.

У табл. 7.7 перераховані оператори (МО), що виконуються відповідними підсхемами М-автомата.

###### Таблиця 7.7

Керуючі сигнали й оператори (МО) М-автомата

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вибірка операндів | | Перетворення слів | | Завантаження результату | |
| bj | А2:=sj | m | Z:=m(А1,А2) | dk | sk:=Z |
| b1 | А2:=A | 1 | Z:=1510 | d1 | A(1:15):=Z |
| b2 | А2:=B | 2 | Z:=А1+А2(1:15) | d2 | B(1:16):=Z |
| b3 | А2:=C | 3 | Z:=R1(А1(16).А2(1:16)) | d3 | С(0):=Z(0) |
| b4 | А2:=СЧ | 4 | Z:=А1(0.А2 ) | d4 | С:=Z |
|  |  | 5 | Z:=А2 - 1 | d5 | СЧ:=Z |
|  |  | 6 | Z:=А1 + 1 | d6 | ПП:=Z(0) |
|  |  | 7 | Z:=100...0 |  |  |
|  |  | 8 | Z:=А1(1:15) |  |  |
|  |  | 9 | Z:= А1 + 11.А(1:15) + 1 |  |  |
|  |  | 10 | Z:=L1(А.0) |  |  |
|  |  | 11 | Z:=L1(А2(1:16).А1(0)) |  |  |
|  |  | 12 | Z:=R1(0.А2(1:16)) |  |  |

***У таблицю заносяться тільки попарно різні оператори***, яким ставляться у відповідність керуючі сигнали ai, bj, m, dk. ***У прикладі, що розглядається, на шину А1 вибирається тільки слово С, тому відсутня необхідність в операторі*** a1: А1:=C*.*

4.***Кодування МО наборами керуючих сигналів****.*

MO уmYфункціональної мікропрограми виконується в М-автоматі під впливом набору керуючих сигналів ai, bj, m, dk. Набори сигналів визначаються на основі табл. 7.6, 7.7 і представляються у вигляді табл. 7.8, що використовується для побудови закодованого графа мікропрограми, необхідного для синтезу KA.

Таблиця 7.8

Кодування МО наборами керуючих сигналів

|  |  |  |  |
| --- | --- | --- | --- |
| МО | Набір керуючих сигналів | | |
| bj | m | dk |
| y1 | - | - | d4 |
| y2 | - | 1 | d5 |
| y3 | b1 | 2 | d4 |
| y4 | b2 | 3 | d2 |
| y5 | b3 | 4 | d4 |
| y6 | b4 | 5 | d5 |
| y7 | - | 6 | d4 |
| y8 | - | 7 | d3 |
| y9 | b1 | 8 | d4 |
| y10 | b2 | 8 | d1 |
| y11 | b1 |  | d4 |
| y12 | - |  | d6 |
| y13 | b3 |  | d4 |
| y14 | - | - | d5 |
| y15 | - | - | d2 |
| y16 | b2 | 11 | d2 |
| y17 | b2 | 12 | d4 |

5. ***Визначення класів еквівалентних МО****.* Для мінімізації витрат устаткування в комбінаційній схемі Ф, що реалізує оператори {m}, множина МО (операторів) {m} розбивається на класи еквівалентних МО K1, …, Kj.

У прикладі, що розглядається**,** МО 1, …, 12 утворюють наступні класи еквівалентності:

K1={1}={Z:=1510};

;

;

;

;

. Класи K1, K4містять по одній МО установки. У клас K2 об'єднані МО додавання і рахунку, що реалізуються шляхом додавання операндів. MO рахунку Z:=A2-1 також реалізується додаванням значення A2 з додатковим кодом одиниці 11...1. У клас K3включені МО зсуву вправо, що розрізняються в основному тільки значеннями, що вводяться у вивільнюваний розряд слова, що зрушується. Клас K5 містить MO передачі. У клас K6 включені МО зсуву слова A2 вліво на один розряд.

6. ***Побудова узагальнених операторів****.*

Для кожного класу еквівалентних МО K1, …, Kj будується узагальнений оператор. Допоміжні змінні будемо позначати ідентифікаторами A3, A4, …

***Класам K1, …, K6 відповідають наступні*** ***узагальнені оператори***:

1) ;

2) ;

;

;

;

3) ;

4) Z:=100…0;

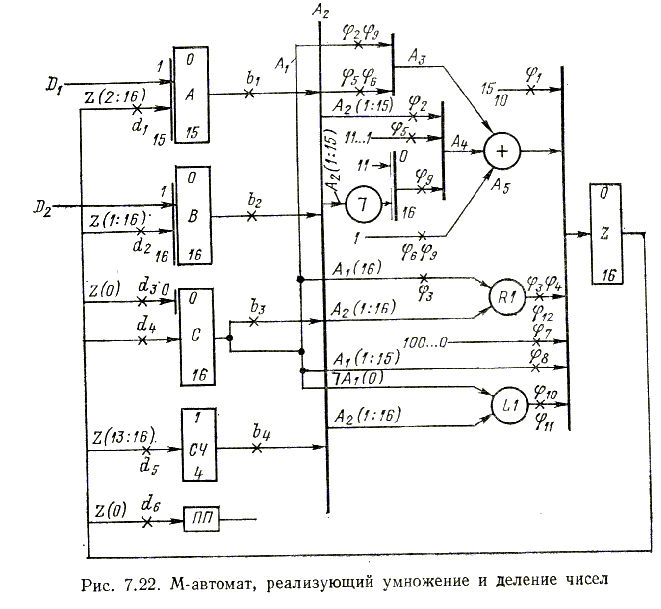
5) Z:=A1(1:15);

6) .

При побудові узагальнених операторів 3 і 6 слово A2, що бере участь у MO зсуву 4 і 10, замінено полем A2(1:16), оскільки при цьому зміст МО 4, 10 і, відповідно, МО y5, y13 (див. табл. 7.6) не змінюється, а узагальнені оператори спрощуються.

7. ***Побудова структурної схеми автомата****.*

На основі опису слів, операторів і списку ЛУ будується структурна схема М-автомату. При цьому слова, оператори і ЛУ інтерпретуються елементами структурного базису (регістрами, шинами і KC) так само, як і при синтезі канонічної структури. Структура М-автомата, що призначена для множення і ділення чисел, синтезується на основі опису слів, представленого в табл. 6.4, списку операторів (табл. 7.7), що забезпечують вибірку операндів і завантаження результатів, вищенаведених узагальнених операторів і списку ЛУ (табл. 6.6). Структурна схема М-автомата приведена на рис. 7.22.



У даному прикладі вихідна шина регістра Сутворює шину A1М-автомата. Виходи регістрів А, В, С, СЧчерез мультиплексор приєднані до шини A2*.* Будь-який вхід суматора + є керованою шиною, у зв'язку з чим не нульове значення на виході суматора можливо тільки в момент виконання МО 2, 5, 6, 9. Тому вихід суматора з'єднується з входом допоміжного регістра Z некерованою шиною. Оскільки до шини А1підключений тільки регістр С, значення A1(0), що використовується при виконанні МО зсуву L1, знімається безпосередньо з виходу регістра Сбез використання інвертора.



**Розширники комбінаційної частини автомата.** З наведеного прикладу видно, що довжину регістрів, що використовуються для збереження операндів, проміжних значень і результатів, необхідно збільшувати на один розряд, що необхідний для збереження двійкового значення, яке отримують в результаті проміжних дій при зсуві чи додаванні для наступного округлення. Через це збільшується число МО, за допомогою яких реалізується алгоритм у М-автоматі, ускладнюється структура автомата і зменшується ступінь її регулярності. Щоб виключити ці негативні явища, використовується наступний прийом.

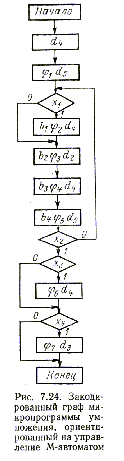
Комбінаційна частина автомата Ф забезпечується однорозрядними регістрами s0, sp (p=N+1), що підключаються за схемою (рис. 7.23). Коли при виконанні МО над n-розрядними операндами виходить (n *+* 1)-розрядний результат, то додатковий старший розряд результату заносять у регістр s0 і, аналогічно, додатковий молодший розряд - у регістр sp*.* Значення s0 i sp можуть використовуватися в якості операндів у наступних MO, а також як ЛУ xL+1, xL+2*.* Розширники s0, spвикористовуються в такий спосіб. Припустимо, що необхідно виконати зсув слів s1 і s2 на один розряд вправо з передачею молодшого розряду слова s1 в старший розряд слов s2. Ці дії реалізуються двома послідовно виконуваними MO: s1.sp:=0.s1, s2 :=R1(sp.s2),де s1.sp - складене (n+1)-розрядне слово. У результаті виконання першої МО змінна spприйме значення n-го розряду слова s1, а слово s1 буде складене з нуля і значень розрядів 1, …, n. Перенос, що з'являється при додаванні n-розрядних слів, може бути зафіксований у регістрі s0 за допомогою МО s0.s1:= s1+s2. Надалі значення s0 може бути введене в слово s1шляхом зсуву вправо на один розряд, причому спадаючий при зсуві молодший розряд можна зберегти в регістрі sp*.*

У переважній більшості випадків застосування розширників благотворно впливає на властивості М-автомату. Тому при використанні М-автоматів список МО, отриманий з функціональної мікропрограми, модифікується з урахуванням того, що додаткові розряди слів представляються значеннями s0 i sp*.* Так МО (табл. 6.5), що використовуються для множення і ділення чисел, перетворяться до виду, представленому в табл. 7.9. Нагадаємо, що МО y1,…, y8 використовуються при множенні, а МО y3, …, y17 - при діленні.

Таблиця 7.9

МО, що орієнтовані на використання розширників s0, sp

|  |  |  |  |
| --- | --- | --- | --- |
| ym | MO | ym | MO |
| y1 | C:=0 | y10 | A(1:15) :=B(1:15) |
| y2 | СЧ:=1510 | y11 | s0.C := s0.C + 11.A(1:15) + 1 |
| y3 | s0.C:=s0.C+A(1:15) | y12 | ПП:= 1 |
| y4 | B(1:15):=R1(0.B(1:15)) | y13 | s0.C := C.0 |
| y5 | C.sp := 0.C | y14 | СЧ:= 0 |
| y6 | СЧ:= СЧ – 1 | y15 | B(1:15):= 0 |
| y7 | C:= C + 1 | y16 | B(1:16).sp:=B(2:16).sp.s |
| y8 | C(0):= 1 | y17 | C := B(1:15) |
| y9 | C:= A(1:15) |  |  |

При наявності розширників слова А, В, С, щопредставляють значення операндів, проміжних і кінцевих результатів, зображуються в єдиному 16-розрядному форматі. Додатковий розряд у слові С вводиться за рахунок розширника s0, значення якому надаються в MOх y3, y11, y13*.* Остання є мікрооперацією зсуву вліво на один розряд. При зсуві двійкова змінна s0 приймає значення старшого розряду слова С*.* У МО y5(зсуві слова Свправо на один розряд), що використовується при множенні, для збереження 16-го розряду добутку використовується двійкова змінна sp, за значенням якої виробляється округлення добутку. Аналогічно, для збереження 16-го розряду частки, що формується мікрооперацією зсуву y16на один розряд уліво, використовується змінна s0, якій приписується значення інверсії знака залишку s0. Табл. 7.9 використовується в якості вихідних даних для визначення набору операторів, що реалізуються М-автоматом.

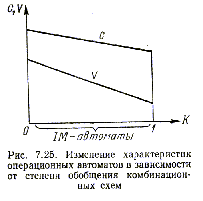
Коли в мікропрограмі здійснюються зсув на *т* розрядів і передача m-розрядних значень між словами, що зрушуються, як розширники s0, spвикористовуються m-розрядні регістри.

**Перетворення закодованого графа мікропрограми, що пов'язаний з М-автоматом.** М-автомат у кожному такті реалізує тільки одну MO, що ініціюється не одним керуючим сигналом ym, анабором сигналів ai, bj, m, dk.З цієї причини закодований граф мікропрограми, що визначає функцію керуючого автомату, повинен бути перетворений у такий спосіб. По-перше, кожна операторна вершина, що містить *q* функціонально сумісних МО *,* повинна бути замінена послідовністю з *q* операторних вершин, що містять МО . По-друге, кожен символ ymYповинен бути замінений відповідним набором керуючих сигналів ai, bj, m, dk*.*

Ha рис. 7.24 приведений граф мікропрограми множення, еквівалентний графу на рис. 6.8, а. Набори керуючих сигналів, що ініціюють М-автомат, вибираються з табл. 7.8.

**Тема: СТРУКТУРНА ОРГАНІЗАЦІЯ І СИНТЕЗ IM-АВТОМАТІВ**. **КЛАС S-АВТОМАТІВ**

**Клас IM-автоматів.** Структурна організація I-автоматів базується на принципі закріплення KC, що використовуються для виконання МО, за кожним з регістрів s1,…, sN. За рахунок цього усі функціонально сумісні МО можуть виконуватися паралельно в одному такті. Структурна організація М-автоматів базується на узагальненні KC стосовно всіх регістрів, за рахунок чого зменшуються витрати устаткування в автоматі. Ці два класи автоматів мають діаметрально протилежні властивості: I-автоматам властива максимальна продуктивність при найбільших витратах устаткування, а М-автоматам - мінімальна продуктивність при найменших витратах устаткування. Варто очікувати, що між цими двома класами структур OA лежать варіанти структур, що володіють проміжними властивостями: досить високою продуктивністю при помірних витратах устаткування. Ці міркування ілюструються рис. 7.25. Вісь абсцис представляє ступінь узагальнення KC між регістрами. Значення K=0 відповідає I-автоматові, а значення K=1 - М-автоматові, якому властиво максимальне узагальнення схем, що використовуються для виконання МО і, можливо, для обчислення значень ЛУ.

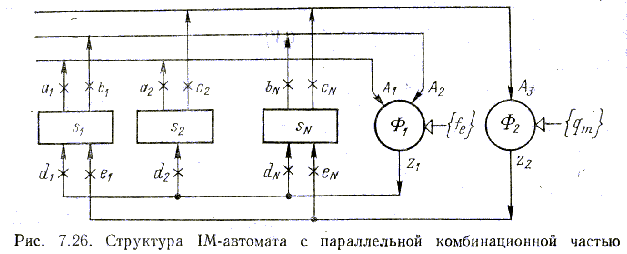
OA, структурна організація яких характеризується ступенем узагальнення KC 0<K<1, виділяються в особливий клас автоматів - клас IM-автоматів*.*

*IM-автоматами* називаються OA, структурна організація яких вносить обмеження на сумісність МО і одночасно з цим забезпечує виконання за такт більше однієї МО функціональної мікропрограми. IМ-автомати базуються на спільності деяких KC, що реалізують МО, і, можливо, обчислюють значення ЛУ, для обслуговування декількох регістрів. Однак ступінь узагальнення схем не настільки висока, як у М-автоматах, і дозволяє в одному такті виконувати більш складні дії, чим або , типові для М-автоматів.

Структури IM-автоматів можуть породжуватися двома способами: використанням для виконання МО паралельних і послідовних KC. Перший спосіб приводить до структур, називаним IM-автоматами *з паралельною комбінаційною частиною,* а другий - до IM-автоматів *з послідовною комбінаційною частиною.*

**IМ-автомати з паралельною комбінаційною частиною.** Варіант такої структури IM-автомату приведений на рис. 7.26. Для збереження слів використовуються регістри s1,…, sN здовжинами, що дорівнюють довжинам відповідних слів. Для обчислення двійкових виразів, які використовуються у MO, служать KC Ф1 і Ф2, що реалізують функції і відповідно. У даному варіанті структури схема Ф1 виконує бінарні операції - додавання, кон’юнкцію, диз'юнкцію, заперечення рівнозначності - над допоміжними змінними A1, A2, значення яких надходять з регістрів si, sj,що виділяються керуючими сигналами ai, bj.Схема Ф2 реалізує унарні операції - передачу, інвертування, зсув, формування констант і так далі - над допоміжною змінною ,значення якої надходить з регістра sk, що виділяється керуючим сигналом ck*.* Результати Z1, Z2 завантажуються в регістри sp, sr, обумовлені керуючими сигналами dp, er. В одному такті автомат може виконувати дві МО , щоініціюються набором керуючих сигналів .Цим сигналам відповідають наступні операції, що реалізуються схемами IM-автомата:

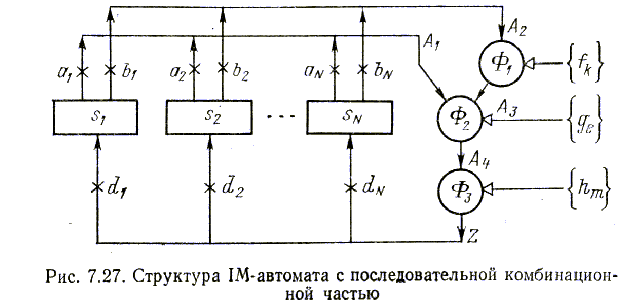
 



Максимальна продуктивність IM-автомата з Впаралельними KC Ф1, …, ФB, 1BNдорівнює *В* MO за такт і збільшується зі збільшенням числа KC. Отже, кількість KC Ф1, …, ФB визначається вимогами до швидкодії операційного пристрою - обмеженням на час виконання операцій.

IM-автомат можна розглядати як композицію з ВМ-автоматів, що мають загальну пам'ять s1, …, sN*.* Виходячи з цього, синтез IM-автомата з паралельною комбінаційною частиною зводиться до розбивки множини МО Y={y1, …, yM}на *В* підмножин Y1, …, YB і синтезу *В* М-автоматів, що реалізують зазначені підмножини МО.

**IM-автомати з послідовною комбінаційною частиною.** Принцип послідовної організації комбінаційної частини ОА приводить до структур, показаним на рис. 7.27. У даному випадку комбінаційна частина складається з трьох схем Ф1, Ф2, Ф3, що реалізують операції з множин {fk}, {gl}, {hm}відповідно. Операції поширюються на пари слів si, sj,що вибираються на входи A1, A2 комбінаційної частини під впливом керуючих сигналів ai, bj, щоініціюють передачі A1:=si, A2:=sj. KC Ф1, Ф2, Ф3 налаштовуються на виконання необхідних МО керуючими сигналами fk, gl, hm,що ініціюють наступні перетворення: A3:=fk(A2); A4:=gl(A1, A2), Z:= hm (A4),де A3, A4, Z - допоміжні змінні. Запис результату Z у пам'ять автомата ініціюється керуючим сигналом dn: sn:=Z*.*



Таким чином, сукупність МО fk, gl, hm над словами si, sj з метою обчислення слова sn ініціюється набором керуючих сигналів (ai, bj, fk, gl, hm, dn),під впливом яких автомат за один такт реалізує наступне перетворення:

sn:= hm(gl(si, fk(sj))), (7.9)

еквівалентне трьом послідовно виконуваним MO fk, gl, hm*.* За рахунок цього максимальна продуктивність IM-автомата зі структурою на рис. 7.27 у три рази перевищує продуктивність М-автомата.

Щоб забезпечити можливість виконання МО fk, gl, hm у будь-якому сполученні, кожна зі схем Ф1, Ф2, Ф3 повинна реалізувати МО передачі слів: A3:=A2; A4:=A1; A4:=A3; Z:=A4. За рахунок цього комбінаційна частина крім перетворення (7.9) може виконувати перетворення наступного виду: Z:=A1; Z:=A2; Z:=fk(A2); Z:=gl(A1, A2); Z:=hm(A1); Z:= gl(A1, fk(A2)) і так далі.

При використанні IM-автоматів в якості OA процесорів KC Ф1, Ф2, Ф3, як правило**,** реалізують наступні функції. Схема Ф1 використовується для формування констант, полів і інверсій слів і називається *формувачем кодів.* Для формувача кодів типовий наступний набір МО:

і т.д. MO f0 забезпечує передачу слова A2 без перетворення. По сигналах f1, f2, f3 на виході формувача утворюються значення необхідних констант. MO f4 служить для передачі цифрових розрядів слова A2 (модуля операнда), MO f5 - для передачі слова з від’ємним знаком. MO f6 формує обернений код слова A2, а MO f7 виділяє перший байт операнда.

Схема Ф2 використовується для виконання бінарних операцій +, , ,  основною з яких є додавання. Тому схему Ф2 часто називають *суматором* і покладають на неї виконання МО наступного виду:

де g0, g1- МО передачі слів із входу на вихід суматора.

KC Ф3 використовується для виконання МО зсуву і називається *зсувачем. Зсувач* реалізує МО наступного виду:

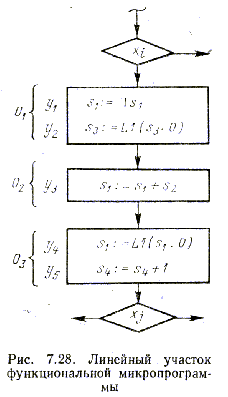
де h0- MO передачі і h3, h4 - МО циклічного зсуву. Різним значенням керуючих сигналів відповідають різні перетворення. Наприклад,

Як і в М-автоматі, у комбінаційній частині IM-автомату можуть використовуватися розширники для збереження переносу зі старшого розряду суматора і розрядів, що спадають при зсуві слова A4.

Синтез IM-автомата з послідовною комбінаційною частиною здійснюється на основі функціональної мікропрограми шляхом представлення послідовностей МО у формі виразу (7.9). Для цього мікропрограма розділяється па лінійні ділянки виду (рис. 7.28), що складаються з послідовності операторів O1, …, Or*.* Лінійні ділянки класифікуються по рангах r=1, …, R*.* До рангу rвідноситься ділянка, що складається з rоператорів O1, …, Or*.* У результаті функція ОА представляється множинами L1, …, LR лінійних ділянок

(7.10)

Потім визначається множина виразів, що породжуються лінійними ділянками L1, …, LR*.* Лінійним ділянкам першого рангу L1відповідають МО .

Вирази, що відносяться до лінійної ділянки рангу r=2, ..., *R,* знаходяться в такий спосіб. Серед МО, що входять до складу операторів лінійної ділянки, виділяються МО ,пов’язані з обчисленням того самого слова sn.Шляхом послідовних підстановок виразів формується MO sn:=m(), що еквівалентна послідовності МО *.* МО виключаються з лінійної ділянки, і процес породження виразів виду (7.9) повторюється для МО лінійної ділянки, що залишилися. Наприклад, лінійна ділянка (рис. 7.28) породжує наступні вирази:

(7.11)

перший з яких еквівалентний послідовності трьох МО y1, y3, y4*.* Якщо використовувати М-автомат, то ділянка мікропрограми (рис. 7.28) буде виконуватися за п'ять тактів. Якщо ОА буде побудований по виразах (7.11), то для виконання цієї ділянки буде потрібно три такти. Множини виразів, що породжуються лінійними ділянками Lr рангу r, визначається шляхом об'єднання виразів, породжуваних окремими лінійними ділянками,

Множина виразів, що породжуються функціональною мікропрограмою,

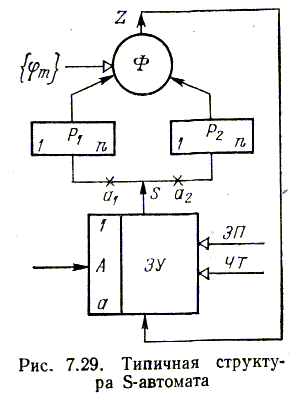
Множина Ммістить усі МО з набору Y,що не покриті виразами, отриманими з лінійних ділянок, і вирази, сформовані шляхом об'єднання МО, що послідовно виконуються на лінійних ділянках мікропрограми.

Множину Мможна розглядати як множину МО, реалізація яких покладається на ОА. Застосуванням процедури синтезу М-автомата до множини МО Мвизначається набір операторів що реалізуються схемами вибірки операндів , комбінаційною частиною і схемою запису результату Z у регістри s1, …, sN*.* По набору операторів будується структурна схема IМ-автомату. Кількість ступенів у комбінаційній частині автомата визначається максимальним числом дій у двійкових виразах М.Так, у (7.11) найбільш складним є перший вираз, що складається з трьох дій: інвертування, додавання і зсув.

**Область застосування IM-автоматів з паралельною і послідовною комбінаційною частиною.** Структура IM-автоматів з паралельною комбінаційною частиною добре пристосована для реалізації мікропрограм, у яких велике число операторів містять трохи сумісних микропераций і лінійні ділянки не містять МО, пов’язаних з обчисленням одного слова. У цьому випадку множина МО *Y* розділяється на два, три чи більш підмножини, що реалізуються схемами Ф1, …, ФKй у кожному такті автомат забезпечує спільне виконання до *K* МО.

IM-автомати з паралельною комбінаційною частиною доцільно застосовувати для реалізації мікропрограм, у яких присутні багаторазово виконувані послідовності МО, що забезпечують обчислення одного слова snS. Такі послідовності найбільше легко виявляються на лінійних ділянках мікропрограми. Однак з метою збільшення продуктивності автомата можна покладати на комбінаційну частину обчислення умовних виразів, значення яких залежить від проміжних результатів.

**Використання запам'ятовуючих пристроїв у пам'яті автомата і клас S-автоматів**

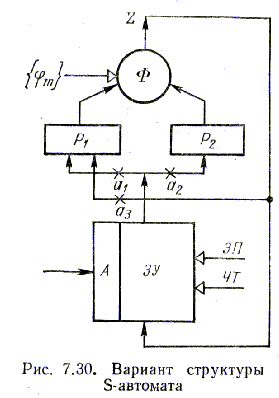
У деяких операційних пристроях операції виконуються над великим числом внутрішніх слів, від десятків до сотень. Прикладами таких пристроїв є мультиплексні канали, процесори з убудованим каналом введення-виведення, спеціалізовані процесори введення-виведення і т.д. Для зменшення вартості таких пристроїв регістрова пам'ять ОА заміняється запам'ятовуючим пристроєм (ЗУ). ЗУ містить схему адресації, що забезпечує в кожен момент часу доступ тільки до однієї комірки ЗУ, тобто до одного слова інформації. Цей фактор накладає відбиток на структурну організацію ОА. ОА, пам'ять якого складається з запам'ятовуючого пристрою, називається *S-автоматом.*

**Структура S-автомата.** Типова структура S-автомата зображена на рис. 7.29. ЗУ забезпечує збереження 2a n-розрядних слів з адресами 0,1, …, 2a-1. Звертання до ЗУ ініціюється сигналами читання ЧТ: S:= ЗУ[А] і запису ЗП: ЗУ[A]:=Z, де A *-* адреса, що ідентифікує слово, яке записується або читається. Адреса генерується KA у момент кожного звертання до ЗУ. Для збереження операндів, що беруть участь у МО, використовуються регістри P1, P2. Задана операція m реалізується KC Ф, що обчислює значення слова Z:=m(P1, P2)*.* Значення Z завантажується в ЗУ по заданій адресі А=А3.Такт Тавтомата розділяється на послідовність дій: 1) читання з ЗУ першого операнда P1:= ЗУ[А1],заданого адресою А1*,* 2) читання з ЗУ другого операнда P1:=ЗУ[А1];3) виконання операції Z:=m(P1, P2), що ініціюється керуючим сигналом m; 4) запис результату ЗУ[A3]:=Z. Таким чином, у кожному такті автомат реалізує MO , де sA- слово, що зберігається в комірці АЗУ.

Як правило, комбінаційна частина S-автомата будується за схемою М-автомата, що забезпечує реалізацію однієї МО за такт. Швидкодія S-автомата визначається в основному швидкодією ЗУ - тривалістю циклу звертання до ЗУ. У загальному випадку швидкодія S-автомата нижче швидкодії автоматів з регістровою пам'яттю, оскільки регістрова пам'ять забезпечує одночасну вибірку всіх операндів і час вибірки слова з регістра менше тривалості циклу ЗУ.

У залежності від способу побудови комбінаційної частини Ф, що виконує набір МО, S-автомат синтезується як I-, М- чи IМ-автомат.

**Способи підвищення швидкодії S-автоматів.** Швидкодія S-автомата може бути підвищена такими способами:

1) завантаженням результату МО Z у регістри P1, P2 з метою його використання в якості операнда в наступній МО;

2) розширенням регістрової пам'яті автомата за рахунок додаткових регістрів P3, P4, …

Найбільш простий варіант структури, що забезпечує використання результату Z у якості операнда в наступній МО, приведений на рис. 7.30. Автомат реалізує МО наступного виду: P1:=m(ЗУ[A1], ЗУ[A2]); P1:=ЗУ[A1]:=m(P1, ЗУ[A2]). У MO першого типу обоє операндів вибираються з ЗУ, і результат операції m завантажується в регістр P1*.* У MO другого типу в якості операнда використовується слово P1і результат операції записується в регістр P1і комірку A1 ЗУ. У цьому випадку в кожному такті виконуються два звертання до ЗУ, у результаті чого зменшується тривалість такту.

Другий спосіб збільшення швидкодії S-автомата зводиться до введення додаткових регістрів P3, P4, …Кожен додатковий регістр зменшує інтенсивність потоку звертань до ЗУ, оскільки збільшується імовірність того, що операнд зберігається в регістрах P1, …, PK ірезультат МО може бути записаний у відповідний регістр. Великий ефект дає введення в автомат регістрів (розширників комбінаційної частини), що використовуються для збереження значень переносів і розрядів, що спадають при зсуві слова.

**Тема:** **КЕРУЮЧІ АВТОМАТИ ІЗ ПРОГРАМОВАНОЮ ЛОГІКОЮ. СТРУКТУРА Й ФУНКЦІОНУВАННЯ КЕРУЮЧИХ АВТОМАТІВ (КА) З М’ЯКОЮ ЛОГІКОЮ**

**Принцип керування по збереженій мікропрограмі**

*Функція керуючого автомату* (КА) визначається: 1) множиною вхідних (інформаційних) сигналів , щовідображають стан операційного автомату (ОА); 2) множиною вихідних (керуючих) сигналів *,* щоініціюють МО, реалізовані ОА; 3) закодованим графом мікропрограми, що задає порядок проходження керуючих сигналів *Y* залежно від значень інформаційних сигналів *X.* Функціонування КА зводиться до генерування послідовності керуючих сигналів *Y*,що визначається мікропрограмою й відповідної послідовності інформаційних сигналів *X.*

**Мікрокоманди.** КА може бути побудований на основі принципу *програмного керування, що* використовує операційно-адресну структуру керуючих слів. Керуюче слово визначає порядок функціонування пристрою протягом одного такту й називається *мікрокомандою*. Сукупність мікрокоманд утворить масив МК[0:P], окремі мікрокоманди в якому виділяються за допомогою адреси, що дорівнює номеру 0, 1, ..., Pелемента масиву МК. Мікрокоманда містить інформацію про МО, які повинні виконуватися в даному такті роботи пристрою, і (або) інформацію про адресу наступної мікрокоманди.

Визначимо найпростішу структуру керуючих слів, що достатня для подання мікрокоманд.

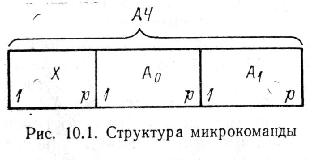
Нехай множина *Y* містить *М* мікрооперацій, яким привласнимо номера 1, 2, ..., M*.* Номера мікрооперацій будемо кодувати m-розрядними двійковими числами, де . Коди мікрооперацій вказуються в операційній частині мікрокоманди. Поле *Y* визначає номер мікрооперацій, що збуджуються мікрокомандою. Якщо *Y* = 0, то поле *Y* є порожнім і не збуджує ні однієї МО.

Для визначення адреси наступної мікрокоманди скористаємося способом примусової адресації. *Примусова адресація* мікрокоманд полягає в тому, що в кожній мікрокоманді вказуються адреси наступних мікрокоманд. Адреса наступної мікрокоманди може задаватися безумовно, тобто незалежно від значень інформаційних сигналів, або вибиратися за умовою, що обумовлена поточними значеннями інформаційних сигналів. Приймемо, що в кожній мікрокоманді можна задавати значення тільки однієї умови з множини *X.*

Тоді адресна частина (АЧ) мікрокоманди буде мати структуру, зображену на рис. 10.1. Поле X визначає номер 1, ..., L інформаційного сигналу x1, ..., xL,значення якого аналізується мікрокомандою. Якщо , то адреса наступної мікрокоманди визначається залежно від значення ,при наступною виконується мікрокоманда з адресою , при - мікрокоманда з адресою *.* Якщо , то адреса наступної мікрокоманди дорівнює . Довжина адрес і залежить від кількості мікрокоманд, що становлять мікропрограму. Якщо мікропрограма містить не більше *р* мікрокоманд, то *.*

**Постійні запам'ятовувальні пристрої (ПЗП).**

Для зберігання мікропрограм використовуються ПЗП, принцип структурної організації яких зображений на рис. 10.2. ПЗП розділяється на дві частини: ЗЧ - *запам'ятовуючу* й АЧ - *адресну.* Запам’ятовуюча частина складається з комірок, яким привласнені адреси . Комірка ПЗП забезпечує зберігання k-*розрядного* слова інформації, що представляє, як правило, одну мікрокоманду. Інформація в комірку заноситься на стадії виготовлення ПЗП і не може бути змінена інакше як шляхом механічного втручання.



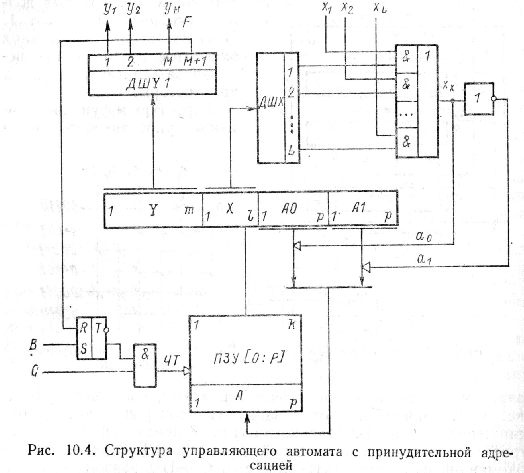
Адресна частина складається з дешифратора адреси, за допомогою якого n-*розрядна* двійкова адреса перетворюється в сигнал, що опитує комірку ЗЧ.

Звертання до ПЗП виконується шляхом передачі адреси А на адресну шину й посилки керуючого сигналу ЧТ, що ініціює процес читання слова інформації. У результаті цього на вихідній шині Сформується слово , що дорівнює константі, яка зберігається в комірці А ПЗП.

Принцип побудови ЗЧ ПЗП ілюструється рис. 10.3. ЗЧ складається з адресних ланцюгів , по яких надходять сигнали з дешифратора адреси ДША, і інформаційних ланцюгів *,* з яких знімається код слова С. Адресні й інформаційні ланцюги утворюють сітку, у вузлах якої розміщаються елементи, що передають сигнали з ланцюга в ланцюг . Якщо j-й розряд слова з адресою *i* повинен містити одиницю, то ланцюг з'єднується з ланцюгом . Код нуля формується при відсутності з'єднання між зазначеними ланцюгами. На рис. 10.3 з'єднання між ланцюгами позначається рисою, що «направляє» сигнал у ланцюг . У частині ПЗП, що зображена на рис. 10.3, сигнали , щоопитують комірки з адресами , формують коди

Елементи, що використовуються в ЗЧ ПЗП, повинні передавати сигнал тільки в одному напрямку, тобто мати ненульову провідність у напрямку й нульову провідність в напрямку *.* У ЗЧ найбільш часто використовуються елементи трансформаторного типу. У цьому випадку занесення інформації зводиться до розведення ланцюгів по тим сердечникам, з яких повинен зчитуватися сигнал 1. В якості запам'ятовуючих елементів в ПЗП можуть використовуватися також діоди й транзистори.

**Структура й функціонування КА.** Нехай мікрокоманди мають структуру, зображену на рис 10.1, причому для кодування мікрооперацій виділяється єдине поле *Y.* У такому випадку КА будується за схемою, що представлена на рис. 10.4. Для зберігання мікрокоманд використовується ПЗП ємністю (Р+1) k-*розрядних* слів. Керуючий сигнал *ЧТ* ініціює операцію читання слова *МК :* = *ПЗП* [А], у результаті виконання якої з комірці *А* ПЗП на регістр *МК* заноситься мікрокоманда. Вибрана із ПЗП мікрокоманда обробляється в такий спосіб. Поле *Y* дешифрується, і вихідний сигнал з дешифратора надходить як керуючий сигнал в операційний автомат, збуджуючи в ньому задану мікрооперацію. При назначеній структурі мікрокоманди в кожному такті може виконуватися не більше однієї МО. Після виконання МО здійснюється перехід до наступної мікрокоманди, адреса якої визначається полем або залежно від значення поля X і логічних умов ,що формуються в операційному автоматі. Якщо , то адреса *А* призначається рівним або залежно від значення ,виділеного полем , якщо ; , якщо . Умовно вважається, що логічна умова ,тотожно дорівнює 0. Тому при адреса . Це правило обчислення адреси наступної мікрокоманди реалізується схемою, що складається з дешифратора *ДШХ,* підсхеми Й-АБО й елемента НЕ, на виході яких формуються керуючі сигнали й *.* Останні ініціюють передачі й відповідно. Таким чином,



до кінця такту на адресній шині *А* буде сформована адреса наступної мікрокоманди, обираної із ПЗП у черговому такті. Перед початком роботи КА регістр мікрокоманди *МК* установлюється в нульовий стан, у результаті чого адреса й першої буде обрана мікрокоманда, збережена в нульовому комірці ПЗП. Запуск автомата виробляється сигналом *В,* що перемикає тригер *Т* у стан 1, у результаті чого синхронізуючий сигнал *з* надходить на керуючий вхід ЧТ, збуджуючи в кожному такті процес читання мікрокоманди із ПЗП. В останньому такті реалізації мікропрограми тригер *Т* повинен бути переключений у стан 0. Для цього в систему мікрооперацій уводиться додаткова мікрооперація ,що інтерпретується у вигляді сигналу *F, що* зупиняє роботу КА.

**Тема:** **КОДУВАННЯ МІКРООПЕРАЦІЙ (МО) ТА ЇХ НАБОРІВ**

МО із множини збуджуються мікрокомандою (МК), у якій вказуюються найменування (коди) МО, що виконуються ОП сумісно протягом такту. Для кодування МО, що можуть виконуватися сумісно, у мікрокоманді виділяються *операційні поля* ,кількість яких визначає граничне число МО, що виконуються сумісно. У загальному випадку поле може збуджувати деяку підмножину МО *,* для кодування яких використовується двійкових розрядів, причому єознакою пустоти поля. Торба визначає *довжину операційної частини* мікрокоманди.

Довжина операційної частини МК впливає на кількість обладнання в КА і шводкодію ОП. Зокрема, зменшення довжини приводити до скорочення розрядності ПЗП, внаслідок чого зменшужться вартість останнього. Якщо слідувати криторію економії обладнання, то доцільно по можливості зменшувати довжину операційної частини МК.

**Розподіл МО по полях МК.** Нехай автомат керує МО , відношення між якими заданий матрицею сумісності:

Елемент - характоризує сумісність МО і , якщо МО несумісні; , якщо вони сумісні. Матриця S являється булевою і симетричною . Покладемо, що для кодування МО використовується *H* операційних полів *.* Щоб виконати МО *,* достатньо приписати її одному операційному полю *.* Мінімізуючи довжини операційної частини МК, можна вимагати, щоб число операційних полів було мінімальним і кожна МО ytбула приписане єдиному полю, тобто (10.1)

Очевидно, що в одному полі можуть розміщуватися тільки несумісні МО, тобто, якщо , то для всіх . Підмножини повинні відповідати вимозі повноти (10.2)

із якого слідує, що кожна МО винна бути включена хоча б в одне поле . Таким чином, завдання розподілу МО Y, для яких визначені відношення сумісності, зводиться до визначення мінімального числа підмножин , кожна із яких містить тільки несумісні МО і задовольняє умовам (10.1) і (10.2). Розглянемо один із способів вирішення даної задачі, що називається *методом прямого включення.*

Процес розподілу МО по підмножинам разділяється на *М* кроків. На кажному кроці для чергової МО шукають підмножину , у якові можна включити МО . МО включаеться в підмножину , якщо вона несумісна з жодною МО цієї підмножини. Якщо серед підмножин така підмножина відсутня, то для МО визначається нова підмножина , у яку включається . Стан процесу на кожному кроці будемо характоризувати матрицею включення

Елемент дорівнює 1, якщо , і дорівнює 0 у протилежному випадку. Топер умову включення МО в підмножину можна сформулювати наступним чином: МО включається в підмножину , якщо i-*а* рядок матриці S не перетинається з j-*ю* строчкою матриці *R,* тобто, якщо .

Для илюстрації методу прямого включення розглянемо приклад розподілу МО ,відношення сумісності яких задане матрицею

*Крок* 1. Множина порожня. Для МО назначаємо підмножину :

*Крок* 2. Оскільки ,то для МО назначається нова підмножина :



*Крок 3.* Оскільки *,* то МО включається в підмножину :

*Крок* 4. Оскільки , то МО включається в підмножину :

*Крок 5.* Оскільки і , то МО включається в підмножину :

*Крок* 6. Оскільки і ,то для МО назначається нова підмножина :

Продовжуючи процес, після кроку 10 отримуємо:

Цьому відповідає слідуючий розподіл МО: 

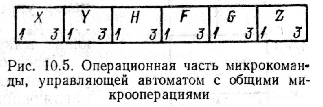
Для кодування МО потрібно  двійкових розрядів, причому код використовується в якості ознаки пустоти поля.

Розподіл МО по полях слід вважати оптимальним, якщо він мінімізує сумарну довжину полів де - кількість МО, що включені в поле . Метод прямого включення дає в більшості випадків неоптимальне рішення, яку лише в рідких випадках можна визнати задовільним.

Оптимальний розподіл МО по операційним полях будується на основі *методу знаходження максимальних сумісних підмножин,* що розроблений А. Д. Закревським.

Для задачі розподілу МО початковою є матриця сумісності *,* яка може формуватися на основі різноманітних передумов. Матриця може визначатися виходячи із функціональних властивостой МО з урахуванням структурних обмежень на сумісність, що накладуються ОА. У цьому випадку ознаки сумісності встановлюються між кожною парою МО *.* При цьому припускається, що може виникнути необхідність у сумісному використанні будь-якої парі МО і *.* Другий підхід до формування матриці базується на аналізі мікропрограм, в операторах яких перераховуються МО, що використовуються сумісно. У матриці сумісності елементу присваюється значення 1 тільки в тому випадку, коли МО і використовуються в одному операторі. У результаті цього сумісні МО і *,* але що не використовуються сумісно в мікропрограмі, відмічаються ознакою несумісності . Такий підхід приводити до зменшення кількості одиниць у матриці , у результаті чого велике число МО розглядається в якості несумісних і об’єднується в одне операційне поле. Значити, формування матриці на основі списку мікрокоманд дозволяє скоротити довжину операційної частини МК, але при цьому втрачається «універсальність» МК, оскільки зміни, що вносяться в мікропрограму, можуть породити нові функціональні оператори, що реалізуються за декілька тактів. Якщо матриця формується на основі функціональних властивостой МО, то будь-який новий функціональний оператор мікропрограми реалізується за один такт.

**Кількість операційних полів.** Якщо ОА будується на основі принципу узагальнення МО, то кількість операційних полів дорівнює числу груп несумісних МО. Так, для управління IM-автоматом (рис. 7.27) необхідні МК з шістьма операційними полями (рис. 10.5). У полях указуюються двійкові номера МО, що належати наборам і збуджуються в одному такті. Кожне операційне поле дешифрується, і сигнали, що формуються на виходах дешифраторів, поступають в операційний автомат у якості управляючих сигналів.

Якщо ОА будується на основі принципу закріплення МО і допускає сумісне виконання до HМО, то число операційних полів у мікрокоманді можна приймати рівним K=1, 2, ..., H*.* Якщо K=1, то операційна частина МК має мінімальну довжину, але в кожному такті реалізується тільки одна МО, у результаті чого функціональний оператор мікропрограми, що складається із kсумісних МО викунується за kтактів. При K=H будь-який функціональний оператор викунується за один такт, але операційне поле має максимальну довжину, що збільшує вартість ПЗП. Таким чином, зменшення кількості операційних полів дозволяє економити обладнання в КА, але одночасно зменшує швидкодію ОП.

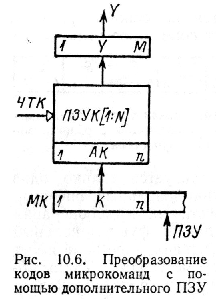
**Кодування наборів МО**

Керуючі сигнали можна формувати в автоматі наступним чином.

Нехай автомат генерує різних мікрокоманд , що складаються із наборів сумісних МО. МК і розрізняються наборами МО, що входять у них. Склад наборів можна описати булевою матрицею

(10.3)

*i-я* рядок якої описує набір і элементи приймають наступні значення: , якщо , і в протилежному випадку.

Номера наборів закодуємо n-*розрядними* двійковими позиційними кодами *K,* які будуть представлятися в операційній частині керуючого слова автомата. У такому випадку керуючі сигнали можна формувати по схемі на рис. 10.6. Мікрокоманда, що вибирається із ПЗП автомата, надходить на регістр мікрокоманди МК. Для перетворення множини кодів наборів у множину керуючих сигналів використовується ПЗП ємнітю N М-розрядних слів. Слова представляють значення рядків матриці (10.3). Код набору розглядається в якості адреси *АК* комірки ПЗП. При надходженні керуючого сигналу *ЧТК* у *ПЗУК* ініціюється операція читання Y:=ПЗУК[AK],у результаті чого на регістр МО вибирається слово. Набір одиниць у розрядах слова *Y* породжує набір керуючих сигналів , що збуджують відповідні МО.

Доцільність використання вказаного способу формування керуючих сигналів визначається витратами обладнання і часу в схемі (рис. 10.6).

Затрати обладнання, що необхідні для формування сигналів *Y,* залежативід класу мікропрограм, що реалізуються автоматом. Нехай мікропрограм а складається із *Р* мікрокоманд. Для кодування в мікрокоманді наборів МО потрібно не менше ніж розрядів, де *N* - число різних наборів. Якщо для кодування наборів МО використовується спосіб розподілу МО *Y* по полях *,* то, використовуючи припущення про рівномірний розподіл МО по полях, можна вважати, що довжина операційної частини МК буде не менше *.* Довжина МК зменшується, якщо , тобто якщо *.* Нехай *k* - довжина адресної частини. Тоді в результаті зменшення довжини операційної частини МК вартість основного ПЗП емністю -розрядних слів зменшується до *.*

Використання способу кодування наборів МО вимагає введення в склад автомата додаткового ПЗП, емністю *N М-*Розрядних слів з ціною *С(N, M).* Якщо виходити із критерію економії обладнання, то спосіб кодування наборів МО доцільний, якщо

(10.4)

Коли разрядність ПЗП має порядок , вартість ПЗП приблизно пропорційна розрядності. З урахуванням цього формулу (10.4) можна представити в наступному виді:

Спосіб кодування наборів МО приводити до збільшення тривалості такту на годину, необхідний для читання слова йз додаткового ПЗП. Затрати години на формування керуючих сигналів визначаються, в основному, годиною звернення до ПЗП, тому можна вважати, що схема (рис. 10.6) збільшує година формування керуючих сигналів приблизно у два рази.

**Тема:**  **АДРЕСАЦІЯ МІКРОКОМАНД**

В автоматах із програмованою логікою мікрокоманди виділяються своїми адресами, що визначають номера комірок ПЗП, у яких розміщаються адресовані мікрокоманди. *Спосіб адресації* мікрокоманд задає правило визначення адреси наступної мікрокоманди. Використовуються два основних способи адресації: примусова й природна адресація.

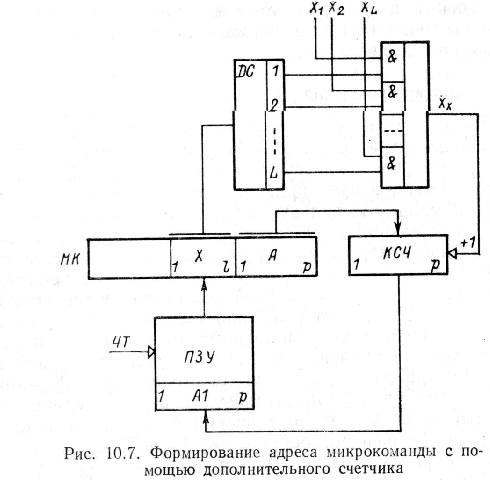
**Примусова адресація мікрокоманд**. Примусова адресація зводиться до вказівки в кожній мікрокоманді адреси наступної мікрокоманди. Цей спосіб реалізований у керуючому автоматі (рис. 10.4), що працює з мікрокомандами (рис. 10.1). Адреса наступної мікрокоманди визначається залежно від коду *Х* і значення *х*Хабо полем А0*,* або полем А1.

*З* метою скорочення довжини мікрокоманди для формування адреси наступної мікрокоманди виділяється єдине поле *А.* Якщо поле Х=0, то значення *А* безумовно визначає адресу наступної мікрокоманди. Якщо Х≠0, то адреса наступної мікрокоманди дорівнює (А+*х*Х)*,* де *х*Х- значення логічної умови з номером *X.* У результаті цього реалізується умовний перехід: якщо *х*Х=0, то до мікрокоманди з адресою *А;* якщо *х*Х =1, то до мікрокоманди з адресою *(А* + 1). Зазначений порядок формування адреси реалізується схемою на рис. 10.7. Виконавча адреса А1=А+*х*Хформується лічильником *КСЧ* комбінаційного типу. Скорочення довжини слова ПЗП на *р* двійкових розрядів знижує вартість ПЗП на (p/k)C(P,k)*,* де C(P,k)- вартість ПЗП ємністю *Р k-розрядних* слів. Якщо вартість р-*розрядного* комбінаційного лічильника дорівнює S(p)*,* то схема на рис. 10.7 приводить до економії встаткування в порівнянні зі схемою на рис. 10.4, коли

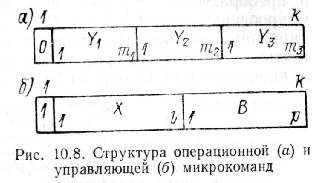


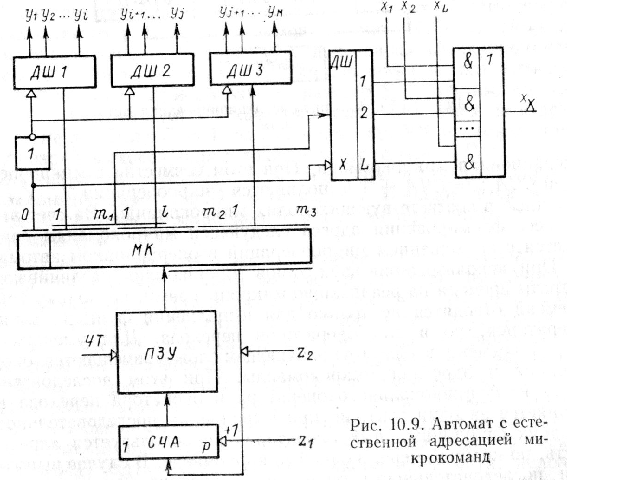
Введення лічильника в схему зменшує швидкодія автомата, оскільки тривалість такту повинна бути збільшена на час виконання МО рахунку в р-*розрядному* лічильнику.

**Природна адресація мікрокоманд.** При природній адресації адреса наступної мікрокоманди приймається рівною збільшеній на одиницю адресі попередньої мікрокоманди, тобто якщо *А* - адреса виконуваної мікрокоманди, то наступна мікрокоманда вибирається з комірки з адресою *(А* + 1). При природній адресації відпадає необхідність у введенні адресного поля в кожну мікрокоманду. Якщо мікрокоманди слідують у природному порядку, то процес адресації реалізується лічильником адреси мікрокоманди, стан якого збільшується на одиницю після читання чергової мікрокоманди. Отже, мікрокоманди, які задають функціональні перетворення, що складаються з набору мікрооперацій, можуть містити тільки операційну частину, що представляється полями Y1, Y2, …, YH*.*



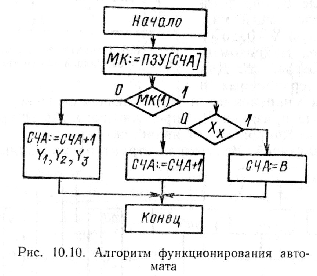
Після виконання мікрокоманди з адресою *А* може виникнути необхідність у переході до мікрокоманди з адресою B≠А+1. Перехід може бути безумовним або залежати від поточного значення *х*Х*.* Умовні переходи реалізуються в такий спосіб: якщо *х*Х=0, то виконується наступна мікрокоманда з адресою (*А+*1); якщо *х*Х=1, то наступною виконується мікрокоманда з адресою *В.* Для реалізації умовних переходів у мікрокоманду вводиться адресна частина, що складається з полів X і В.

При природній адресації, як правило, використовуються мікрокоманди двох типів: *операційні й керуючі.* Операційна мікрокоманда задає набір мікрооперацій Y1, Y2, …, YHі неявно задає адресу наступної мікрокоманди рівною (*А+*1). Керуючі мікрокоманди використовуються для зміни природного порядку проходження мікрокоманд, що зводиться до виконання безумовних і умовних переходів. Керуюча мікрокоманда містить поле X, що визначає номер логічної умови, і поле *В, що* визначає адресу наступної мікрокоманди. Якщо Х=0, то адреса наступної мікрокоманди безумовно дорівнює *В.* Для виділення операційних і керуючих мікрокоманд у керуючому слові вводиться однорозрядне поле ознаки *Р, що* визначає тип мікрокоманди: якщо Р=0, то мікрокоманда є операційною; якщо Р=1 - керуючою. Можлива структура керуючих слів, побудованих по зазначеному способу, зображена на рис. 10.8.



Ознака *Р* розміщається в першому розряді керуючого слова.

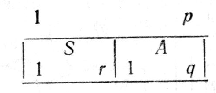
Автомат, що працює з мікрокомандами (рис. 10.8), будується за схемою на рис. 10.9. Дешифратори ДШ1, ДШ2, ДШ3*,* на виході яких формуються керуючі сигнали y1, y2, …, yM стробуються сигналом *Р, що* приймає значення 1 при виконанні операційної мікрокоманди. Дешифратор *ДШХ* стробується сигналом *Р* = *МК*(1), що дорівнює 1 при обробці керуючої мікрокоманди. Адреса мікрокоманди зберігається й перетворюється на лічильнику адреси *СЧА,* з яким зв'язані МО z1: СЧА:=СЧА+1 й z2: СЧА:=В*.* Функціонування автомата розділяється на послідовність мікротактів, протягом яких виконуються дії, обумовлені мікропрограмою на рис. 10.10. Якщо МК(1)=0, то виконується



операційна мікрокоманда. При цьому разом з мікрооперацією СЧА : = СЧА + 1 виконуються МО Y1, Y2, Y3*,* задані у відповідних полях мікрокоманди. За рахунок цього процес формування адреси наступної мікрокоманди сполучається з виконанням мікрооперацій в операційному автоматі. При використанні двох типів мікрокоманд збільшуються витрати часу на реалізацію мікропрограм, оскільки такти часу приділяються не тільки для виконання функціональних операторів, але й для операторів переходу. Для зменшення витрат часу в одному керуючому слові сполучаються операційна й адресна мікрокоманди. При цьому послідовність із функціонального оператора й оператора переходу виконується за один такт, але при виконанні послідовності із двох функціональних операторів не використовується адресна частина, принаймні, в одній мікрокоманді. У випадку виконання послідовності операторів переходу не використовується операційна частина більшості мікрокоманд. Таким чином, прагнення підвищити продуктивність КА приводить до непродуктивного використання ємності ПЗП, у результаті чого збільшуються витрати устаткування на реалізацію мікропрограми.

**Тема:** **СЕГМЕНТАЦІЯ ПЗП.**

Для зменшення довжини адресної частини мікрокоманди використовується спосіб сегментації. ПЗП розділяється на сегменти, що складаються із 2q сусідніх комірок. При цьому адреса комірки ПЗП розділяється на два поля:



Поле S визначає адресу сегмента, а поле *А* - адресу комірки в сегменті S. Таким чином, комірки 0, 1, …, 2q утворюють сегмент 0, комірки 2q, 2q+1, …, 2q+1-1 - сегмент 1 і т.д.

Адресація мікрокоманд виконується в такий спосіб. Спеціальною мікрокомандою встановлюється адреса сегмента S. У наступних мікрокомандах вказується адреса комірки *А* в сегменті. Оскільки адреса *А* є лише частиною повної р-*розрядної* адреси комірки ПЗП, то адресна частина мікрокоманди скорочується на k(p-q)розрядів, де *k -* кількість адрес у мікрокоманді.

У випадку примусової адресації обробка сегментованих адрес реалізується схемою на рис. 10.11, *а.* За аналогією з мікрокомандами на рис. 10.1 адресна частина мікрокоманди розділена на поля X, A0 й A1*.* Для зберігання адреси сегмента в схемі (рис. 10.11, а)використовується регістр S. Допустимо, що код *X* = 11...1 використовується як ознака s переходу до іншого сегмента. Якщо s = 1, то мікрокоманда МК змінює старші rрозрядів адреси, що визначають адресу сегмента. Якщо s = 0, то мікрокоманда змінює тільки молодші *q* розрядів адреси, що визначають адресу комірки в сегменті S. Порядок обчислення адреси *А* наступної мікрокоманди описаний фрагментом мікропрограми (рис. 10.11, б).



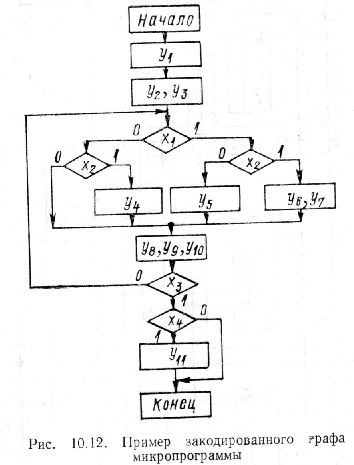
У випадку природної адресації обробка сегментованих адрес реалізується за схемою на рис. 10.11, *в.* У керуючій мікрокоманді (P=1) виділені три поля: S - однорозрядна ознака зміни сегмента; *X* - номер логічної умови; *А -* адреса комірки сегмента. У даному варіанті прийнято, що адреса сегмента розміщується в полі X. Мікрокоманда обробляється в схемі (рис. 10.11, б) у порядку, обумовленому рис. 10.11, *г.*

Прі сегментації ПЗП зменшується довжина основних мікрокоманд, але одночасно із цим у мікропрограму включаються додаткові мікрокоманди, необхідні для передачі керування між сегментами. Величина економії пам'яті, використовуваної для зберігання основних мікрокоманд, і ємність пам'яті, що споживається додатковими мікрокомандами, залежать від розміру сегмента.

Сегментація ПЗП дозволяє заощаджувати устаткування, але через введення в мікропрограму додаткових мікрокоманд збільшується час виконання мікропрограм. Якщо мікропрограма виконується в середньому за *t* тактів, то при сегментації час виконання мікропрограм збільшується в середньому до (1 + *Q) t* тактів, де *Q* - імовірність появи додаткової мікрокоманди.

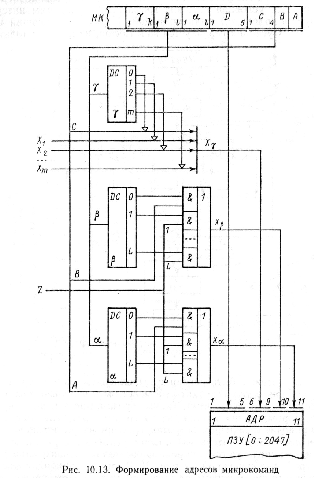
Шляхом раціонального (мимовільного) розміщення мікропрограми по сегментах можна зменшити число переходів між сегментами. У результаті цього зменшується ємність ПЗП і витрати часу на виконання мікропрограми.

**Формування адрес мікрокоманд.** Переходи між функціональними операторами мікропрограми можуть залежати від значень декількох логічних умов. Наприклад, у мікропрограмі (рис. 10.12) напрямок переходу від оператора y2, y3залежить від значень логічних умов x1і x2*,* і спадкоємцем мікрокоманди, що збуджує МО y2, y3*,* є одна із чотирьох мікрокоманд. Якщо в мікрокоманді виділяється єдине поле *X* і визначаються дві можливих адреси A0й A1наступної мікрокоманди, то при програмуванні переходів кожна умовна вершина графа мікропрограми інтерпретується окремою мікрокомандою. Перехід, що залежить від *k* логічних умов реалізується за *k* тактів, з яких тільки



перший такт може бути пов'язаний з виконанням мікрооперацій, у той час як інші такти не можуть бути використані для керування мікроопераціями. Таким чином, застосування мікрокоманд, що породжують тільки два можливих напрямки переходу, приводить до більших витрат часу на реалізацію мікропрограм зі значним числом умовних вершин. Якщо в автоматі використовуються мікрокоманди тільки одного типу (зі структурою, зображеної на рис. 10.1), то операційна частина в багатьох мікрокомандах виявляється порожньою, що свідчить про неефективне використання інформаційної ємності окремих керуючих слів і ПЗП у цілому.

При використанні способу примусової адресації для скорочення числа адрес у мікрокоманді й витрат часу на реалізацію складних переходів застосовується метод формування адрес мікрокоманд. Цей метод реалізований, у частку в ЕОМ серії IBM/360 [18].



Адреси мікрокоманд формуються схемою рис. 10.13. У цьому випадку адреса представляється в мікрокоманді *МК 11-*розрядним кодом і розділяється на поля *D, С, В* і *А.* Поле D визначає адресу сегмента, що складається із 64 сусідніх комірок ПЗП. Адреса комірки в сегменті *D* представляється складеним словом С.В.А*,* поле С якого вказує адресу однієї з 16 зон сегмента, що складається з 4 сусідніх комірок. Адреса комірки в зоні С задана 2-розрядним кодом *В. А.* З множин логічних умов *X* виділяються m=2k-1 4-х розрядних набори . Інші логічні умови розглядаються як самостійні одиниці. Умови, значення яких перевіряється мікрокомандою для формування адреси наступної мікрокоманди, виділяються полями й . Поле визначає номер набору й поля й - номера умов з множини Z. Якщо поле або є порожнім (містить код 0), то адреса наступної мікрокоманди формується незалежно від умов, пов'язаних з відповідним полем. Якщо рівні 0, то адреса наступної мікрокоманди є безумовним У протилежному випадку адреса формується залежно від поточних значень логічних умов , у результаті чого зона в сегменті D визначається значенням набору логічних умов і одна з 4-х комірок зони - значеннями й . Таким чином, полючи визначають найменування логічних умов, значення яких породжують 64 можливі адреси наступної мікрокоманди, що належить сегменту D, тобто визначають 64 різні напрямки переходу. Якщо й , , то мікрокоманда породжує чотири напрямки переходу. Вибір напрямку переходу відбувається залежно від значень 1, 2, 4, 5 або 6 логічних умов.

Метод формування адрес мікрокоманд є досить гнучким при організації складних переходів, напрямок яких залежить від значень декількох логічних умов. Оскільки одна мікрокоманда породжує велику кількість напрямків переходу, скорочуються витрати часу на реалізацію мікропрограм. Однак розглянутий метод накладає жорсткі обмеження на порядок розміщення мікрокоманд у ПЗП: мікрокоманда повинна розміщатися в тій комірці сегмента, адреса якої співпадає зі значенням набору логічних умов, що викликають перехід до даної мікрокоманди. Ця вимога значно ускладнює процес розробки мікропрограм, породжуючи необхідність у використанні складних алгоритмів розподілу мікрокоманд в адресному полі ПЗП.

**Тема:** **СТРУКТУРНІ МЕТОДИ ПІДВИЩЕННЯ ШВИДКОДІЇ АВТОМАТІВ**

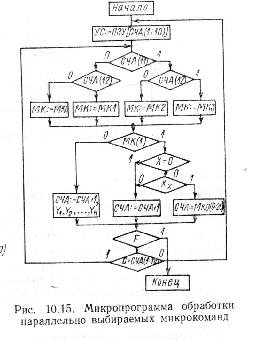
Швидкодія автомата характеризується часом, затрачуваним на формування одного набору керуючих сигналів. Цей час складається із трьох складових: 1) часу формування адреси наступної мікрокоманди; 2) часу звернення до ПЗП; 3) часу дешифрування операційної частини мікрокоманди. Основна частка часу припадає на читання мікрокоманд із ПЗП, тому відчутне збільшення швидкодії автомата може досягатися або за рахунок зменшення часу звернення до ПЗП, або за рахунок скорочення числа звернень, що виконуються у процесі функціонування автомата. Зменшення часу звернення до ПЗП досягається шляхом використання в ПЗП більш швидкодіючих елементів. При фіксованій швидкодії ПЗП швидкодію автомата можна підвищити за рахунок використання спеціальних структурних рішень.

**Паралельна вибірка мікрокоманд.** Довжину слова ПЗП можна призначити рівній сумарній довжині *К* мікрокоманд. У такому випадку за одне звертання до ПЗП вибирається слово, що містить *До* мікрокоманд, які будуть оброблятися послідовно в порядку, диктуемом мікропрограмою. Очевидно, що з *К* обраних мікрокоманд реалізується в середньому 1 < *N* < *К* мікрокоманд, у результаті чого витрати часу на вибірку із ПЗП однієї мікрокоманди зменшуються в *N* раз, тобто ефективна швидкодія ПЗП збільшується в *N* раз.

Автомати, одне слово яких містить кілька мікрокоманд, називаються автоматами з паралельною вибіркою мікрокоманд. Паралельна вибірка використовується винятково з метою збільшення швидкодії.

Як приклад організації паралельної вибірки мікрокоманд розглянемо автомат, у якому застосовуються спосіб природної адресації й мікрокоманди двох типів: операційні й керуючі. Тип мікрокоманди кодується її старшим розрядом: операційні мікрокоманди відзначаються значенням 0 і керуючі - значенням 1. Схема автомата зі словами, що представляють чотири мікрокоманди, зображена на рис. 10.14 і порядок його функціонування - на рис. 10.15. Адресація мікрокоманд забезпечується за допомогою лічильника адреси *СЧА,* десять старших розрядів якого визначають адресу слова в ПЗП і два молодших розряди - адресу мікрокоманди в даному слові. Робота автомата починається із читання слова, адреса якого задана полем *СЧА* (1:10). Слово, вибране із ПЗП, заноситься на регістр керуючого слова УС. Одночасно із цим адреса обраного слова запам'ятовується на регістрі *С*(10). Керуюче слово *УС* розділяється на чотири сегменти МК0, МК1, МК2, МК3*,* кожен з яких представляє одну мікрокоманду. Стан СЧА(11:12), що визначає адресу мікрокоманди в слові, дешифрируется, і відповідний сегмент керуючого слова надходить на входи декодувальних ланцюгів автомата.

Виконання мікрокоманди зводиться до збудження мікрооперацій, зазначених в операційних полях Y1, Y2, …, YHмікрокоманди, і (або) зміни стану лічильника *СЧА,* у результаті чого на лічильнику буде сформована адреса наступної мікрокоманди. Наступна мікрокоманда може належати раніше обраному слову УС або будь-якому іншому слову із числа збережених у ПЗП. Ці ситуації розділяються шляхом порівняння на рівність адреси *С* слова УС і адреси наступного УС, що представляється в *СЧА*(1:10). Рівність адрес *С* = *СЧА*(1:10) свідчить про те, що наступна мікрокоманда міститься в слові УС, і вибірка мікрокоманди виробляється шляхом передачі слова МК0, МК1, МК2 або МК3 в шину МК без звернення до ПЗП. У протилежному випадку із ПЗП вибирається нове керуюче слово УС.



Функціонування автомата триває до одержання мікрокоманди, що містить мікрооперацію *F, що* припиняє роботу автомата.

Ефективність паралельної вибірки мікрокоманд можна характеризувати числом звернень *k* до ПЗП, необхідних для реалізації однієї мікрокоманди. Очевидно, що число звернень *k* не може бути більше 1, оскільки в кожному обраному слові реалізується хоча б одна мікрокоманда. Наявність у мікропрограмі циклів, які охоплюють мікрокоманди, розміщені в одному керуючому слові, створює можливість отримання миттєвих значень , де *К -* кількість паралельно вибраних мікрокоманд.

**Випереджаюча вибірка мікрокоманд.** Такт *Т* роботи операційного пристрою визначається (рис. 10.16, *а)* сумою часу Ty*,* що витрачається на формування адреси Aй вибірку мікрокоманди B, і часу T0 роботи операційного автомата, що складається із часу виконання МО Y й обчислення логічних умов X.

З метою зменшення тривалості такту можна починати вибірку наступної мікрокоманди ще до моменту закінчення мікрооперацій (рис. 10.16, *б),* тобто вибирати мікрокоманди з випередженням у часі. У результаті цього процес вибірки наступної мікрокоманди

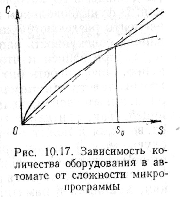


сполучається із процесом реалізації попередньої мікрокоманди в операційному автоматі й такт роботи пристрої буде дорівнює *,* причому *.* Адреса наступної мікрокоманди визначається тільки приблизно, оскільки вона може залежати від значень логічних умов, що переобчислюються виконуваною мікрокомандою. При використанні способу примусової адресації, коли в мікрокоманді вказуються дві можливих адреси А0й А1наступної мікрокоманди, можна апріорно покласти, що адреса наступної мікрокоманди дорівнює А0.

**Порівняння характеристик керуючих автоматів с твердою й програмованою логікою**

Та сама мікропрограма може бути реалізована автоматом як із твердою, так і із програмованою логікою. Розходження в принципах організації керуючих автоматів породжує розходження у витратах устаткування, необхідного для реалізації тих самих функцій, тобто у вартості автоматів. Кількість устаткування в автоматі залежить, у першу чергу, від складності мікропрограми, що інтерпретується автоматом. У першому наближенні ступінь складності мікропрограми можна характеризувати сумарним числом операторних і умовних вершин. Якісний характер залежності кількості устаткування *С* у автоматі від складності S мікропрограми, що інтерпретується автоматом, ілюструється рис. 10.17. Кількість устаткування в автоматі із твердою логікою зростає майже пропорційно складності мікропрограми. Для автоматів із програмованою логікою типові значні питомі витрати встаткування при реалізації відносно нескладних мікропрограм. При реалізації складних мікропрограм питомі витрати устаткування зменшуються. З рис. 10.17 слідує, що в області малих значень 0<S<S0 більш економічними є автомати із твердою логікою (1), а в області більших значень S>S0 - автомати із програмованою логікою (2). Як правило, значення S0лежить у межах 150-250 вершин графа мікропрограми.

Автомати із твердою логікою мають більш високу швидкодію, чим автомати із програмованою логікою. Якщо - витрати часу на формування керуючих



сигналів (сума часу формування сигналів збудження, перемикання стану пам'яті автомата й формування керуючих сигналів), то витрати часу на формування керуючих сигналів в автоматах із програмованою логікою рівні , де - тривалість циклу ПЗП. Таким чином, при використанні автомата із програмованою логікою тривалість такту роботи пристрою збільшується на , тобто на 0,4 - 1,0 мкс. Втрати часу можна зменшити в кілька разів за рахунок використання паралельної вибірки мікрокоманд і сполучення процесу вибірки наступної мікрокоманди з виконанням попередньої.

Керуючі автомати із твердою й програмованою логікою відрізняються по ступені гнучкості, що визначає можливість внесення змін у закон функціонування автомата при зміні мікропрограми. Необхідність у зміні мікропрограми досить часто виникає в процесі проектування ЕОМ через наявних у мікропрограмі помилок. Зміна мікропрограми приводить до необхідності перебудови структури автомата, що може торкнутися більшості елементів і ланцюгів, тобто більшою мірою або повністю змінити конфігурацію автомата. Програмована логіка дозволяє вирішити завдання перебудови відносно просто - шляхом введення в ПЗП додаткових мікрокоманд і зміни кодів окремих мікрокоманд. Таким чином, можна вважати, що гнучкістю володіють тільки автомати із програмованою логікою.

Ряд додаткових міркувань про області застосування автоматів із твердою й програмованою логікою висловлений в [68].