1. Катя Новикова +
2. Nik Дворянинов
3. Александра Бондаренко
4. Саня Бурлака
5. Александр Матвиенко +
6. Павел Сырых
7. Max Kovtun
8. Андрей Сердега
9. Вова Басюк
10. Владислав Тютюнник
11. Юра Пионтковский
12. Діма Колежнюк
13. Вова Козырь
14. Антон Долгополов
15. Юрий Иванов
16. Леонид Чепель
17. Веталь Григоренко
18. Misha Smaga +
19. Александр Матвиенко +
20. Павел Сырых
21. Max Kovtun
22. Юра Пионтковский
23. Владислав Тютюнник
24. Діма Колежнюк
25. Nik Дворянинов
26. Катя Новикова +
27. Антон Долгополов
28. Вова Козырь
29. Веталь Григоренко (Савчук Роман)
30. Саня Бурлака
31. Вова Басюк
32. Андрей Сердега
33. Юрий Иванов
34. Александра Бондаренко
35. Леонид Чепель

36. Misha Smaga +
37. Владислав Тютюнник
38. Nik Дворянинов
39. Александр Матвиенко +
40. Діма Колежнюк
41. Вова Басюк
42. Саня Бурлака
43. Катя Новикова+
44. Александра Бондаренко (Савчук Роман)
45. Юра Пионтковский
46. Леонид Чепель
47. Павел Сырых
48. Андрей Сердега
49. Юрий Иванов (Савчук Роман)
50. Misha Smaga
51. Антон Долгополов
52. Веталь Григоренко
53. Вова Козырь
54. Max Kovtun
55. Александр Матвиенко
56. Андрей Сердега
57. Misha Smaga +
58. Павел Сырых
59. Діма Колежнюк
60. Владислав Тютюнник
61. Юра Пионтковский
62. Веталь Григоренко
63. Леонид Чепель
64. Вова Басюк
65. Max Kovtun
66. Юрий Иванов
67. Вова Козырь
68. Саня Бурлака
69. Антон Долгополов
70. Александра Бондаренко

71. Катя Новикова
72. Nik Дворянинов

1. Оцінка якості функціональних схем.
2. Серії та види логічних елементів.
3. Основи кубічного числення
4. Метод мінімізації Квайна
5. Знаходження покриття ф-цій методом Петрика
6. Мінімізація систем булевих функцій
7. Суматор. Схема, принцип дії, застосування.
8. Мультиплексор. Схема, принцип дії, застосування.
9. Демультиплексор. Схема, принцип дії, застосування.
10. Шифратор (кодер). Схема, принцип дії, застосування.
11. Дешифратор. Схема, принцип дії, застосування.
12. Програмуємі логічні схеми. Принцип дії, застосування.
13. Контроль по Хеммінгу (https://habrahabr.ru/post/140611/)
14. Асинхронні тригери. Схеми, принцип дії, застосування.
15. Синхронні тригери. Схеми, принцип дії, застосування.
16. Одноступінчаті та двоступінчаті тригери (Master-Slave). Схеми, принцип дії, застосування.(Leo)
17. RS-тригери. Схеми, принцип дії, застосування.
18. JK-тригери. Схеми, принцип дії, застосування. [Misha Smaga]
19. D-тригери. Схеми, принцип дії, застосування.
20. T-тригери. Схеми, принцип дії, застосування.
21. Регістри (послідовні, паралельні). Схеми, принцип дії, застосування.
22. Лічильники (прямі, обернені). Схеми, принцип дії, застосування.
23. Синтез керуючого автомата Мура на базі регістра зсуву.
24. Структурний синтез автомата Мілі.
25. Структурний синтез автомата Мура.
26. Компаратор. Схема, принцип дії, застосування.
27. Перетворення з коду Грея до двійкового позиційного, і навпаки.

28. Програмуємі логічні матриці.

29. Перемножувачі. Схеми, принцип дії, застосування.

30. Перехідні процеси в логічних схемах.[більш повна інфа http://prom-komplekt.com/content/perekhodnye-protsessy-v-logicheskikh-skhemakh]

31. Перегони в цифрових схемах.

32. Перегони по виходу.

33. Тригери. Схеми, принцип дії, застосування.

34. Шестиелементні тригери (Вебба). Схеми, принцип дії, застосування.

35. Послідовно-паралельні регістри. Схеми, принцип дії, застосування.

36. Однофазна та двофазна синхронізація. [Misha Smaga]

37. Циклічний регістр зсуву. Схема, принцип дії, застосування.

38. Схема регістру зсуву, що дозволяє зсувати сигнал як праворуч, так і ліворуч.

39. Правила переходу з КНФ до базису Пірса (або-ні).

40. Правила переходу з ДНФ до базису Шефера (і-ні).

41. Кубічне представлення логічних функцій.

42. Двійково-десяткові коди.

43. Метод підбору коефіцієнтів (перевод чисел з однієї позиційної системи числень в іншу)

44. Метод переводу діленням на основу нової системи (перевод чисел з однієї позиційної системи числень в іншу)

45. Метод переводу чисел діленням на основу в додатній степені (перевод чисел з однієї позиційної системи числень в іншу)

46. Перевід правильних дробів множенням на основу системи (перевод чисел з однієї позиційної системи числень в іншу)

47. Кодування від’ємних чисел в ЕОМ

48. Форми подання чисел в ЕОМ

49. Машинні алгоритми перетворення чисел

50. Додавання чисел із знаком у машинних кодах [Misha Smaga]

51. Додавання і віднімання чисел у зворотних кодах. Побудувати схему.

52. Додавання і віднімання чисел у доповнювальних кодах. Побудувати схему.

53. Зсуви машинних кодів

54. Арифметичний зсуви чисел поданих у зворотному коді.

55. Арифметичний зсуви чисел поданих у доповнювальному коді.

56. Множення чисел (1- спосіб)

57. Множення чисел (2- спосіб) [Misha Smaga]

58. Множення чисел (3- спосіб)

59. Множення чисел (4- спосіб)

60. Прискорення множення чисел поданих паралельним кодом (схема матриці суматорів із розповсюдженням переносів по рядкам)

61. Прискорення множення чисел поданих паралельним кодом (схема матриці суматорів із розповсюдженням переносів по діагоналі)

62. Ділення чисел без відновлення від'ємного залишку (з подвоєнням залишку)

63. Ділення чисел без відновлення від'ємного залишку (з зменшенням вдічі дільника)

64. Додавання чисел з плаваючою комою.

65. Множення чисел з плаваючою комою.

66. Ділення чисел з плаваючою комою.

1. Оцінка якості функціональних схем.

Оцінка по двум параметрам – затримці T та апаратним затратам W. Найчастіше, для обрахування середньої затримки на елементах, вважають що елементи І-НЕ,І,І-АБО-НЕ мають однакову затримку яка дорівнює t. Для підрахунку апаратної затримки, використовують відношення площі корпусів. Найбільший за площею приймається за 1, а всі інші рахуються відносно нього, потім все додається. Підрахунки зручно виконувати у 1/12 долях корпуса, бо 12 – це число логічних виводів корпусу найменшого розміру. Складність схеми оцінюється кількістю елементів, що входить в схему. У теоретичних розробках орієнтуються на довільну елементну базу і тому для оцінки витрат устаткування використовується оцінка складності схем по Квайну. Складність за Квайну визначається сумарним числом входів логічних елементів у складі схеми (Cb). При такій оцінці одиниця складності - один вхід логічного елемента. Такий підхід до оцінки складності виправданий з наступних причин: 3 - Складність схеми легко обчислюється по булевим функціям , на основі яких будується схема. Усі класичні методи мінімізації булевих функцій забезпечують мінімальність схеми саме в сенсі ціни по Квайну . Практика показує , що схема з мінімальною ціною по Квайну зазвичай реалізується найменшим числом конструктивних елементів - корпусів інтегральних мікросхем .

2. Серії та види логічних елементів.

Серией микросхем называют группу микросхем, выполненных по одинаковой или близкой технологии , имеющие близкие технические характеристики и предназначены для совместной работы в составе цифровой аппаратуры. Условные обозначения :

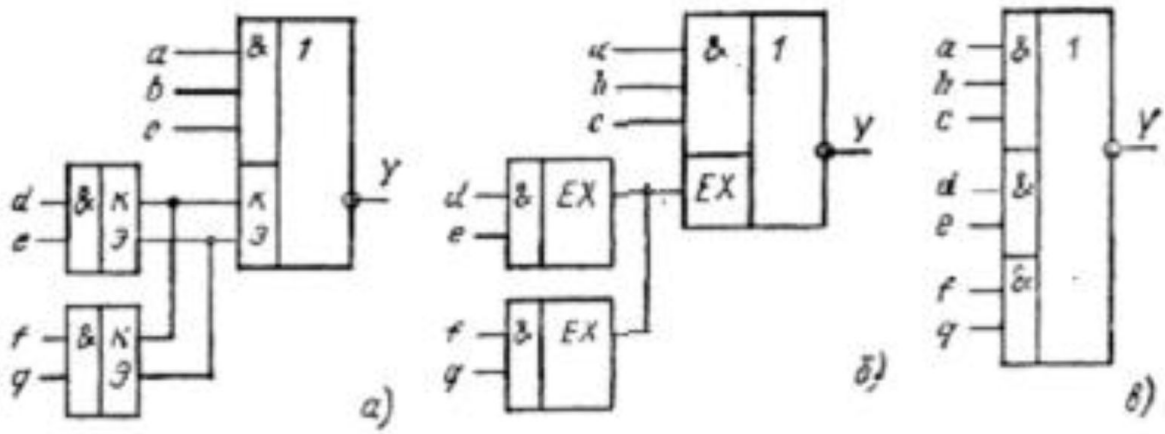
- буквы , характеризируют стойкость микросхемы к воздействию окружающей среды и связанный с этим тип корпуса. Отсутствие буквы означает что там «нулевая буква»
- из трех или четырех цифр обозначают номер серии. Предыдущая буква считается первым наименованием серии
-две буквы - характеризуют выполняемую функцию
-одна или две цифры-тип микросхемы внутри функциональной группы

-одна буква-характеризирующая возможные вариации значений некоторых параметров (ее чаще всего не бывает)

Пример: К155ЛА2 –микросхема К155 , выполняет И-НЕ, второго типа . Для ТТЛ и ТТЛШ используют элемент И-НЕ .Они чаще всего используются для построения сложных узлов.Вторая особенность–простота изготовления двухступенчатых элементов И-ИЛИ-НЕ.
Обозначение поясняется в следующих примерах :

2-2-2И-3ИЛИ-НЕ –фун. Y= N(abVcdVef); 3-3И-2ИЛИ-НЕ – Y=N(abcVdef); 2-2-4И-3ИЛИ-НЕ – Y=N(abVcdVejgh)

В серии ТТЛ есть расширяемые элементы И-ИЛИ-НЕ к которым подключают расширители увеличивая количество входов ИЛИ. Сумма входов ИЛИ внутру корпуса и добавляемых за счет расширителей , обычно ограничена 8.Выпускают расширители двух типов – с числом входов И равным 4 и 8 . Тут показано различные по степени подробности способы изображения расширяемого элемента И-ИЛИ-НЕ с подключаемым расширителем.(а- для электрических схем, б- для функциональных подробных схем, в-для общих функциональных схем).



При оценке общей задержки цифрового блока нужно суммировать не какие-то унифицированные задержки функциональных узлов , а задержки именно по тем трактам которые срабатывают при выполнение узлом рассматриваемой функции . В проэктируемой схеме нужно уметь выделять тракты срабатывающие в раличных режимах , группировать

идентичные входы ,оценивать влияние трактов друг на друга. Списки задержек описаны в справочниках. Краткие справочники ограничиваются задержками по самым длинным трактам. Такая информация не дает возможности реализовать все быстродействие схемы

3. Основи кубічного числення

Будь-який набір змінних в кубічному представленні логічних функцій прийнято називати кубом або вектором в n-вимірному просторі. Змінні куба прийнято називати координатами, а компоненти вектора - розрядами. Кількість змінних в кубі визначає його мірність (розрядність вектора). Для представлення логічних функцій в кубічної форми використовується чотирьох символьний теоретико-множинний алфавіт, заснований на двох логічних примітивах 0 і 1, тобто кубічний алфавіт - A = {0, 1, U, X}. Символ X в цьому алфавіті може розглядатися як універсум, тобто X = 0 ∪ 1 ∪ U. Ø("порожньо") і U ("unnown state") розглядаються як синоніми. Кодовою відстанню d між двома кубами називається кількість координат які відрізняються (розрядів). Наприклад, якщо A = 0XX0 і B = 01X1, то кодова відстань між ними дорівнює 2 (d = 2). Кількість символів X в кубі визначає його ранг. Наприклад, куб 0011 має нульовий ранг, 0X11 - перший ранг, XX11 - другий ранг і т.д. Сукупність кубів, що покривають всі вхідні набори функції, називається покриттям або кубічним покриттям (КП). Безліч кубів, на яких функція дорівнює 0, називають нульовим покриттям (C^0), а на яких дорівнює 1 - одиничним покриттям (C^1).

Перетин: C = A ∩ B

∩	0	1	x
0	0	Ø	0
1	Ø	1	1
x	0	1	x

Операції поглинання: A є B, якщо

Обєднання C = A ∪ B = ((a1 ∪ b1),...(ai ∪ bi))

∪	0	1	x
0	0	X	X
1	X	1	X
x	X	X	X

<table>
<tr>
<td>A ∩ B = A
0X0X ∩ X10X = 010X
0X0X ∩ 11øX = ø
0X00 ∩ 0X0X = 0X00</td>
<td>0X0X ∪ 0X01 = 0X0X
0X01 ∪ 0X00 = 0X0X
0101 ∪ 0110 = 01XX</td>
</tr>
<tr>
<td>Кубічна операція конюнкція
C = A & B = ((a₁&b₁),....,(a_n&b_n))
<table>
<tr><td>&</td><td>0</td><td>1</td><td>x</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>X</td></tr>
<tr><td>x</td><td>0</td><td>X</td><td>X</td></tr>
</table>
0X01 & 11X1 = 0X01</td>
<td>Кубічна операція дизюнкції
C = A ∨ B = ((a₁∨b₁),....,(a_n,b_n))
<table>
<tr><td>∨</td><td>0</td><td>1</td><td>x</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>X</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>x</td><td>X</td><td>1</td><td>X</td></tr>
</table>
0X01 ∨ 11X1 = 11X1
0110∨01XX = 011X</td>
</tr>
<tr>
<td>Виключне Або
C = A ⊕ B = ((a₁ ⊕b₁),....,(a_n ⊕b_n))
<table>
<tr><td>⊕</td><td>0</td><td>1</td><td>X</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>X</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table>
0X01 ⊕ 11X1 = 1XX0
0110 ⊕ 0101 = 0011</td>
<td></td>
</tr>
</table>

4. Метод мінімізації Квайна

Вихідною формою подання перемикальної функції для виконання мінімізації методом Квайна є

ДДНФ. Метод базується на використанні наступних співвідношень:

- $Ax \vee A\overline{x} = Ax \vee A\overline{x} \vee A$ – неповне склеювання;
- $BC \vee C = C$ – поглинання.

Етапи алгоритму:

1. Записати перемикальну функцію у вихідній формі, якою є ДДНФ.

$$f = xyz \bigvee xy\overline{z} \bigvee x\overline{y}z \bigvee \overline{x}yz \bigvee \overline{xyz} \bigvee \overline{x}y\overline{z}$$

2. Застосувати співвідношення неповного склеювання послідовно до конституент одиниці (шматків ДДНФ), потім до імплікант (n - 1)-го рангу, (n - 2)-го рангу і т. д., поки формування нових імплікант можливе.

- $xyz \vee xy\overline{z} = xyz \vee xy\overline{z} \vee xy$, тут xyz і $xy\overline{z}$ – конституенти одиниці (3-ій ранг), xy – імпліканта 2-го рангу;
- $xy\overline{z} \vee \overline{x}y\overline{z} = xy\overline{z} \vee \overline{x}y\overline{z} \vee y\overline{z}$;
- $xyz \vee x\overline{y}z = xyz \vee x\overline{y}z \vee xz$;
- $x\overline{y}z \vee \overline{xy}z = x\overline{y}z \vee \overline{xy}z \vee \overline{y}z$;
- $\overline{xy}z \vee \overline{xyz} = \overline{xy}z \vee \overline{xyz} \vee \overline{xy}$;
- $\overline{xyz} \vee \overline{x}y\overline{z} = \overline{xyz} \vee \overline{x}y\overline{z} \vee \overline{xz}$;

Подальше утворення імплікант із новоутворених імплікант неможливе

3. Виконати всі можливі поглинання. Результат – визначаються всі прості імпліканти (більше не можуть поглинати/поглинатися), які складають скорочену ДНФ.

$$f = xy \bigvee y\overline{z} \bigvee xz \bigvee \overline{y}z \bigvee \overline{xy} \bigvee \overline{xz}$$

4. Побудувати таблицю покриття (імплікантну матрицю) для подальшого спрощення запису функції:

кількість рядків таблиці дорівнює кількості отриманих простих імплікант, а кількість стовпців збігається з кількістю конституент одиниці ДДНФ. Якщо в деяку конституенту ДДНФ входить будь-яка з простих імплікант, то на перетині відповідного стовпця і рядка ставиться позначка;

знаходження суттєвих імплікант: якщо в будь-якому із стовпців таблиці є тільки одна позначка, то проста імпліканта у відповідному рядку є суттєвою. Стовпці, які відповідають суттєвим імплікантам, викреслюються;

викреслення зайвих стовпців: якщо в таблиці є два стовпці, у яких всі позначки в однакових рядках, то один із стовпців викреслюється;

викреслення зайвих простих імплікант: якщо з'являються рядки, які не мають жодної позначки, то вони викреслюються.

	xyz	$xy\bar{z}$	$x\bar{y}z$	$\bar{x}yz$	$\bar{x}\bar{y}z$	$\bar{x}y\bar{z}$
xy	∨	∨				
$y\bar{z}$		∨	∨			∨
xz	∨		∨			
$\bar{y}z$			∨	∨		
$\bar{x}\bar{y}$				∨	∨	
$\bar{x}\bar{z}$					∨	∨

5. Визначити ядро перемикальної функції (сукупність імплікант, які не можна вилучити із СДНФ) та всі тупикові ДНФ. Із таблиці обрати таку сукупність простих імплікант, що містить позначки в усіх стовпцях. При декількох можливих варіантах такого вибору надається перевага варіанту покриття з мінімальним сумарним числом букв в імплікантах, що створюють покриття. Тупикові форми заданої функції складатимуться з суми суттєвих імплікант і простих імплікант, які покривають конституенти, що залишились. Із числа отриманих ТДНФ обрати мінімальну ДНФ.

$$f = xy \bigvee \bar{x}\bar{z} \bigvee \bar{y}z = xz \bigvee y\bar{z} \bigvee \bar{x}\bar{y}$$

5. Знаходження покриття ф-цій методом Петрика

Використовується для знаходження можливих покриттів перемикальної фун. З табл.. покриття , яка одержана будь-яким методом знаходження СДНФ. Суть методу – табл. покриття подають у вигляді логічного виразу , спрощення якого призводить до визначення всіх можливих множин імпліканту , що покривають фунцію. Реалізація методу :

- Прості імпліканти , що входять до СДНФ , позначаються довільними символами
- Для і-го стовпця таблиці будується умова покриття відповідної конституанти у вигляді диз'юнкції всіх імплікантів , що покривають цю конституанту
- Записується умова покриттят всіх всієї функ. Яка представляє собою кон'юнкцію одержаних в 2-ому пункті диз'юнкцій
- Одержаний лог. вираз приводиться до бездужкової функ. Після спрощення виразу та виконання всіх можливих поглинань формується диз'юнкція конюкцій . Кожна кон'юнкція в одержаній формі віповідає

множині простих імплікант , що покривають функцію(ТДНФ).Потім обирають МДНФ

Функция покрытия строится следующим образом. Соединяются знаком дизъюнкции и заключаются в скобки имена (символы) тех простых импликант, которые соответствуют строкам, содержащим метки в рассматриваемом столбце. Это выполняется для каждого невычеркнутого столбца, затем все скобки соединяются знаком конъюнкции, поскольку рассматриваемые простые импликанты покрывают в своїй совокупности оставшиеся минтермы функции. Полученная функция преобразуетсяпутем использования закона поглощения булевой алгебры и записывается в виде ДНФ. Каждый ее терм соответствует неизбыточному набору импликант, покрывающему оставшиеся минтермы функции. Совокупность импликант ядра и каждого из полученных наборов соответствует одной тупиковой ДНФ функции.

(D v E)(A v E)(A v C)(B v C) = (DA v DE v AE v E) (AB v AC v CB v C) = (DA v E) (AB v C) = DAB v DAC v EAB v EC

Тупиковая ДНФ1: $y = \overline{x_1}x_4 \vee \overline{x_2}x_3 \vee \overline{x_1x_2} \vee x_1\overline{x_3x_4} \vee x_2\overline{x_3}x_4$ (FGDAB)

ТупиковаяДНФ2: $y = \overline{x_1}x_4 \vee \overline{x_2}x_3 \vee \overline{x_1x_2} \vee x_1\overline{x_3x_4} \vee x_1x_2\overline{x_3}$ (FGDAC)

ТупиковаяДНФ3: $y = \overline{x_1}x_4 \vee \overline{x_2}x_3 \vee \overline{x_2x_4} \vee x_1\overline{x_3x_4} \vee x_2\overline{x_3}x_4$ (FGEAB)

ТупиковаяДНФ4: $y = \overline{x_1}x_4 \vee \overline{x_2}x_3 \vee \overline{x_2x_4} \vee x_1x_2\overline{x_3}$ (FGEC)

МДНФ – тупиковаяДНФ4 (онаже – икратчайшаяДНФ). МДНФ может быть упрощена, например, в тупиковой ДНФ4 можно вынести за скобки x2 , тогда получим скобочную форму записи функции:

$$y = \overline{x_1}x_4 \vee \overline{x_2}(x_3 \vee \overline{x_4}) \vee x_1x_2\overline{x_3}.$$

Скобочная форма является более минимальной, чем МДНФ, и выходит из класса нормальных форм. Ее реализация приводит к более простым схемам. Действительно минимальную форму не всегда можно получить из МДНФ.

6. Мінімізація систем булевих функцій

Розглянемо два підходи до мінімізації систем булевих функцій.

- Мінімізація кожної функції окремо.
- Спільна мінімізація функцій.

Мінімізація картами Карно. Для кожної функції будуємо карту для кожної функцій окремо. В результаті отримаємо мінімізовані функції. (***я втирать не буду сильно, ізі же.***)

X_1	X_2	X_3	f_1	f_2	f_3
0	**0**	**0**	**1**	**0**	**0**
0	**0**	**1**	**1**	**0**	**1**
0	**1**	**0**	**1**	**0**	**0**
0	**1**	**1**	**X**	**0**	**X**
1	**0**	**0**	**0**	**0**	**X**
1	**0**	**1**	**X**	**1**	**1**
1	**1**	**0**	**X**	**0**	**0**
1	**1**	**1**	**0**	**X**	**1**

X_1 \ X_2X_3	00	01	11	10
0	1	1	X	1
1	0	X	0	X

X_1 \ X_2X_3	00	01	11	10
0	0	0	0	0
1	0	1	X	0

X_1 \ X_2X_3	00	01	11	10
0	0	1	X	0
1	X	1	1	0

f1=$\overline{x1}$

f2 = $x1x3$

f3= x3

Мінімізація методом Квайна Мак-Класкі.

Вихідною формою для мінімізації систем булевих функцій методом Квайна Мак-Класкі є ДДНФ системи перемикальних функцій, для отримання якої необхідно подати в ДДНФ кожну функцію. При цьому кожній конституенті одиниці приписується множина міток, що визначають її приналежність до певної функції.

Етапи мінімізації системи перемикальних функцій:

- Записати ДДНФ системи перемикальних функцій;
- Виконати склеювання термів
- Виконати всі можливі поглинання
- Скласти таблицю покриття
- Вибрати покриття для кожної функції

K0	K1	K2
~~000 (f1)~~	~~0X0 (f1)~~	0XX (f1)
~~001 (f1,f3)~~	0X1 (~~f1~~,f3)	
~~010 (f1)~~	10X (f3)	
~~011 (f1,f3)~~	1X1 (f2,f3)	
~~100 (f3)~~	X01 (f1,f3)	
~~101 (f1,f2,f3)~~	X10(f1)	
~~110 (f1)~~		
~~111 (f2,f3)~~		

Імпл.	f1			f2	f3		
	000	001	010	101	001	101	111
0X1 (f1,f3)		X			X		
10X (f3)						X	
1X1 (f2,f3)				X			X
X01 (f1,f3)		X			X	X	
X10 (f1)			X				
0XX (f1)	X	X	X				
XX1 (f3)					X	X	X

$f1 = \overline{x1}$

$f2 = x1x3$

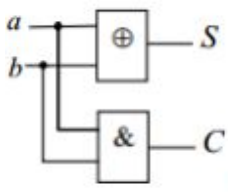
f3= x3

7. Суматор. Схема, принцип дії, застосування.

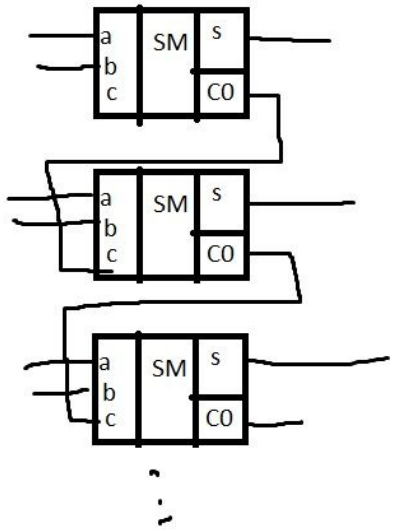
Суматор це пристрій для виконання арифметичної операції додавання і віднімання як двійкових так і десяткових чисел, а також використовується для побудови цифрових пристроїв для більш складних арифметичних операцій. Суматор має: а) n-входів, розрядів числа А; б) n-входів, розрядів числа В; в)вхід переносу; г) n-виходів суми; д) число бітів переносу;

a	*b*	*S*	*C*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Таблица истинности для полусумматора S = abVba = a⊕b, C = ab.



представлена схема полусумматора.

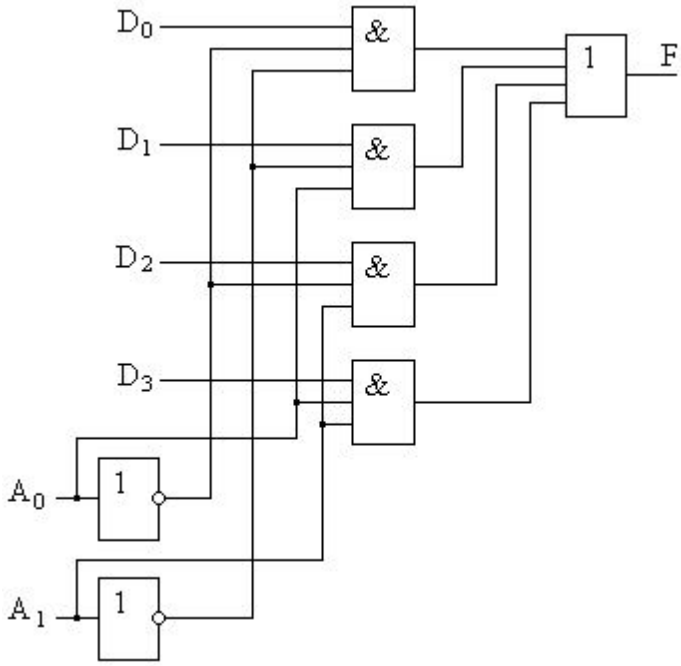


Суматори класифікують за такими ознаками:

- способом додавання — паралельні, послідовні та паралельно-послідовні;
- кількістю вхідних клем — напівсуматори, однорозрядні або багаторозрядні суматори;
- організацією зберігання результату додавання — комбінаційні, накопичувальні, комбіновані;
- системою числення — позиційні (двійкові, двійково-десяткові, трійкові) та непозиційні, наприклад, у системі залишкових класів;
- розрядністю (довжиною) операндів — 8-, 16-, 32-, 64-розрядні;
- способом подання від'ємних чисел — в оберненому або доповнювальному кодах, а також їх модифікаціях;
- часом додавання — синхронні та асинхронні.

8. Мультиплексор. Схема, принцип дії, застосування.

Комбінаційний вузол, що здатний передавати інформацію з декількох вузлів на один вихід. С помощью мультиплексора осуществляется временное разделение информации, поступающей по разным каналам. Мультиплексоры имеют две группы входов и один, реже два - взаимодополняющих выхода.



A - адресні входи, D - інформаційні входи
Таблиця істинності

F = !A1 !A2 D0 v !A1 A2 D1 v A1 !A2 D2 v A1 A2 D3

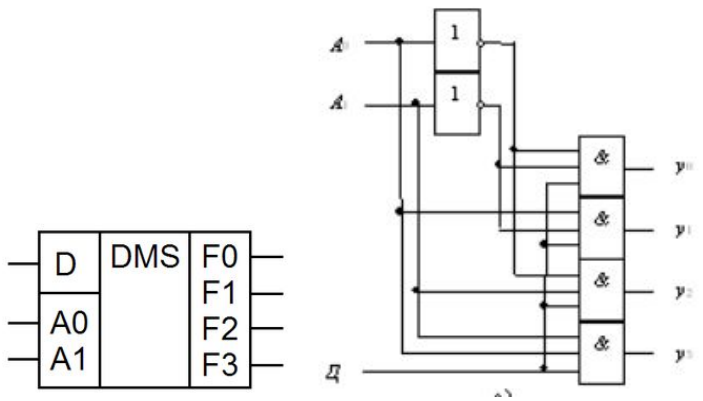
A 1	A 2	D0	D1	D2	D3	F
0	0	1	X	X	X	1
0	0	0	X	X	X	0
0	1	X	1	X	X	1
0	1	X	0	X	X	0
1	0	X	X	1	X	1
1	0	X	X	0	X	0
1	1	X	X	X	1	1
1	1	X	X	X	0	0
X	X	X	X	X	X	0

Еще есть схема с разрещаюзим сигналом. В конце добавить F ^ V

Мультиплексоры могут использоваться в делителях частоты, триггерных устройствах, сдвигающих устройствах и др. Мультиплексоры могут использоваться для преобразования параллельного двоичного кода в последовательный. Для такого преобразования достаточно подать на информационные входы мультиплексора параллельный двоичный код, а сигналы на адресные входы подавать в такой последовательности, чтобы к выходу поочередно подключались входы, начиная с первого и заканчивая последним.

9. Демультиплексор. Схема, принцип дії, застосування.

Демультиплексор - типова комбінаційна схема, що підключає інформаційний вхід до одного з декількох інформаційних виходів відповідно до сигналів, що подаються на керуючі входи.



Д	A_0 A_1	y_0 y_1 y_2 y_3	Д	A_0 A_1	y_0 y_1 y_2 y_3
0	0 0	0 0 0 0	0	1 0	0 0 0 0
1	0 0	1 0 0 0	1	1 0	0 0 1 0
0	0 1	0 0 0 0	0	1 1	0 0 0 0
1	0 1	0 1 0 0	1	1 1	0 0 0 1

10. Шифратор (кодер). Схема, принцип дії, застосування.

Шифратор - типова комбінаційна схема, призначена для перетворення просторового унітарного коду в двійковий код з природним порядком вагів. Виконує функцію протилежну дешифратору.

Принцип роботи шифратора зображений в таблиці істиності (розглядається шифратор на 4 входи та 2 виходи)

Q0	Q1	Q2	Q3	D1	D2
1	0	0	0	0	0

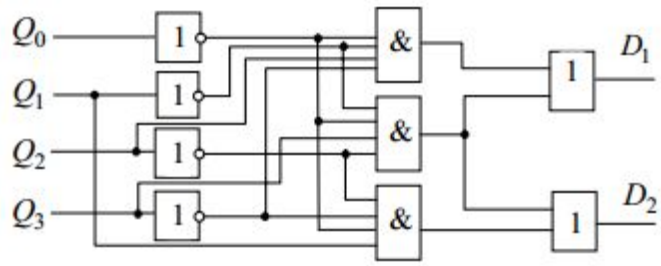
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

З таблиці істиності отримуємо функції збудження для D1 та D2:

$$D_1 = \overline{Q}_0\overline{Q}_1Q_2\overline{Q}_3 \vee \overline{Q}_0\overline{Q}_1\overline{Q}_2Q_3$$

$$D_2 = \overline{Q}_0Q_1\overline{Q}_2\overline{Q}_3 \vee \overline{Q}_0\overline{Q}_1\overline{Q}_2Q_3$$

Виходячи з цих рівняння можна скласти схему шифратора на логічних елементах:



Шифратори використовуються в всіх пристроях управління на двійковій логіці, які для зручності користувача мають десяткову клавіатуру. Приклад: калькулятор, домофон і т.д. Якщо до виходу схеми виділення старшої одиниці підключити шифратор, то вийде пріорітетний шифратор, який формує на виході номер самої старшої одиниці.

11. Дешифратор. Схема, принцип дії, застосування.

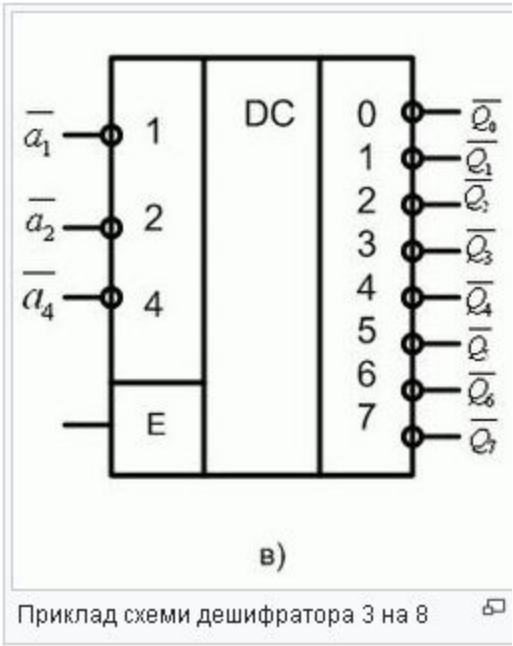
Дешифра́тор або **деко́дер** — логічний пристрій, який перетворює код числа, що поступило на вхід, в сигнал на одному з його виходів. Вихідними функціями дешифратора є різноманітні конституенти одиниці.

За принципом дії розрізняють такі види дешифраторів:

- Послідовні,
- Паралельні,
- Паралельно-послідовні.

Розрізняють дешифратори першого та другого роду:

1. Дешифратори першого роду реалізують систему функцій, кожна з яких приймає одиничне значення при відповідному одиничному значенні вхідного слова.
2. Дешифратори другого роду реалізують систему функцій, кожна з яких приймає одиничне значення при визначених діапазонах вхідного слова.

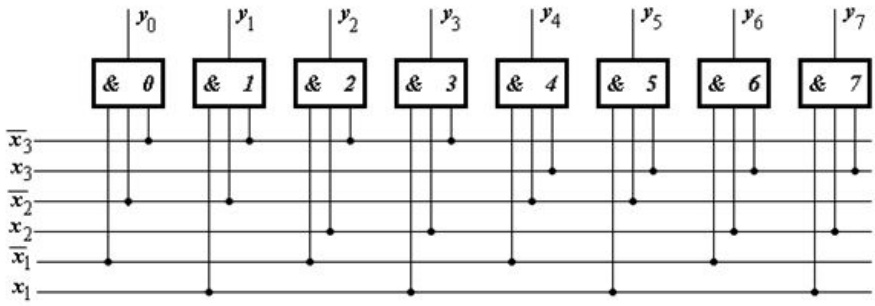


Приклад схеми дешифратора 3 на 8

Дешифратори використовуються для перетворення двійкових чисел в десяткові числа і знаходять застосування в друкувальних пристроях. У таких пристроях двійкове число, вступаючи на вхід дешифратора, викликає поява десяткового числа тільки на одному певному його виході.

Повний дешифратор– дешифратор, що має стільки виходів ***m***, скільки різних значень може мати ***n***-розрядне двійкове число на його входах, тобто $m = 2^n$.

На рис. 2 наведена схема одноступеневого лінійного дешифратора на три входи, у якого число виходів $m = 2^3 = 8$.



Якщо довжина двійкового слова, що дешифрується, більше можливого числа входів елементів ***І***(&), то використовують багатоступневу (каскадну) будову дешифратора.

В прямокутному (матричному) дешифраторі слово, що дешифрується, розділене на кілька підслів. Підслова дешифруються на окремих лінійних дешифраторах, утворюючи вихідні значення, які називають **частковими**. Ця група лінійних дешифраторів уявляє собою перший каскад прямокутного дешифратора. В будь-якому наступному каскаді виконується операція кон'юнкції часткових вихідних значень, утворених лінійними дешифраторами попереднього каскаду.

http://itcj.sethost.net/pdf/epivt_2_1_26.pdf

12. Програмуємі логічні схеми. Принцип дії, застосування.

Програмовані логічні матриці є типовою комбінаційною схемою і може реалізувати систему перемикальних функцій поданих у певному базисі

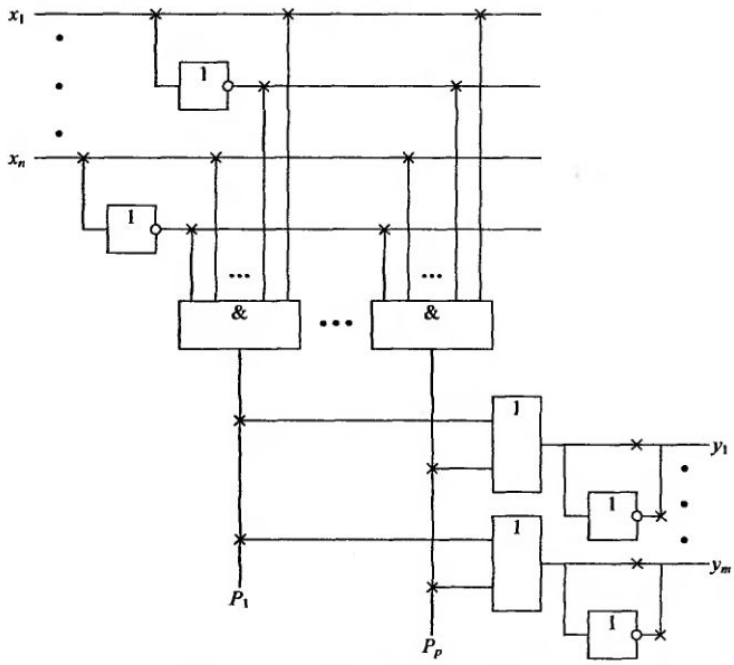


Рис. А1-4.16. Логічна структура ПЛМ

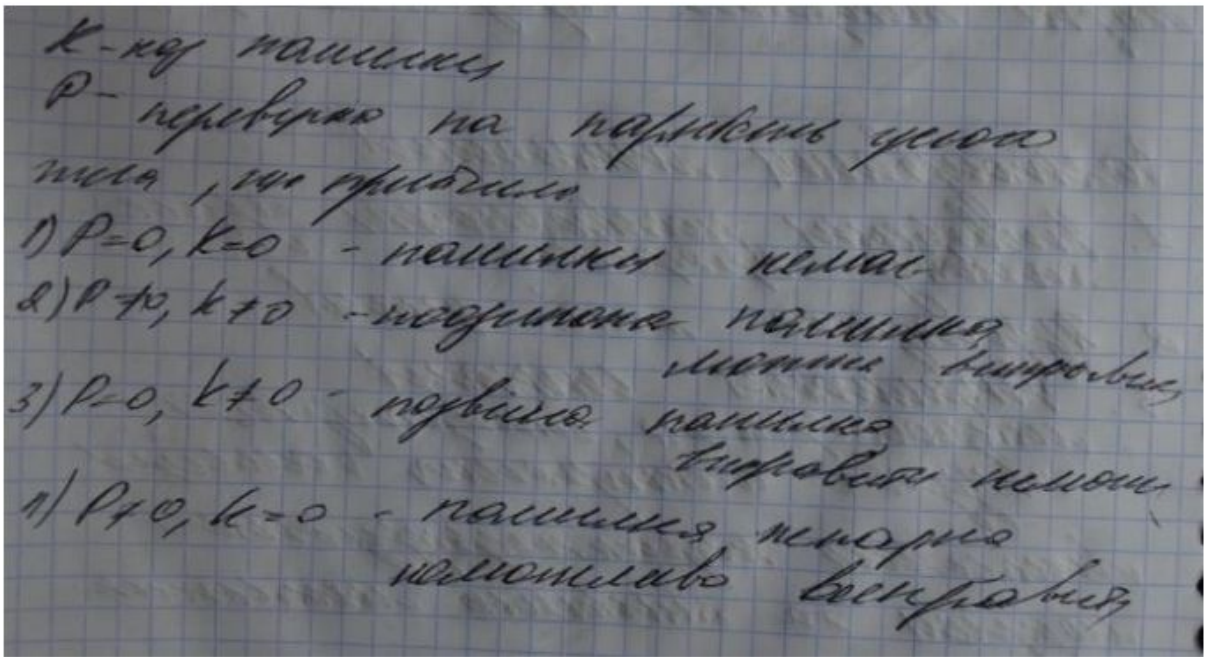
Точками відзначено зв'язки між шинами, випалюючи які ми отримуємо здатність реалізувати системи перемикальних функцій. Під час побудови комбінаційних схем на базі ПЛМ враховуються різні цільові функції проектування комбінаційних схем які зводиться до сумісної мінімізації системи перемикальних функцій

13. Контроль по Хєммінгу (https://habrahabr.ru/post/140611/)

Корректирующий код Хемминга позволяет исправить одиночную ошибку. Возможность исправления ошибки основана на повторении k раз контроля по четности, но не всего слова сразу, а к определенной группе его разрядов. Слово разбиение на группы так, чтобы номер любого разряда однозначно определялся по его принадлежности или непринадлежности к этим группам.Пусть в слове 11 разрядов, тогда для контроля нужно 4 разряда.Группы контроля по четности компонуют из разрядов кодового слова следующим образом: 1) Любой разряд кодового слова входит в состав стольких групп, столько раз, сколько единиц содержится в двоичном коде его номера (разряды 1,2,4,8 входят в состав 1 контрольными групп и сами являются контрольными разрядами; 7 входит в состав 3 контрольных групп по четности); 2) В любую i-ю контрольную

группу входят те разряды, в двоичном номере которые на i-ом номере стоит единица (2-ю контрольную группу образуют: 0,3,6,7,10,11,14,15).

Модифицированный код Хемминга способен исправлять единичную ошибку и обнаруживать двоичную. При этом добавляют еще один разряд для контроля по четности всего слова.



14. Асинхронні тригери. Схеми, принцип дії, застосування.

Особенность последовательностной схемы (в отличии от комбинационной) состоит в том, что значения на выходах схемы в текущий момент времени зависят в каком состоянии находилась схема в предыдущий момент времени. Триггер это последовательностная схема, элементарный автомат, содержащий элемент памяти (запоминающий элемент) и схему управления элементом памяти. На схему управления подаются входные сигналы (информационные) и сигналы обратной связи с выхода элемента памяти. В некоторых простейших триггерах схема управления может отсутствовать. Состояние выхода триггера определяется элементом памяти, сигналом на его прямом выходе Q. Обычно триггер имеет и инверсный выход $\overline{Q}$, иногда он обозначается Q*.

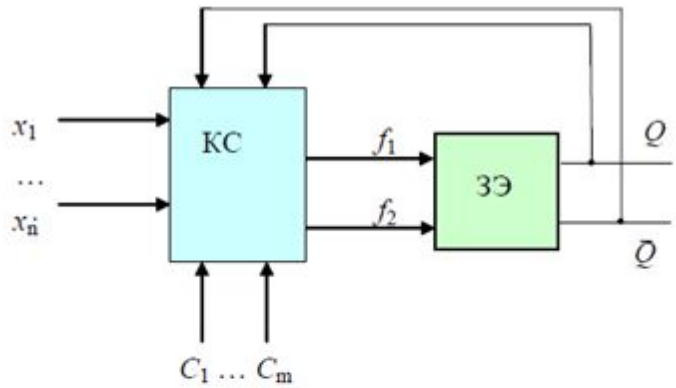
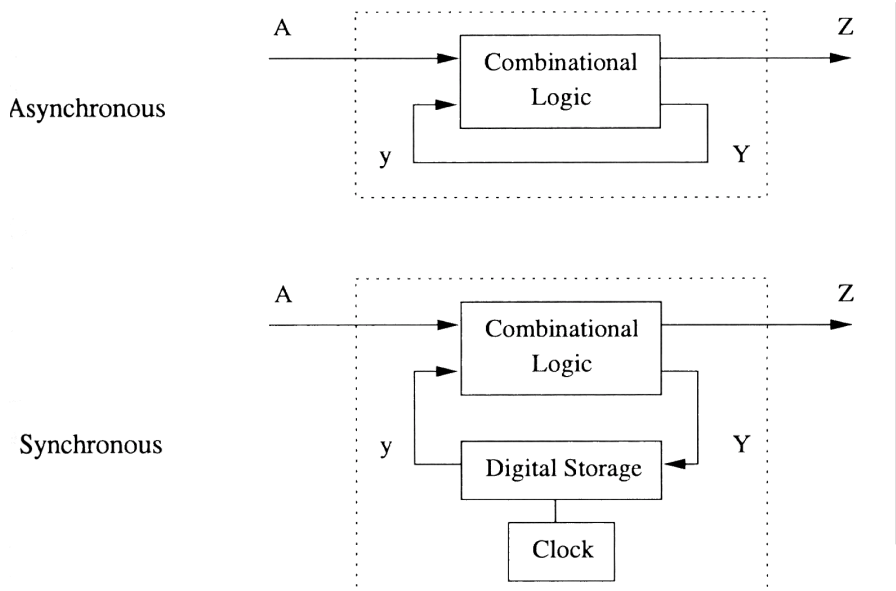


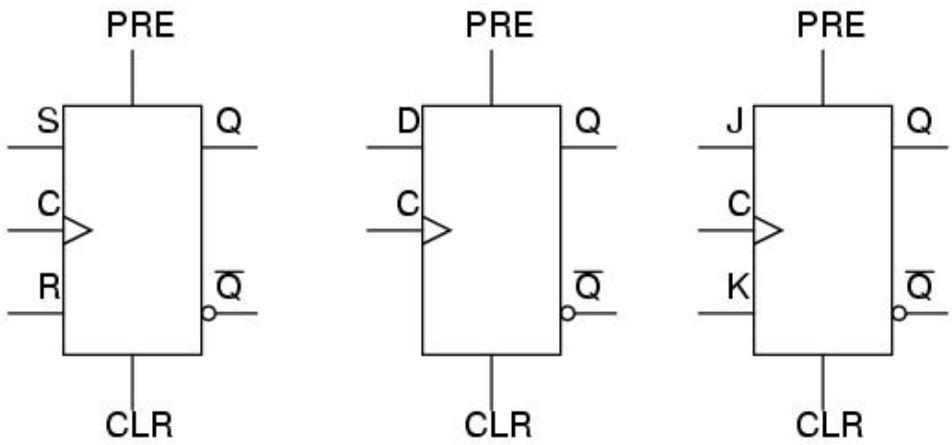
Рисунок 3.1– Структурная схема триггера

Классификация триггеров проводится по закону логического функционирования (триггеры типа RS, R*S*, JK, J*K* и другие), по способу записи информации в триггер (асинхронные и синхронные), по способу восприятия триггером тактовых сигналов (управляемые уровнями и управляемые фронтами), по структуре (одноступенчатые и двухступенчатые). Синхронные триггеры – триггеры, у которых переход в новое состояние вызывается не только изменениями информационных сигналов, но и синхросигнала. Синхросигнал (тактовый сигнал) дважды в течение такта меняет свое значение.

Асинхронные триггеры – триггеры, у которых переход в новое состояние вызывается изменениями информационных входных сигналов. Т.е. без тактовых или синхронизирующих сигналов.



Нормальные входы данных на тригеры (D, SR, или JK), называются синхронными входами, поскольку они имеют влияние на выходы (Q и не-Q) только с тактовым сигналом. Дополнительные входы, которые асинхронными,могут установить или сбросить триггер независимо от состояния тактового сигнала. Как правило, они называются PRE и CLR:



Когда PRE вход активирован, то триггер будет установлен (Q = 1, не-Q = 0), независимо от значения синхронных входов. Когда CLR вход активирован, триггер будет сброшен (Q = 0, а не-Q = 1). Если оба PRE & CLR входы активируются, мы получаем недопустимое состояние на выходе, где Q и не-Q одном состоянии, так же, как и RS триггер. PRE и CLR входы используются, когда несколько триггеров собираются вместе,

чтобы выполнить функцию мульти-разрядного двоичного слова, и одна строка необходима, чтобы установить или переустановить их все сразу.

15. Синхронні тригери. Схеми, принцип дії, застосування.

Синхронные триггеры реагируют на информационные сигналы только при наличии соответствующего сигнала на так называемом входе синхронизации С (от англ. clock). Этот вход также обозначают термином «такт». Такие информационные сигналы называют синхронными. Синхронные триггеры в свою очередь подразделяют на триггеры со статическим и с динамическим управлением по входу синхронизации С.

Триггеры со статическим управлением воспринимают информационные сигналы при подаче на вход С логической единицы (прямой вход) или логического нуля (инверсный вход).

Триггеры с динамическим управлением(по фронту) воспринимают информационные сигналы при изменении (перепаде) сигнала на входе С от 0 к 1 (прямой динамический С-вход) или от 1 к 0 (инверсный динамический С-вход).

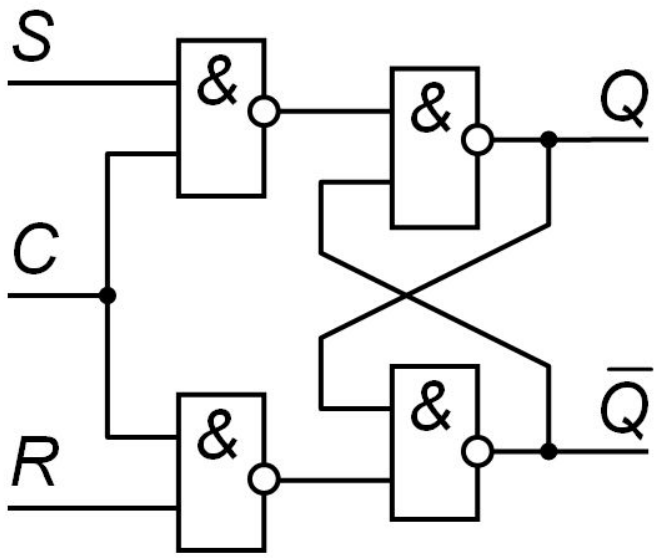


Схема синхронного RS-триггера на элементах 2И-НЕ.

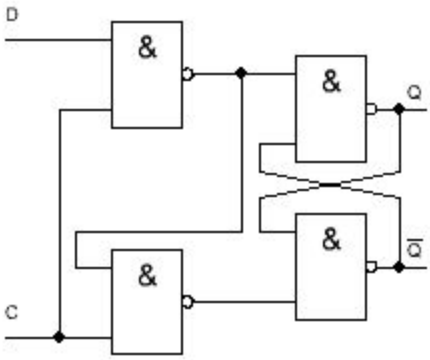
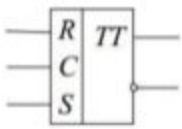


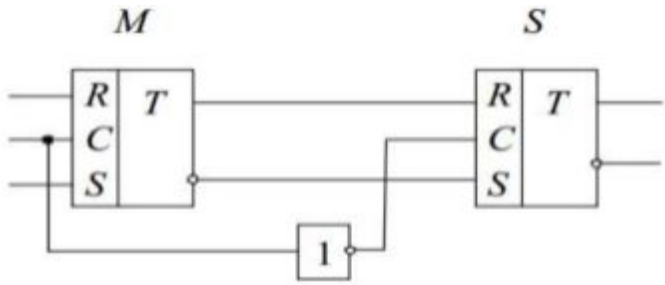
Схема синхронного D-триггера на элементах 2И-НЕ.(здесь есть логические гонки).
Можна використовувати всюди де потрібно контролювати запис даних.

16. Одноступінчаті та двоступінчаті тригери (Master-Slave). Схеми, принцип дії, застосування.(Leo)

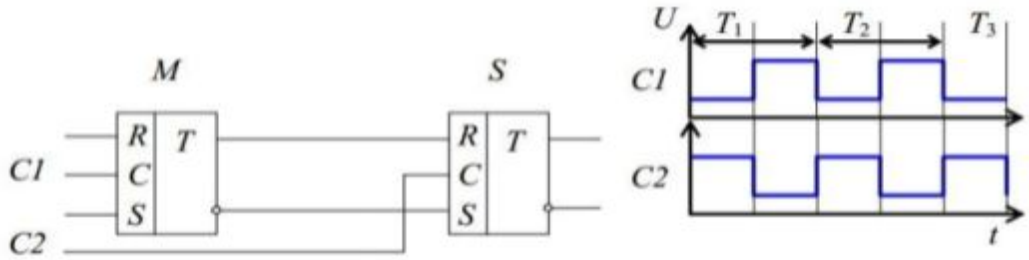
Одноступінчаті тригери - складаються з одного елемента пам'яті і мають як правило, статичне керуванням. Двоступінчаті тригери мають два елементи пам'яті, з'єднані послідовно. Запис інформації в них виконується послідовно. Така послідовність тригерів називається MS- структурою (Master-Slave (Майстер-Помічник)) або просто MS-тригерами. У умовних позначеннях MS-тригери позначаються двома буквами ТТ.



Функціональні властивості схеми задаються першим тригером, а другий тригер, як правило, звичайний синхронний RS-тригер. Двоступеневі тригери можуть керувати одним або двома синхроімпульсами. При статичному управлінні вхід С тригера М з'єднується з входом С тригера S через інвертором у випадку керування одним синхросигналом.



Коли С = 1, виконується запис інформації в М тригер (якщо синхронізація здійснюється по рівню 1), а в S тригері зберігається стара інформація , оскільки на його синхровхід в даному випадку завдяки інвертору подається 0. Коли С = 0 , тригер М зберігає свій попередній стан , а на тригер S переписується інформація з тригера М , оскільки на синхровхід тригера S в даному випадку завдяки інвертору подається 1. У разі використання двох синхросигналів для управління тригерами в MS- структурі схема і синхросиг можуть виглядати, як показано на рис.

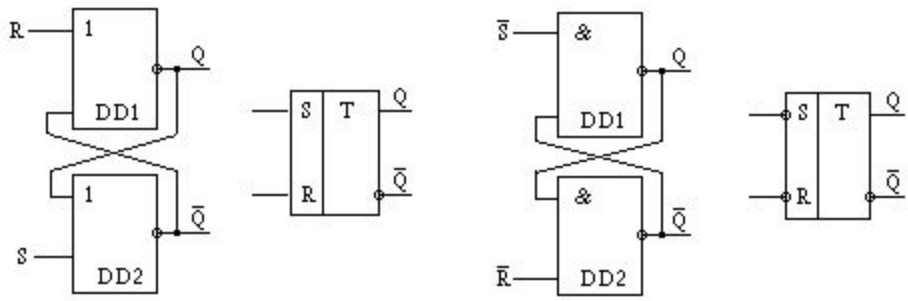


Зазвичай двоступеневі тригери застосовуються в схемах, де логічні функції входів тригера залежать від його виходів, щоб уникнути часових гонок сигналів.

17. RS-тригери. Схеми, принцип дії, застосування.

RS-тригер - це тригер з роздільним установкою станів логічного нуля і одиниці (з роздільним запуском). Він має два інформаційних входи S і R. По входу S тригер встановлюється в стан Q = l (/ Q = 0), а по входу R - в стан Q = O (/ Q = 1).

Асинхронні RS-тригери. Вони є найбільш простими тригерами. В якості самостійного пристрою застосовуються рідко, але є основою для побудови більш складних тригерів. Залежно від логічної структури розрізняють RS-тригери з прямими і інверсними входами.



Стани тригерів під впливом певної комбінації вхідних сигналів

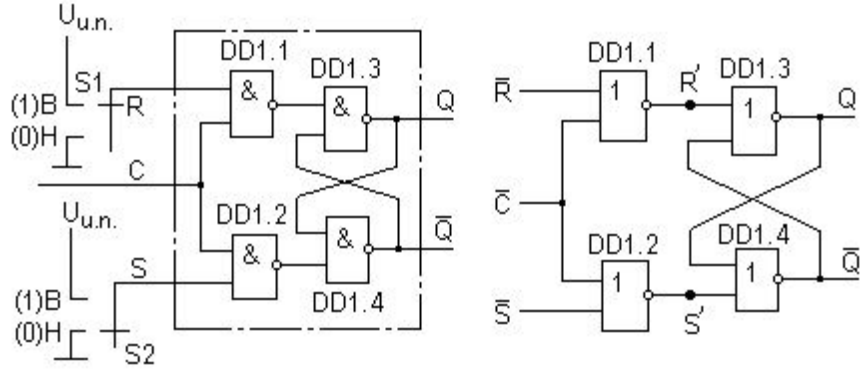
Входы		**Выходы**			
S	**R**	**И-НЕ**		**ИЛИ-НЕ**	
		Qn+1	**/Qn+1**	**Qn+1**	**/Qn+1**
0	0	X		Qn	/Qn
1	0	0	1	1	0
0	1	1	0	0	1
1	1	Qn	/Qn	X	

Якщо зі входу S зняти одиничний сигнал, т. Е. Встановити на вході S нульовий сигнал, то стан тригера не зміниться. Режим S = 0, R = 0 називають режимом зберігання інформації, тому що інформація на виході залишається незмінною.

При подачі вхідних сигналів S =
0, R = 1 відбудеться перемикання тригера, а на виході буде Qт + 1 = 0 (/ Qn + 1 = 1). Такий режим називають режимом запису логічного нуля (режим скидання). При S = R = 1 стан тригера буде невизначеним, тому що під час дії інформаційних сигналів логічні рівні на виході тригера однакові (Qn + 1 = / Qn + 1 = 0), а після закінчення їх дії тригер може рівноймовірно прийняти будь-яке з двох стійких станів. Тому така комбінація S = R = 1 є забороненою.

Для тригера з інверсними входами режим запису логічної одиниці реалізується при / S = 0, / R = 1, режим запису логічного нуля - при / S = 1, / R = 0. При / S = 1 R = 1 забезпечується зберігання інформації. Комбінація вхідних сигналів / S = 0 R = 0 є забороненою.

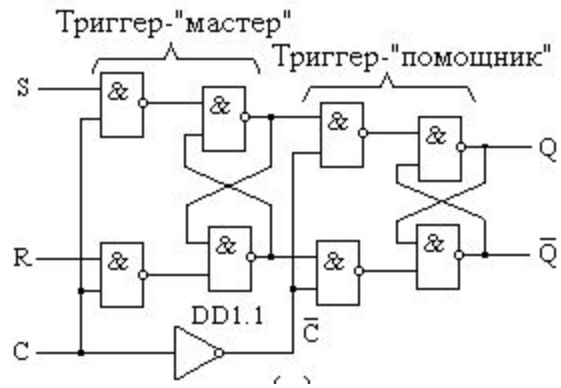
Синхронні RS-тригери. Тригерні осередки - це основа подільників частоти, лічильників і регістрів. У цих пристроях записану раніше інформацію за спеціальним сигналом, званому тактовим, слід передати на вихід і переписати в наступну комірку. Для здійснення такого режиму в RS-тригер необхідно ввести додатковий вхід С, який може бути статичним або динамічним, т. Е. Отримаємо синхронний RS-тригер.



Елементи DD1.1 і DD1.2 утворюють схему управління, а елементи DD1.3 і DD1.4 - асинхронний RS-тригер. Іноді такий тригер називають RST-тригером (якщо вхід С вважати тактовим входом Т).

Тригер має прямі статичні входи, тому керуючим сигналом є рівень логічної одиниці.

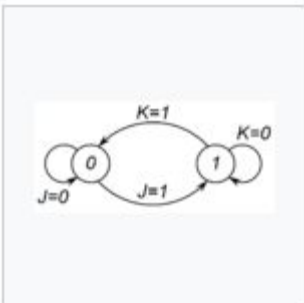
Якщо на вхід З подати сигнал логічної одиниці С = 1, то робота тригера аналогічна роботі найпростішого асинхронного RS-тригера. При С = 0 входи S і R не роблять вплив на стан тригера. Комбінація сигналів S = R = C = 1 є забороненою.
Синхронний двоступеневий RS-тригер (master-slave) складається з двох синхронних RS-тригерів і інвертора. Входи з обох тригерів з'єднані між собою через інвертор DD1.1. Якщо С = 1, то перший тригер функціонує відповідно до сигналам на його входах S і R. Другий тригер функціонувати не-може, т. До, у нього С = 0. Якщо С = 0, то перший тригер не функціонує, а для другого тригера С = 1, і він змінює свій стан згідно сигналам на виходах першоготригера.



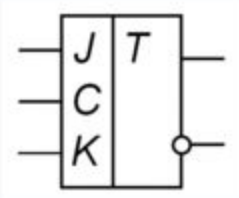
18. JK-тригери. Схеми, принцип дії, застосування. [Misha Smaga]

JK-тригер з двома стійкими станами і двома інформаційними входами. Вхід J (Jark) використовується для установки тригера в «1», вхід K (Kill) для установки в «0». Одночасна подача двух «1» на вхіди J і K призводить до інвертування значення, чим цей тригер і відрізняється від RS-тригера. При подачі «0» на обидва входи тригер зберігає свій стан. На практиці застосовуються лише синхронні JK-тригери, тобто стану основних входів J і K враховуються лише в момент тактування, наприклад по позитивному фронту імпульсу на вході синхронізації.

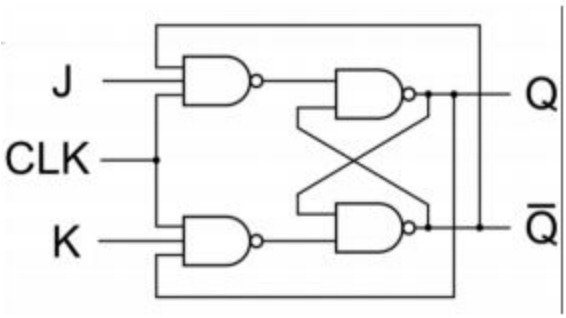
J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



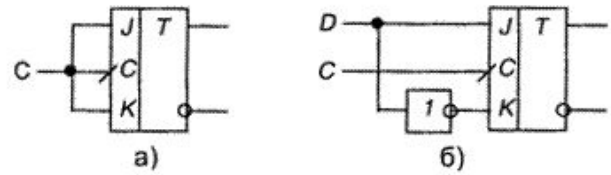
Граф переходів JK-тригера



Умовне графічне позначення JK-тригера зі статичним входом З



На базі JK-тригера можливо побудувати D-тригер або T-тригер. Як можна бачити в таблиці істинності JK-тригера, він переходить в інверсний стан щоразу при одночасній подачі на входи J і K логічної “1”. Ця властивість дозволяє створити на базі JK-тригера T-тригер, об'єднавши входи J і K. На малюнку а) зображена схема T-тригера, а на малюнку б) зображена схема D-тригера.



Алгоритм функціонування JK-тригера можна представити формулою

$$Q(t+1) = \overline{Q}(t) \cdot J + Q(t) \cdot \overline{K}$$

19. D-тригери. Схеми, принцип дії, застосування.

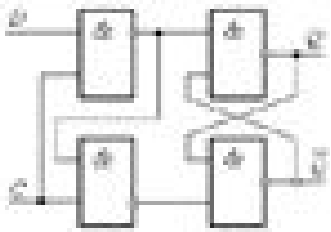
Триггером типа D (Delay - задержка) называется триггер с двумя устойчивыми состояниями равновесия и одним информационным входом D. Значения, поступающие на вход D, записываются на выход Q, т.е. триггер работает как повторитель.

Позначення асинхронного тригера, повна та скорочена матриця переходів:

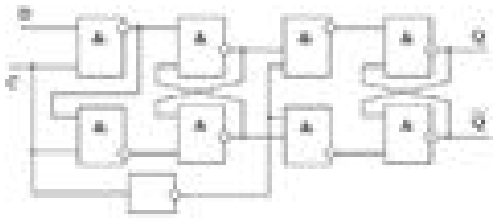
D	Q(t+1)
0	0
1	1

Q(t)-Q(t+1)	D
0-0	0
0-1	1
1-0	0
1-1	1

Одноступінчатий. Всі зміни на вході D одразу передаються на вихід.



Двоступінчатий(статичний Д тригер)

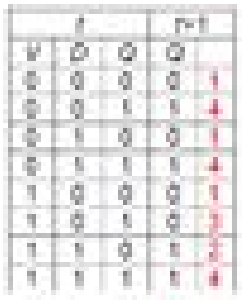


спершу інформація записується в першу ступінь, потім тригер першої ступіні переходить до режиму зберігання, потім інформація переписується до другої ступіні та з’являється на виході.

Часто *D*-триггер используется с дополнительным разрешающим входом *V*. В таком исполнении он называется *VD*-триггером. Когда на вход *V* подается сигнал «0», триггер хранит свое внутреннее состояние, когда на вход *V* подается сигнал «1», тригер работает как *D*-триггер.

Скорочена та повна таблиці переходів

V	D	Q(t+1)
0	0	Q(t)
0	1	Q(t)
1	0	0
1	1	1



Синхронні тригери окрім основних входів мають також вхід С, що служить сигналом синхронізації:

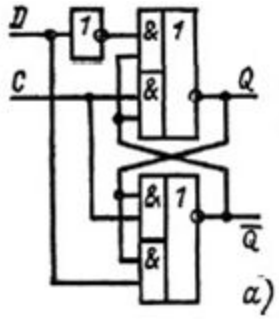
D-тригер в основному використовується для реалізації засувки. Так, наприклад, для зняття 32 біт інформації з паралельної шини, беруть 32 D-тригера і об'єднують їх входи синхронізації для управління записом інформації в засувку, а 32 D входу під'єднують до шини.(слово в слово не катайте).

20. Т-тригери. Схеми, принцип дії, застосування.

Т-триггер работает следующим образом: если на вход Т подается сигнал «0», триггер хранит свое внутреннее состояние, если на вход Т подается сигнал «1», триггер инвертирует свое внутреннее состояние. Но для установки триггера в начальное состояние одного Т-входа недостаточно. Т- триггер реализуется с установочным входом R (входом асинхронной

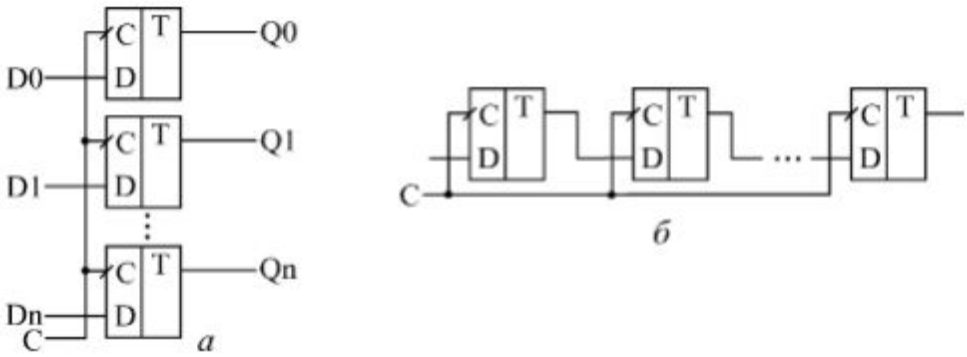
R	*T*	*Q*(*t*+1)
0	0	*Q*(*t*)
0	1	$\overline{Q}(t)$
1	0	0
1	1	0

установки). В таком исполнении он называется RT-триггером (или просто Т- триггером). Когда на вход R подается сигнал «1», триггер устанавливается в ноль, когда на вход R подается сигнал «0», триггер выполняет свою обычную функцию



21. Регістри (послідовні, паралельні). Схеми, принцип дії, застосування.

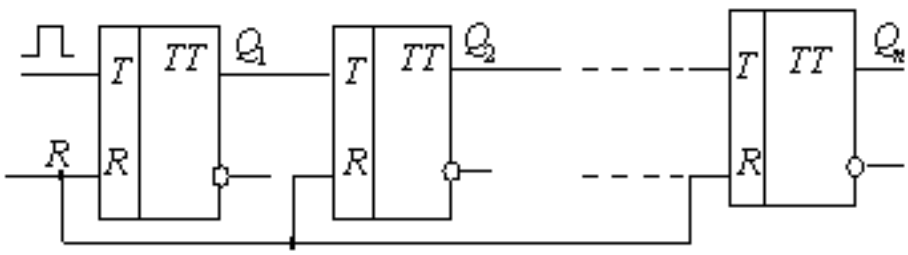
Регістр це упорядкована послідовність тригерів, призначення для збереження слів і виконання мікрооперацій над ними. Мікрооперації це елементарні дії що виконують регістри за один такт (встановлення вхідного сигналу, запис слова, зсув). Число розрядів у регістрі наз. його довжиною, у n-розрядному регістрі може бути записано 2^n слів.



а)паралельний регістр; б)послідовний

22. Лічильники (прямі, обернені). Схеми, принцип дії, застосування.

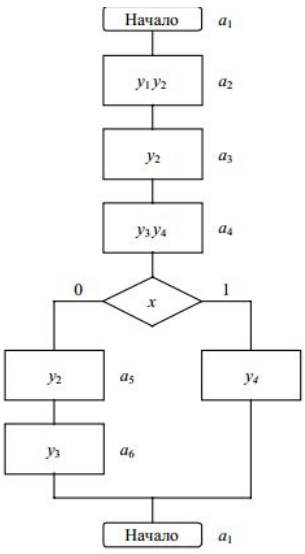
Лічильником називається типовий функціональний вузол комп'ютера, призначений для лічби вхідних імпульсів. Лічильник являє собою зв’язаний ланцюг Т-тригерів, які утворюють пам’ять із заданим числом сталих станів



За цільовим функціональним призначенням лічильники поділяють на два типи: *простi* (підсумовувальні та віднімальні) та *реверсивні*. У *підсумовувальному* лічильнику при подачі на вхід імпульсу код числа, що зберігається у лічильнику, *зростає на одиницю*, а у *віднімальному* - *зменшується на одиницю*. Отже, підсумовувальний, лічильник виконує прямий, а віднімальний - обернений підрахунок числа одиниць, що надійшли на його вхід. *Реверсивний лічильник може працювати в режимі прямого та оберненого підрахунку.*

У зв'язку з тим, що у кожному розряді лічильника, наприклад підсумовувального, виконується операція додавання двійкових чисел, у результаті якої утворюється перенос у старший розряд, лічильники ще розрізняють за способом утворення сигналів переносу. Це, зокрема, лічильники з послідовним, паралельним і послідовно-паралельним переносом. Їх часто називають просто послідовні,паралельні та послідовно-паралельні лічильники. Вони відрізняються способами подачі вхідних тактових імпульсів на входи розрядів. У послідовному лічильнику вхідні імпульси подаються тільки на вхід першого тригера, а у паралельному - одночасно на синхровходи тригерів у всіх розрядах. Різновидом паралельних лічильників є кільцеві лічильники, що будуються на базі регістрів зсуву. Послідовно-паралельний лічильник будують за принципом послідовного з'єднання (каскадування) кількох паралельних лічильників.

23. Синтез керуючого автомата Мура на базі регістра зсуву.



Цей метод дозволяє відмовитись від використання дешифратора при побудові автомату, оскільки стани кодуються унітарним кодом (00001, 00010,00100…). При такому підході в кожен момент часу тільки один тригер знаходиться в стані 1.

Проведемо синтез автомату, ГСА якого зображнено зліва. Згідно з ГСА наш автомат має 7 станів (використовуємо 7 D тригерів). Кодуємо стани унітарним кодом: a1 =1000000, a2=0100000,..., a7 =0000001.

Далі будуємо зворотню таблицю переходів-виходів для автомату.

Згідно до таблиці виписуємо функції збудження: D1 = a6 + a7 D2 = a1 D3 = a2 D4 = a3 D5 = a4 · x D6 = a5 D7 = a4 · x

Також виписуємо вихідні сигнали y: y1 = a2 y2 = a2 + a3 + a5 y3 = a4 + a6 y4 = a4 + a7

a	Ka_m	$a_s(y)$	Ka_s	x	$ФВ$
a_6	0000010	$a_1(-)$	1000000	1	D_1
a_7	0000001			1	D_1
a_1	1000000	$a_2(y_1 y_2)$	0100000	1	D_2
a_2	0100000	$a_3(y_2)$	0010000	1	D_3
a_3	0010000	$a_4(y_3 y_4)$	0001000	1	D_4
a_4	0001000	$a_5(y_2)$	0000100	$\overline{x}$	D_5
a_5	0000100	$a_6(y_3)$	0000010	1	D_6
a_4	0001000	$a_7(y_4)$	0000001	x	D_7

Оскільки стани автомата закодовані унітарним кодом, то можна ототожнити кожен стан з виходом відповідного тригера, тобто прийняти ai = Qi. Для прийнятого способу кодування перехід з одного стану в інший як би супроводжується зрушенням коду, записаного в семирозрядний регістр. Цим і пояснюється назва методу.

Далі будуємо схему керуючого автомату:

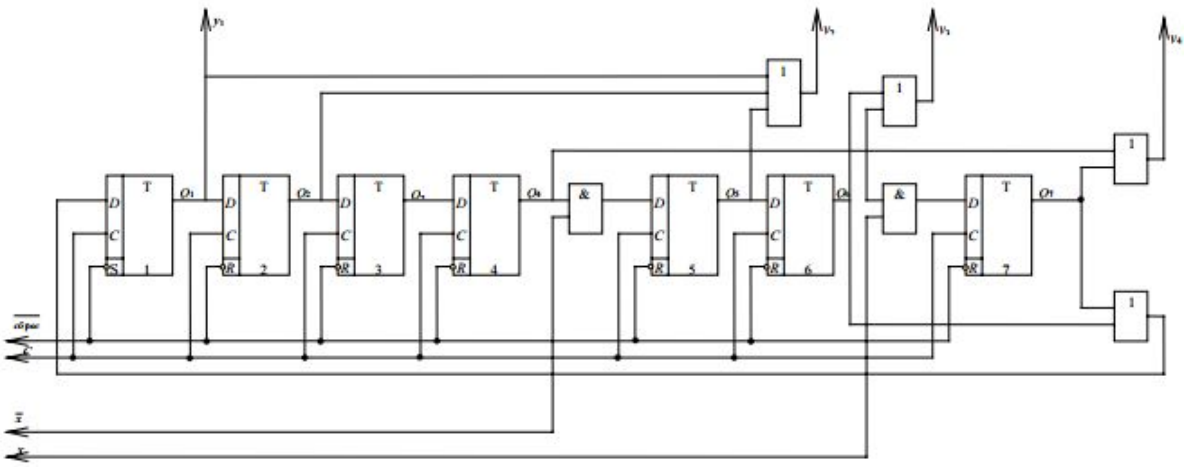


Рис. 62. Функциональная схема автомата Мура на базе регистра сдвига.

24. Структурний синтез автомата Мілі.

Керуючий автомат призначається для управління операційним автоматом і виконання на ньому певного мікроалгоритму. Цифровий автомат – це пристрій, який перетворює вхідну інформацію в вихідну по заданому. Автомат складається з пам’яті (певного набору тригерів) та комбінаційної схеми, що складається з елементарних логічних елементів і виробляє вихідні керуючі сигнали та функції збудження для тригерів.

Етапи проектування:

· Побудова змістовної схеми алгоритму.
· Побудова блок-схеми закодованого мікроалгоритму
· Побудова граф-схеми переходів автомата Мілі
· Побудова таблиці переходів-виходів автомата Мілі
· Кодування станів автомата
· Побудова структурної таблиці переходів-виходів автомата Мілі
· Визначення систем рівнянь переходів
· Визначення системи рівнянь виходів
· Побудова функціональної схеми автомата

На етапі отримання ГСА входи вершин відзначаються a0, a1… та такими правилами:

· Символом a0 позначають вхід вершини, наступної за початкової за початкової, також вхід кінцевої вершини
· Входи різних вершин, наступних за операторними, повинні бути відзначеними різними символами, але тільки один символ на вершину.

Число елементів пам’яті $m= \log_2 M$, де M – кількість станів

25. Структурний синтез автомата Мура.

При заданих типах елементів пам’яті структурний синтез керуючого автомата зводиться до наступних проектних операцій:

1. Кодування внутрішніх станів автомата
2. Побудова структурної таблиці переходів-виходів автомата Мура.
3. Визначення системи рівнянь переходів
4. Визначення системи рівнянь виходів
5. Побудова функціональної схеми автомата
6. Перевірка коректної роботи автомата

Кодування станів - встановлення взаємно-однозначної відповідності між станами автомата і комбінаціями станів елементів пам'яті. При двійковому кодуванні станів автомата число тригерів в його схемі дорівнює числу розрядів коду і обчислюється за формулою: $m = \log 2M\uparrow$, де M–кількість можливих станів Власне кодування станів проводиться різними методами залежно від типу елемента пам’яті (основними методами є алгоритм

кодування для D-тригерів, алгоритм сусіднього кодування та евристичний алгоритм).

Побудова структурної таблиці переходів:

У даній таблиці вказується код вихідного стану та стану переходу, сигнали, під дією яких відбувається перехід, та функція збудження ФЗ. В стовпці ФЗ вказується ті значення функцій збудження, які на даному переході обов'язково рівні 1. Решта (тобто рівні 0) не вказуються. Це еквівалентно тому, що всім невизначеним значенням функцій збудження приписується значення 0, що в загальному випадку не дає мінімальної функції, проте в реальних автоматах мінімізація зазвичай не робиться по причині її неефективності.

Формування системи рівнянь переходів:

Потрібно виписати систему рівнянь збуджень, по одному рівнянню на кожен з елементів пам’яті (тригерів). Вираз для кожної функції виходить у вигляді логічної суми додатків виду s_iX, де s_i - початковий стан, X- умова переходу. Для спрощення отриманих виразів виконуємо всі можливі операції склеювання і поглинання.

Формування системи рівнянь виходів:

Ця система складається на основі структурної таблиці переходів., де X – вхідні, Y – вихідні сигнали. Приклад:

$$\begin{cases} Y_1 = s_1 \\ Y_2 = Y_3 = s_2 \\ Y_4 = s_3 \\ Y_5 = s_4 \\ Y_6 = s_5 \\ Y_7 = Y_8 = s_6 \end{cases}$$

Побудова функціональної схеми автомата:Для побудови функціональної схеми автомата за отриманими виразами необхідно або замінити s_i його значеннями через значення сигналів – виходів тригерів або отримати сигнал, відповідний s_i. Зазвичай використовують другий спосіб і для отримання сигналу s_i застосовують так званий дешифратор станів, на вхід якого надходять сигнали з виходів елементів пам'яті (тригерів).

Перевірка коректної роботи автомату:

Для автомата Мура за значенням Qt і Xt спочатку визначається наступний стан автомата (Qt + 1) а потім за значенням Qt + 1 визначається значення вихідного сигналу (Yt + 1). Отримані значення Qt + 1 і Yt порівнюються з даними таблиці переходів і виходів. При збігу очікуваних і отриманих результатів робиться висновок про правильність синтезу автомата.

Для реальних автоматів з великою кількістю станів і входів виконання перевірки вручну стає скрутним. Для цієї мети в даний час використовуються спеціальні програми.

26. Компаратор. Схема, принцип дії, застосування.

Цифровим компаратором називається комбінаційний логічний пристрій, призначений для порівняння чисел, представлених у вигляді двійкових кодів. Число входів компаратора визначається розрядністю кодів, що порівнюються. На виході компаратора зазвичай формується три сигнали:

F=- рівність кодів;

F>- якщо числовий еквівалент першого коду більше ніж другого;

F<- якщо числовий еквівалент першого коду менше ніж другого.

Роботу компаратора при порівнянні двох однорозрядних кодів пояснює наступна таблиця істинності:

x0	x1	F==	F>=	F<=
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

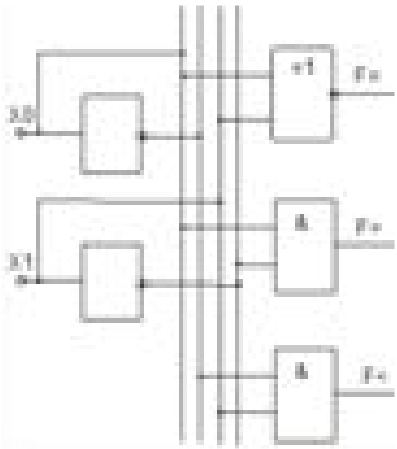
Система ФАЛ, що відповідає наведеній таблиці істинності, має вигляд:

F== nx1*nx0 + x1x0,

F>= nx1*x0,

F<= x1* nx0’

Отже, можна побудувати схему:



27. Перетворення з коду Грея до двійкового позиційного, і навпаки.

Код Грея — одна із систем кодування інформації, в якій два послідовні коди відрізняються значенням лише одного біта. Він мінімізує ефект помилок при перетворенні аналогових сигналів у цифрові

1) Перехід з двійкового коду в код Грея:

Коди Грея легко виходять з двійкових чисел шляхом побітовій операції «виключне АБО» з тим же числом, зрушеним вправо на один біт. Отже, і-й біт коду Грея Gi виражається через біти двійкового коду Bi наступним чином: $G_i = B_i \oplus B_{i+1}$,Де $\oplus$— операція «виключає АБО»; біти нумеруються справа наліво, починаючи з молодшого.

2) Перехід з коду Грея в двійковий код:

Перетворення коду Грея в двійковий код — можна виразити формулою: $B_i = B_{i+1} \oplus G_i$, причому перетворення здійснюється побітно, починаючи зі старших розрядів, і значення Bi + 1, використовуване у формулі, обчислюється на попередньому кроці алгоритму. Дійсно, якщо підставити в цю формулу вищенаведене вираз для і-го біта коду Грея, отримаємо:

$$B_i = B_{i+1} \oplus G_i = B_{i+1} \oplus (B_i \oplus B_{i+1}) = B_i \oplus (B_{i+1} \oplus B_{i+1}) = B_i \oplus 0 = B_i.$$

Однак наведений алгоритм, незручний для програмної реалізації, тому на практиці використовують видозмінений алгоритм:

$$B_k = \bigoplus_{i=k}^{N} G_i,$$

де N — кількість бітів в коді Грея (для збільшення швидкодії алгоритму як N можна взяти номер старшого ненульового біта коду Грея); знак $\oplus$ означає підсумовування за допомогою операції «виключне АБО», тобто

$$\bigoplus_{i=k}^{N} G_i = G_k \oplus G_{k+1} \oplus \ldots \oplus G_{N-1} \oplus G_N.$$

Дійсно, підставивши в формулу вираз для і-го біта коду Грея, отримаємо

$$B_k = \bigoplus_{i=k}^{N} G_i = \bigoplus_{i=k}^{N} (B_i \oplus B_{i+1}) = (B_k \oplus B_{k+1}) \oplus (B_{k+1} \oplus B_{k+2}) \oplus \ldots \oplus (B_{N-1} \oplus B_N) \oplus (B_N \oplus B_{N+1}) =$$
$$= B_k \oplus (B_{k+1} \oplus B_{k+1}) \oplus \ldots \oplus (B_N \oplus B_N) \oplus B_{N+1} = B_k \oplus B_{N+1} = B_k$$

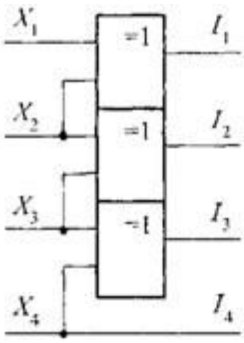


Рис. 4.36. Преобразователь прямого кода в код Грея

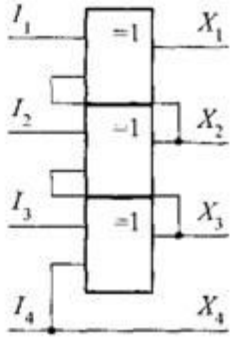


Рис. 4.37. Преобразователь кода Грея в прямой код

28. Програмуємі логічні матриці.

Программируемые логические ИС (ПЛИС, PLD) это ИС с регулярной структурой (повторяющимися одинаковыми блоками), которую можно сконфигурировать в заданную структуру. Они могут быть перепрограммированы. PLD (Programmable Logic Device) – программируемое логическое устройство.

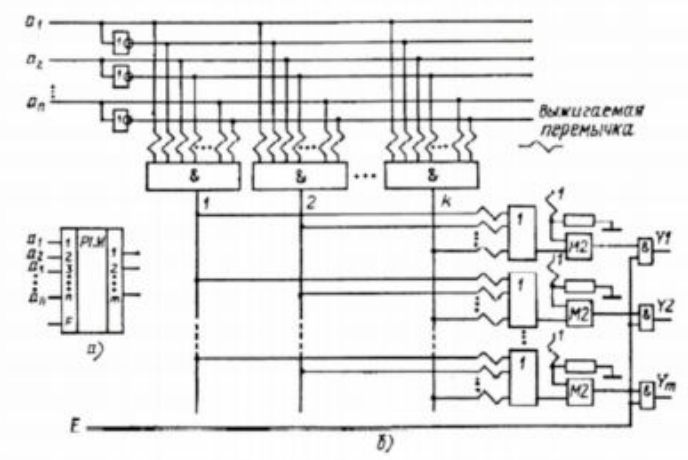


Рис. 3.19. Условное обозначение *(а)* и функциональная схема *(б)* программируемой логической матрицы (зигзагами обозначены разрушаемые перемычки)

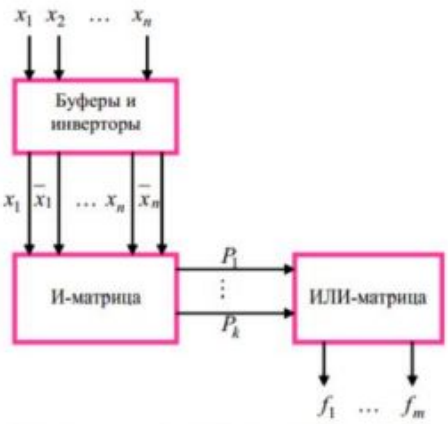


Рисунок 2.84 – Общая структура ПЛМ

Элементы ИЛИ в ПЛМ, так же как и элементы И, имеют на входах выжигаемые перемычки, с помощью которых они подключены ко всем вертикальным шинам. После выжигания на *программаторе* ненужных перемычек у элементов ИЛИ также остаются лишь те связи с вертикалями, которые необходимы потребителю. Техническая реализация элементов ИЛИ такова, что после выжигания перемычек на «ни к чему не подключенных» входах ИЛИ обеспечиваются уровни логического нуля.

В ПЛМ И-матрица и ИЛИ-матрица программируемые. Программируемые переключатели создают две проблемы. Первая - сложностью их точного выполнения при производстве. Вторая - уменьшением быстродействия схемы. Эти проблемы привели к реализации программируемого устройства с программируемой И-матрицей и фиксированной ИЛИ-матрицей (рис.2.87). Такие устройства называются ПМЛ – программируемой матричной логикой. Они проще в производстве, дешевле, имеют характеристики лучше, чем у ПЛМ, поэтому стали более популярными

29. Перемножувачі. Схеми, принцип дії, застосування.

30. Перехідні процеси в логічних схемах.[більш повна інфа http://prom-komplekt.com/content/perekhodnye-protsessy-v-logicheskikh-skhemakh]

Задержка логической схемы состоит из задержек срабатывания логических элементов и задержек распространения сигналов по цепям связи между ними. Трудоемкость учета задержек зависит от соотношения значений

задержек самих логических элементов и задержек в цепях связи. Если эти значения близки, то задержки разных трактов схемы можно определить лишь после размещения элементов на поверхности платы или кристалла БИС, когда станут известными фактические длины связей. Если при этом задержки . некоторых цепей не соответствуют требуемым, то нужно или переставлять элементы, или даже вносить изменения в функциональную схему, снова трассировать связи и снова определять задержки в них.

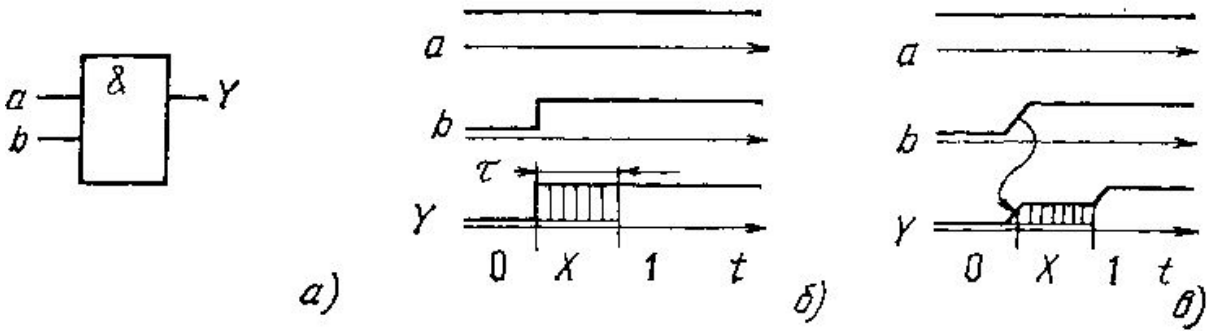


Рис. Способы изображения состояния неопределенности логического элемента

Уровень выхода элемента в течение отрезка времени от минимально возможного до максимально возможного значения задержки, когда фактическое состояние выхода элемента разработчику не известно, называют состоянием неопределенности и обозначают символом x. Состояние x, поступая на входы других логических элементов, может в зависимости от типа элемента порождать на их выходе или определенные состояния 1 или 0, или также неопределенное, обозначаемое X. Поведение логических элементов задается при этом законами уже не двоичной, а одного из видов троичной логики. Эффективным средством анализа переходных процессов в схемах являются временные диаграммы. При их постро-ении состояние неопределенности изображают одним из двух способов, которые показаны для элемента И (рис. 5.2, а). Изображение на рис. 5.2, б строже, но менее наглядно; изображение на рис. 5.2, в нагляднее, но может быть спутано с состоянием высокого импеданса элемента, имеющего три состояния выхода. Линии со стрелками обозначают причинно-следственные отношения в цепочке переключений. Линия начинается на фронте, который непосредственно вызывает переключение рассматриваемого элемента и оканчивается стрелкой на

фронте выходного сигнала этого элемента. Наличие таких указателей заметно облегчает понимание работы сложных схем. На рис. 5.3 показан фрагмент схемы (а) и варианты начертания временных диаграмм переходных процессов. Здесь и в дальнейшем для обозначения выходного сигнала элемента используется номер самого элемента.

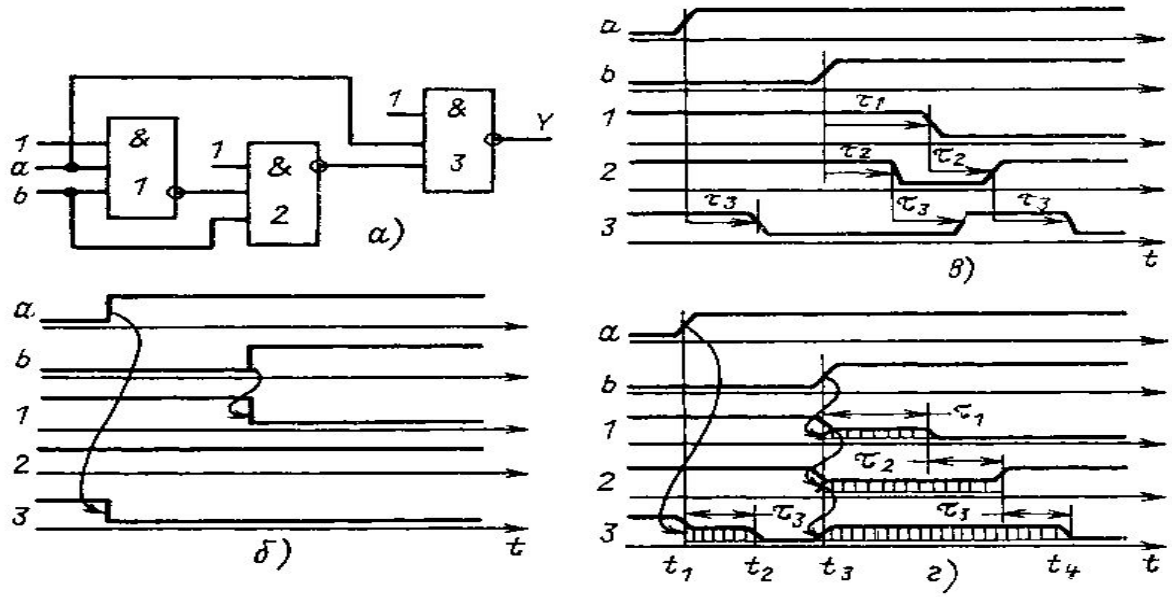


Рис 2. Временные диаграммы переходных процессов

а — фрагмент схемы; б, в, г — изображение переходных процессов; б — без учета задержек элементов, в —в предположении, что задержки максимальны; г —с использованием состояния неопределенности

5.3, б игнорирует переходные процессы в элементах и схеме. Такие диаграммы применяют, когда основной целью является иллюстрация логических и причинно-следственных отношений, а длительностью переходных процессов по сравнению с интервалами между поступлением сигналов можно пренебречь.

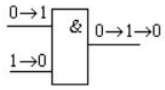
Диаграмма на рис 2 в построена в предположении, что все элементы имеют максимально возможные значения задержки. Эта диаграмма наглядна, поэтому удобна для первого знакомства с поведением сложной схемы. Но она годится лишь для оценки максимальной длительности переходного процесса. Делать по такой диаграмме выводы о состояниях элементов во время переходного процесса непозволительно: это лишь один частный случай из множества возможных процессов.

Диаграмма рис. 5.3, г учитывает состояния неопределенности элементов в соответствии с (5.1). Она достаточно строго моделирует поведение схемы

при любых комбинациях задержек, допускаемых паспортами элементов. Полезно сравнить диаграммы на рис. 5.3, виг, обращая внимание на их расхождения, причиной которых является общность диаграммы г и частность диаграммы е. До момента tl и после момента t/4 обе диаграммы совпадают.

31. Перегони в цифрових схемах.

Всі елементи і лінії зв'язку вносять затримки на поширення сигналів. Різниця часових параметрів реальних елементів, а також різна довжина ланцюгів проходження сигналів можуть бути причиною неодночасної зміни станів входів деяких елементів схем. Наприклад, нехай на елемент I подаються послідовності сигналів–«01», «10». У обох випадкахрівень сигналу на виході елемента–0. Але якщо сигнал на першому вході зміниться раніше, аніж на другому, то на певний час може виникнути комбінація вхідних сигналів «11», і на виході з’явиться короткочасний одиничний імпульс. Маємо вигляд напруги на виході елементу



При деяких поєднаннях затримок на черговому вхідному наборі вихідні сигнали логічних елементів можуть приймати значення, не передбачені законом функціонування схеми. У таких
випадках говорять, що на даному наборі існують так звані перегони між сигналами. В залежності від місця виникнення перегонів розрізняють перегони по входу і перегони по виходу.Усталені значення сигналів в комбінаційних схемах визначаються виключно значеннями сигналів на
зовнішніх входах. У таких схемах досить легко позбутися "помилкових" імпульсів, що з'являються
внаслідок перегонів. Наприклад, сигнали з виходів схеми можна знімати з деякою затримкою. У
схемах з пам'яттю боротьба з перегонами є більш складною, оскільки "помилкові" імпульси
можуть запам'ятовуватися і впливати на подальшу роботу схеми. Наприклад, розглянемо схему:

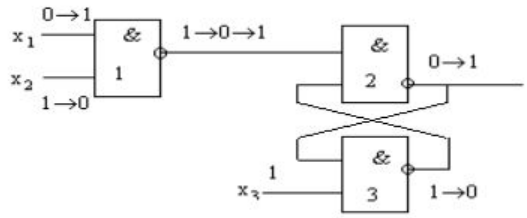


Рис. 6.3

Нехай вхідний набір сигналів 011 змінюється на 101, а на виходах елементів 2 і 3 (які утворюють тригер) спочатку були значення 0 і 1, відповідно. Якщо вхідний сигнал x1 зміниться раніше, ніж x2, то на виході елемента 1 з'явиться короткочасний нульовий імпульс, що призведе до зміни сигналів на елементах 2 і 3. У разі ж, коли на виході елемента 1 цієї статті не буде одиничного імпульсу, тригер не змінить свого стану. Іншими словами, значення, що встановилися на виходах елементів 2 і 3 будуть залежати від співвідношень моментів змін сигналів на x1 і x2.

Способи уникнення стану перегонів : 1)затримка, поки не встановиться в нормальний сигнал

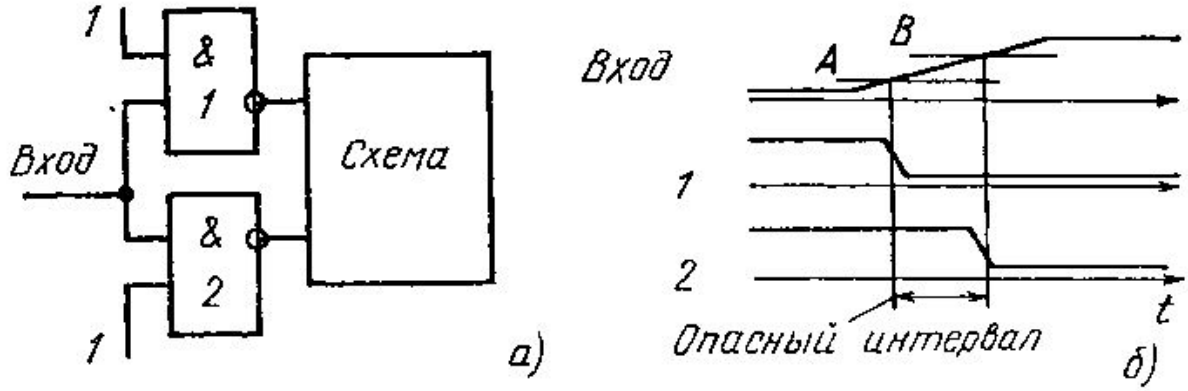
2)зібрати схему без стану перегонів

32. Перегони по виходу.

Основна різниця маж перегонами по входу й по виходу полягає в тому, що перегони по входу викликані елементам, які розміщені в попередніх колах – різницями рівнів спрацьовування, затримками і т. д. Перегони по виходу викликані самими кінцевими елементами й приносять шкоду елементам, що розміщені далі. Головна небезпека полягає в тому, що якщо перегони по вхожу можна хоч якось зменшити – поставити інтегруюче RC-коло і т. д., то перегони по виходу придушити неможливо. Наприклад, якщо на ході елемента змінились значення вхідних сигналів, то вихід утвориться лише через певний проміжок часу – час затримки. Потім вхідні сигнали знову змінюються, вихід змінюється пізніше. Утворюється непередбачуваний сигнал – перегони по виходу. Таким чином, випадкова, незначна зміна вхідних сигналів може викликати перегони на виході. Основний спосіб боротьби з перегонами по виходу – урахування часу затримки елемента та його інерційності в наступних колах (схемах).

Гонки по входу возникают, когда ветвящийся сигнал поступает на элементы, имеющие разброс по уровню срабатывания, а фронт этого

сигнала излишне пологий. Если длительность фронта входного сигнала заметно больше времени срабатывания элементов, то где-то в середине фронта будет существовать отрезок времени, когда с точки зрения одного элемента входной сигнал уже равен 1, а с точки зрения другого — еще равен 0.



33. Тригери. Схеми, принцип дії, застосування.

Триггер — это устройство с двумя устойчивыми состояниями, предназначенное для записи и хранения информации. Под действием входных сигналов триггер может переключаться из одного состояния в другое.

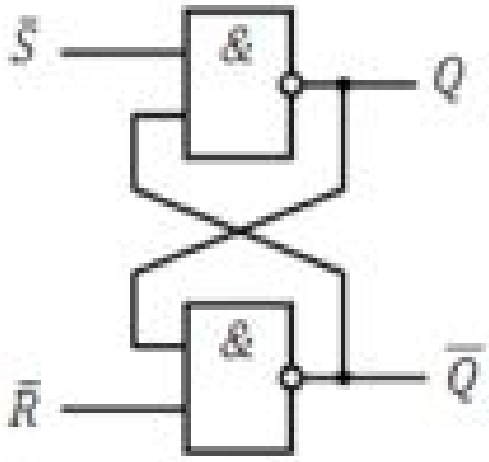
Тригер - елементарний автомат, що складається з елементу пам’яті та керуючої системи.

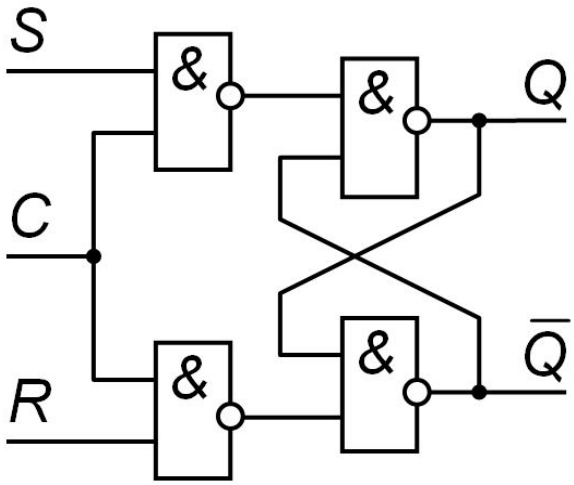
Триггерами называют[5] такие логические устройства, выходные сигналы которых определяются не только сигналами на входах, но и состоянием элементов памяти.

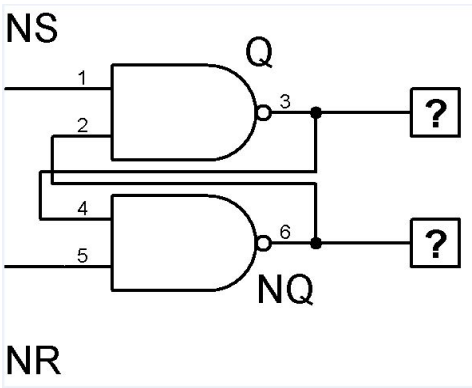
Тригер - логічна схема зі зворотнім зв’язком що має два сталих стани одиниці та нуля.

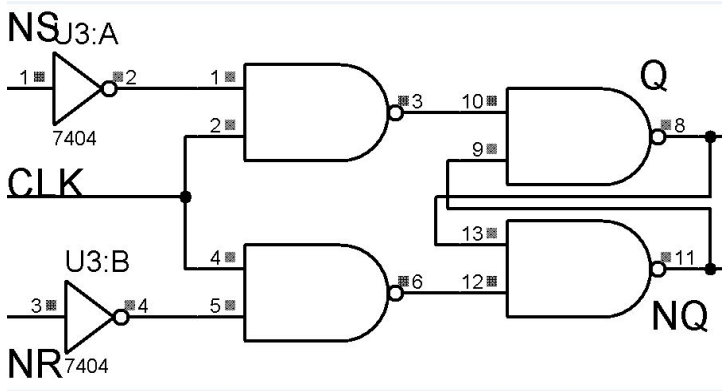
RS, R*S*, JK, D, T

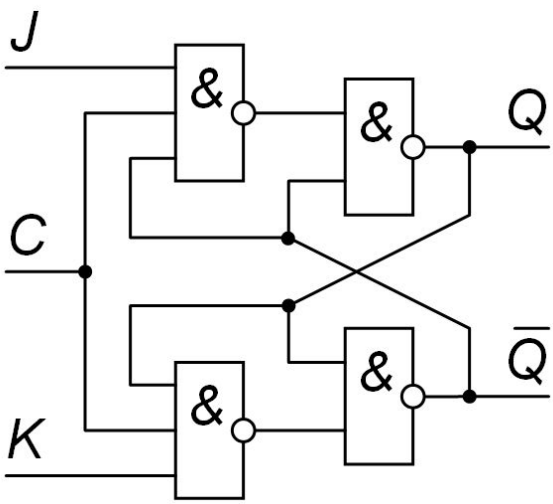
Используются, в основном, в вычислительной технике для организации компонентов вычислительных систем: регистров, счётчиков, процессоров, ОЗУ.

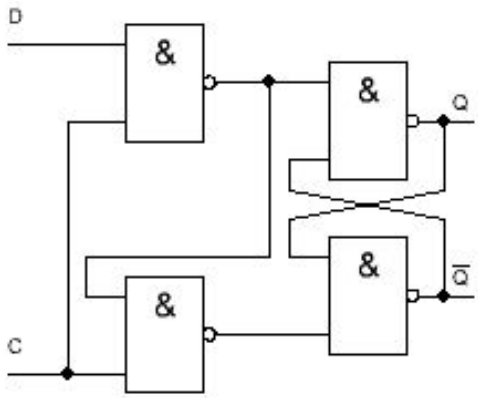

S
R
Q
&

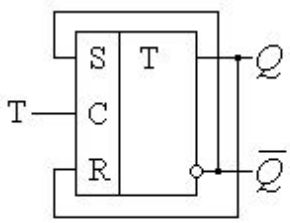

S
C
R
Q
&


NS
NR
Q
NQ
?
?
1
2
3
4
5
6


NS
U3:A
7404
CLK
U3:B
NR
7404
Q
NQ
1
2
3
4
5
6
8
9
10
11
12
13

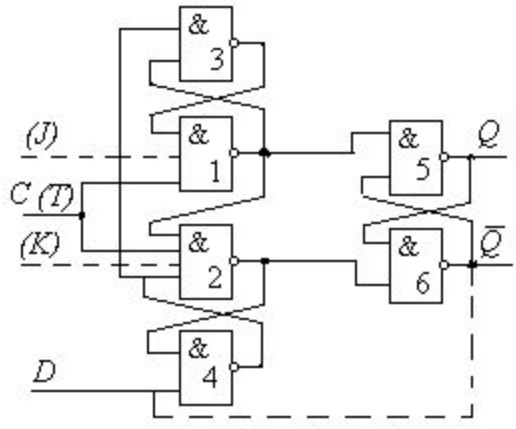

J
C
K
Q
&


D
C
Q
&


S
T
C
R
T
Q

a

34. Шестиелементні тригери (Вебба). Схеми, принцип дії, застосування.



Среди триггеров с динамическим управлением наиболее распространен шестиэлементный или так называемая схема трех триггеров. Другие его названия: триггер Вебба, триггер с самоблокировкой. На основе шестиэлементного триггера могут быть построены *D*, *T* и *JK*-триггеры (см.рис.)

Без учета пунктирных линий на рисунке представлена схема *D*-триггера.

С учетом пунктирной линии,соединяющей выход элемента 6 со входом элемента 4 и игнорировании *D*-входа, на схеме показан *T*-триггер.

С учетом пунктирных линий на входах элементов 1 и 2, пунктирной линии, соединяющей выход элемента 6 со входом элемента 4 и игнорировании *D*-входа на схеме представлен *JK*-триггер. Шестиэлементный триггер фактически состоит из трех *RS*-триггеров (элементы 1 и 2, 3 и 4, 5 и 6), отсюда и название – схема трех триггеров. Выходным триггером является *RS*-триггер на элементах 5 и 6.

Работа схемы как D-триггера.

Элементы 1 и 2 выполняют роль входных конъюнкторов.

При *C*=0 оба они закрыты и триггер (5-6) не реагирует на изменения *D*-входа, находится в режиме хранения.

По фронту *C*-сигнала в зависимости от уровня на *D*-входе 2 открывается один из конъюнкторов (1 или 2), триггер (5-6) переключается или подтверждается его состояние, если оно совпадает с требуемым. Элементы 4 и 3, управляемые *D*-входом, своими уровнями подготавливают один из

конъюнкторов (2 или 1) к тому, чтобы он открылся очередным *C*-сигналом.

Система связей элементов 1-4 обладает следующим свойством: переключившись на фронте *C*-сигнала в 0, элемент 1 по связи выход элемента 1 – вход элемента 2 блокирует для элемента 2 возможность открытия, даже если несколько позже, уже при *C*=1, вход *D* переключится в 0. Аналогично поведет себя и элемент 2: сработав на фронте *C*-сигнала, по цепочке элементов 2-4-3-1 блокирует срабатывание элемента 1.

Таким образом, поведение триггера при *C*=1 показывает, что он не прозрачен по *D*-входу ни для каких его изменений, то есть триггер переключается только по перепаду *C*-сигнала.

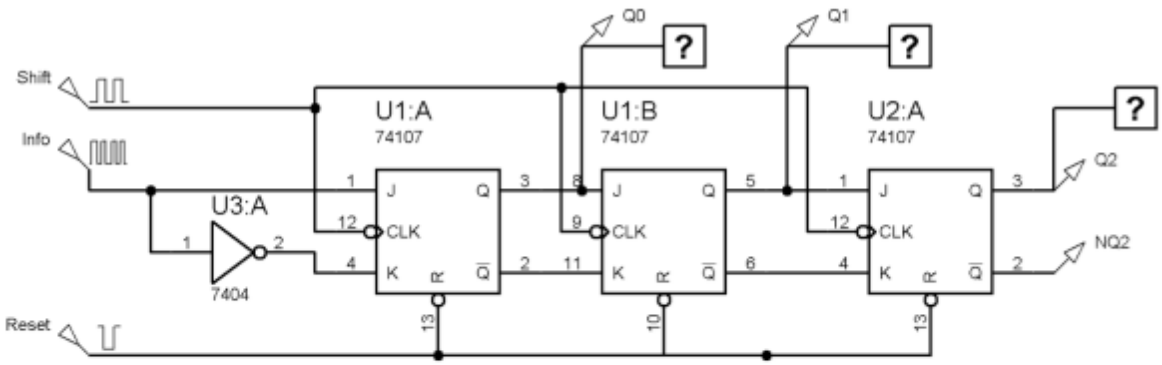
Схема является противогоночной: все элементы переключаются последовательно, то есть в ней отсутствуют параллельные пути прохождения сигналов.

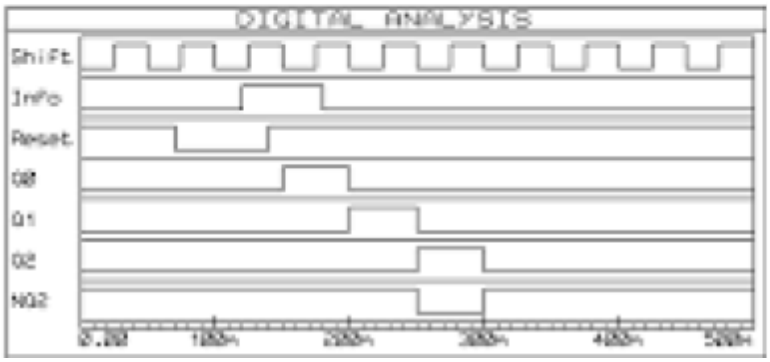
Задержка распространения сигнала по пути *CQ* равна сумме задержек элементов 2, 6 и 5. Время подготовки . определяется временем неопределенного состояния на входах элементов 1 и 2, а время выдержки (время в течение которого после фронта *C*-сигнала уровень *D*-входа не должен изменяться) определяется временем, необходимым для заблокирования элементом 1 входа элемента 2.

35. Послідовно-паралельні регістри. Схеми, принцип дії, застосування.

Последовательно-параллельные регистры имеют один информационный вход для последовательного ввода числа в режиме сдвига и выходные вентили для выдачи n-разрядного числа параллельным кодом. Такие регистры выполняют преобразование последовательного кода в параллельный.

Як бачимо, зсув відбувається послідовно з кожним тактовим сигналом на вхід Shift. Вхід послідовного сигналу Info є джерелом для “розпаралелювання” на виходах Q0-Q2. Одночасно кожен з цих виходів з наступним тактовим імпульсом приймає нове значення (Info->Q0, Q0->Q1, Q1->Q2)





36. Однофазна та двофазна синхронізація. [Misha Smaga]

Існують прилади, в яких перехід в новий стан виконується не тільки за рахунок надходження інформаційних сигналів, а і за рахунок надходження синхронізуючих сигналів. Загалом існує синхронізація по фронту (динамічна синхронізація), та синхронізація по рівню сигналу (статична синхронізація.). Якщо до якогось пристрою, з певних міркувань, надходить 2 послідовності синхроімпульсів (які зсунуті один відносно другого), то в такому випадку кажуть про двохфазну синхронизацію приладу. У випадку коли до пристрою підходить лише одна послідовність синхроімпульсів, то використовується однофазна синхронізація. Двохфазна синхронізація використовується для прискорення системи, шляхом організації роботи їх частин з різними швидкостями. Це робиться за допомогою використання більш високочастотних сигналів в деяких елементах системи.

Система синхронізації характеризується:

1. Довжиною тактового періоду T_τ
2. Довжиною фазового періоду T_ϕ
3. Тривалістю синхроімпульсу T_i

Синхронізація - є найбільш універсальним засобом боротьби з перегонами. Її суть полягає у наступному: по всьому цифровому
пристрою створюється єдина система синхронізуючих сигналів. В залежності від типу елементної бази, використовуються однофазна або двофазна системи синхронізації.

37. Циклічний регістр зсуву. Схема, принцип дії, застосування.

Циклічний зсув представляє собою операцію зсуву, при якій висунутий біт займає звільнився розряд. Існують наступні команди циклічного зсуву:

ROR; Циклічний зсув вправо;

ROL; Циклічний зсув вліво;

RCR; Циклічний зсув вправо з переносом;

RCL; Циклічний зсув вліво з переносом.

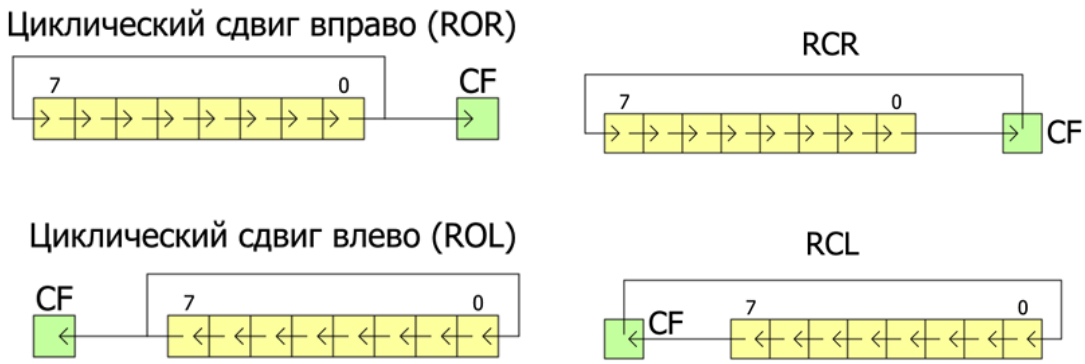
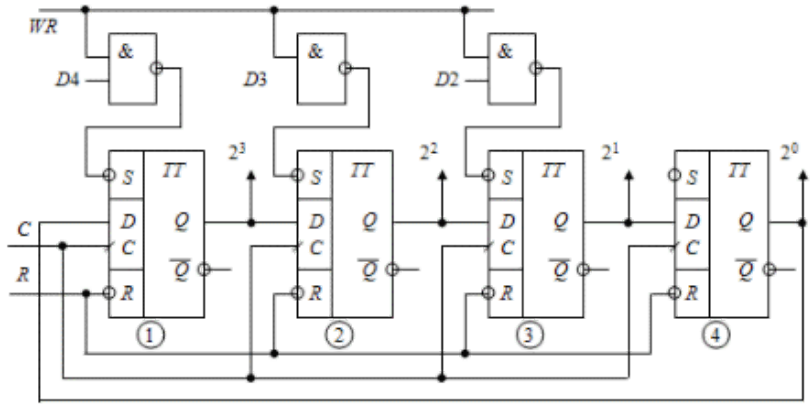


Схема яка реалізує звичайний циклічний зсув:



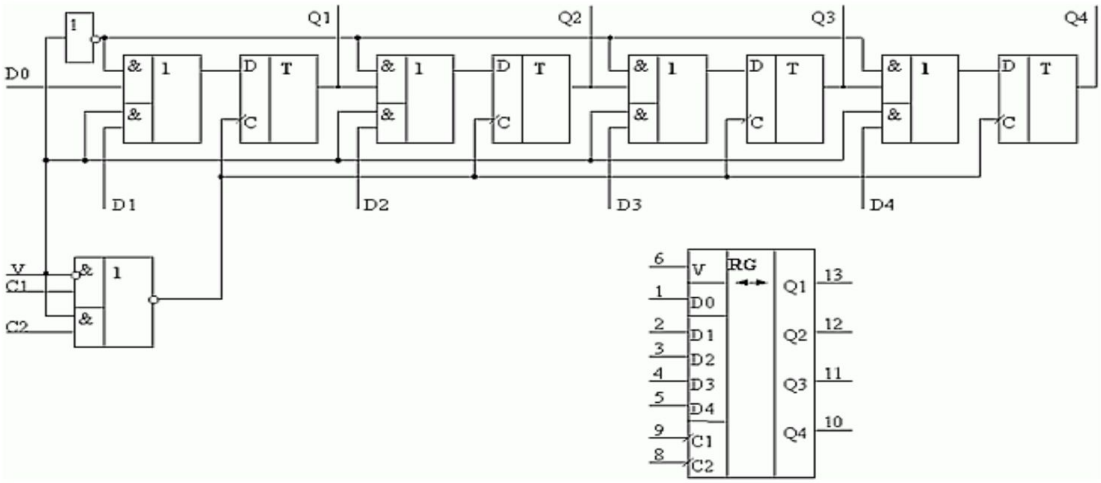
Нехай в регістр було записано двійкове число 0101, тоді після першого синхроімпульса (такту) на виході буде значення 1010, після другого такту на виході буде значення 0101, після третього такту - значення 1010 і т. Д., А якщо записано значення 0110, то будуть значення 0011, 1001, 1100, 0110 і т. д. відповідно.

38. Схема регістру зсуву, що дозволяє зсувати сигнал як праворуч, так і ліворуч.

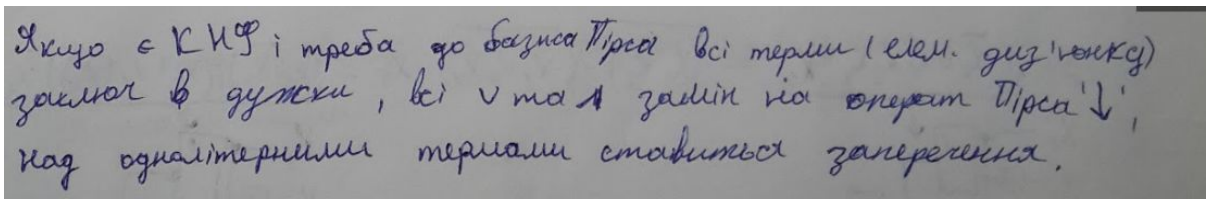
Кожен розряд складається з D-трігера та логічного елемента, який виконує функцію двоканального мультиплексора, який керується сигналом V. Регістр може записувати інформацію порозрядно послідовно в часі і всіма розрядами одночасно. При послідовному способі запису сигнал V повинен бути низького рівня, а код, який записується, повинен надходити на вхід D0. З кожним тактовим імпульсом C1 вхідний код просувається на один розряд в сторону старшого розряду. При паралельному способі запису код подається на входи D4-D1. Запис проводиться в паузі між тактовими імпульсами C1 при V=1 імпульсом C2.

Якщо після запису число потрібно зсунути, то регістр переводять в режим зсуву сигналом V=0 і керують за допомогою імпульсів C1. При умові, що Q1 – вихід молодшого розряду, а Q4 – старшого, інформація в регістрі зсувається вліво (зворотній зсув). Але він може бути перетворений і в регістр із зсувом вправо (прямий зсув). Для цього необхідно виконати наступні зовнішні з’єднання: D3 з Q4, D2 з Q3, D1 з Q2. Код записують по входу D4 при V = 1, а керують регістром тактовими імпульсами C2. Таким чином, в розглянутому регістрі виконуються умови як прямого, так і

зворотнього зсувів. Регістр має виходи від всіх розрядів, отже дозволяє зчитувати записаний код як в послідовній, так і в паралельній формах подання інформації в часі.



39. Правила переходу з КНФ до базису Пірса (або-ні).



Поскольку базис Пирса обладают свойством функциональной полноты, любая ПФ или система ПФ может быть представлена в этих базисах. Следовательно, любая КС может быть синтезирована в базисе Пирса с использованием однотипных логических элементов. Базисы Пирса связаны с каноническим базисом формулами де Моргана, которые в общем виде записываются следующим образом: $x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n = \overline{x_1 \vee x_2 \vee \ldots \vee x_n} = \bar{x}_1 \cdot \bar{x}_2 \cdot \ldots \cdot \bar{x}_n;$ Минимизация логических функций в базисах И-НЕ, требует разработки специальных правил преобразования для выполнения склеиваний и поглощений. Гораздо более простым подходом является применение хорошо разработанных методов минимизации ПФ для канонического базиса И-ИЛИ-НЕ с последующим переходом к базису Пирса. Строго говоря, при таком способе минимизации в общем случае получаются не минимальные, а близкие к минимальным формы представления ПФ.

$D_j = \tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_n;$

$\tilde{x}_k \in \{x, \bar{x}\}, \quad k = 1,2,\ldots,n.$ Тогда СКНФ в общем виде можно записать как

$$\text{СКНФ}: \quad f(x_1, x_2, \ldots, x_n) = \bigwedge_{j=1}^{s} D_j.$$

$\overline{D_i} = \overline{\tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_n} = \tilde{x}_1 \downarrow \tilde{x}_2 \downarrow \ldots \downarrow \tilde{x}_n,$ откудаДля перехода от произвольной КНФ к базису ИЛИ-НЕ (стрелка Пирса) необходимо в исходном выражении: поставить знак инверсии над всем выражением, заменить все знаки конъюнкции на знаки дизъюнкции, над каждым конъюнктивным членом, входившим в КНФ, поставить знак инверсии (заменить все знаки дизъюнкции и конъюнкции на знак операции «стрелка Пирса»). Рассмотрим пример.Осуществим переход от КНФ к базису ИЛИ-НЕ (стрелка Пирса).

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_2 \vee x_4) \cdot (\bar{x}_3 \vee \bar{x}_4) \cdot (x_1 \vee x_2 \vee \bar{x}_4) =$$
$$= \overline{(\bar{x}_2 \vee x_4)} \vee \overline{(\bar{x}_3 \vee \bar{x}_4)} \vee \overline{(x_1 \vee x_2 \vee \bar{x}_4)} = (\bar{x}_2 \downarrow x_4) \downarrow (\bar{x}_3 \downarrow \bar{x}_4) \downarrow (x_1 \downarrow x_2 \downarrow \bar{x}_4).$$

$\bar{x} = \overline{x \vee x} = x \downarrow x,$ т.е. в качестве инвертора выступает элемент ИЛИ-НЕ, все входы которого объединены в один.При синтезе КС в универсальных базисах может возникнуть необходимость в переходе от представления функции в КНФ к представлению в базисе И-НЕ. В такой ситуации можно пользоваться следующими формулами:

$$\bigwedge_{j=1}^{s} D_j = \overline{(\tilde{\tilde{x}}_1 \mid \tilde{\tilde{x}}_2 \mid \ldots \mid \tilde{\tilde{x}}_n)_1 \mid (\tilde{\tilde{x}}_1 \mid \tilde{\tilde{x}}_2 \mid \ldots \mid \tilde{\tilde{x}}_n)_2 \mid \ldots \mid (\tilde{\tilde{x}}_1 \mid \tilde{\tilde{x}}_2 \mid \ldots \mid \tilde{\tilde{x}}_n)_s}.$$

На практике в некоторых случаях применяются логические элементы с ограниченным числом входов, например, элементы 2И-НЕ, т.е. в КС могут входить только двухвходовые элементы. Выражения для функций, описывающих такую КС, могут быть получены путем выполнения преобразований. Для перехода к базису 2И-НЕ:

$x_1 \mid x_2 \mid x_3 = \overline{x_1 \cdot x_2 \cdot x_3} = \overline{\overline{(x_1 x_2)} \cdot x_3} = \overline{(x_1 \mid x_2)} \mid x_3;$

$x_1 \mid x_2 \mid x_3 \mid x_4 = \overline{x_1 \cdot x_2 \cdot x_3 \cdot x_4} = \overline{\overline{(x_1 x_2)} \cdot \overline{(x_3 x_4)}} = \overline{(x_1 \mid x_2)} \mid \overline{(x_3 \mid x_4)}.$

Для перехода к базису 2ИЛИ-НЕ:

$x_1 \downarrow x_2 \downarrow x_3 = \overline{x_1 \vee x_2 \vee x_3} = \overline{\overline{(x_1 \vee x_2)} \vee x_3} = \overline{(x_1 \downarrow x_2)} \downarrow x_3;$

$x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4 = \overline{x_1 \vee x_2 \vee x_3 \vee x_4} = \overline{\overline{(x_1 \vee x_2)} \vee \overline{(x_3 \vee x_4)}} = \overline{(x_1 \downarrow x_2)} \downarrow \overline{(x_3 \downarrow x_4)}.$

40. Правила переходу з ДНФ до базису Шефера (і-ні).

Якщо функція записана у вигляді ДНФ, то всі терми (елементарні кон'юнкції) заключаються в дужки і всі знаки кон'юекції та диз'юнкції змінюються на |, а над однолітерними термами ставиться знак заперечення.

Аксіоми алгебри Шефера:

$$x/0=\overline{x\cdot 0}=1; \qquad x/x=\overline{x\cdot x}=\bar{x}$$
$$x/1=\overline{x\cdot 1}=\bar{x}; \qquad x/\bar{x}=\overline{x\cdot\bar{x}}=1.$$

41. Кубічне представлення логічних функцій.

Кожному набору L, булевих змінних може бути поставлена у відповідність точка L-мірного простору в системі L координат. Так як змінні є булевими, то і простір називається L-мірним булевим простором або L-кубом.На рис. показано кубічне подання функцій L булевих змінних при L = 0,1,2,3. якщо функція дорівнює одиниці на будь-якому наборі, то відповідна вершина куба
обводиться знаком О

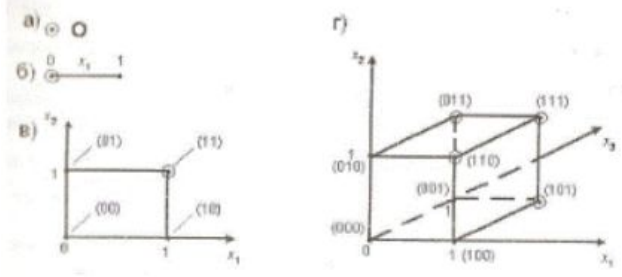


Рис. а) відповідає випадку L= 0, при цьому функції є константами. Так, на рис. а) задана константа 0. Таким чином, булевим функціям при числі змінних L = 0 відповідають 0-куби.Рис. б) задає функцію y=nx1. Функції однієї змінної відповідають ребру або 1-куб. Рис. в) задає функцію y=x1x2. Функції двох змінних відповідає грань або 2-куб. У дужках біля кожної вершини записано значення вхідного набору, який відповідає цій вершині. Набори записуються в порядку зростання індексу булевої змінної. Для функції двох змінних набори відповідають парам (x1x2). .Рис г) задає мажоритарну функцію. Функції трьох змінних відповідає 3-куб, тобто куб, що має три виміри. З зростанням числа вимірів куби починають відрізнятися від

звичних геометричних фігур. Як випливає з аналізу рис. г),3-куб може бути представлений як два 2-куба для змінних x1,x2. Однойменні вершини цих кубів з'єднані ребрами, протилежними кінцям яких відповідають протилежні значення змінної x3. Аналогічним чином куб функції L змінних може бути представлений як два куби функції L-1 змінних. Одному з цих кубів відповідає, наприклад, значення xL = 0, другому- xL = 1.Кубічне уявлення є потужним засобом для введення інформації про булеві функціях в ЕОМ для подальшої обробки. Наприклад, вершини 3-куба 101 і 111 пов'язані ребром (Рис. г). Інформація про це ребро представляється як 1 * 1, де * означає змінюваний елемент набору. Аналогічно, вершини 010, 011, 110, 111 входять в одну грань 3-куба (Рис. 8.1 г) і інформація про цю грань може бути представлена як *1*. Таким чином, якщо 2^n вершин є n-кубом, що входять в L-куб, то вони можуть бути представлені одним вектором, у якого n елементів замінені символом *. Це уявлення є більш компактним, ніж таблиця істинності.

42. Двійково-десяткові коди.

43. Метод підбору коефіцієнтів (перевод чисел з однієї позиційної системи числень в іншу)

Любое число N можно представить полиномом с основанием q1, но это же число можно представить другим полиномом с основанием q2, иначе: N(q1)= N(q2). Представим это следующим выражением:

$$N(q_1)=a_n q_1^n+a_{n-1}q_1^{n-1}+\ldots+a_1q_1^1+a_0q_1^0+a_{-1}q_1^{-1}+\ldots+a_{-m}q_1^{-m}=$$

$$=b_k q_2^k+b_{k-1}q_2^{k-1}+\ldots+b_1q_2^1+b_0q_2^0+b_{-1}q_2^{-1}+\ldots+b_{-s}q_2^{-s}=N(q_2).$$

Таким образом, для перевода числа в другую систему счисления, необходимо:

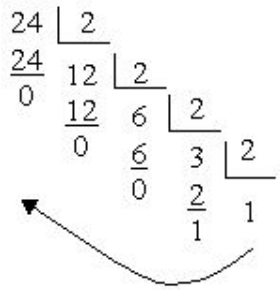
1. определить новые весовые коэффициенты,
2. умножить их на новое основание в степени веса разряда и просуммировать слагаемые.

Правило. Выбираем число со степенью основания 2 так, чтобы оно не превышало данное число, но было близким к нему снизу, ставим коэффициент равный 1, затем подбираем коэффициенты меньших

степеней полинома для остатков, так чтобы сумма давала исходное число. На местах степеней (весов разрядов), входящих в сумму числа, ставим коэффициенты равные 1, на остальных местах - 0. Все действия должны выполняться по правилам арифметики исходной системы счисления.
Пример. Перевести десятичное число 96 в двоичную систему.
Решение. 96 = 1*2^6 +1*2^5 +0*2^4 +0*2^3 +0*2^2 +0*2^1 +0*2^0 = 1100000(2) .96= 1*64 + 1*32.

44. Метод переводу діленням на основу нової системи (перевод чисел з однієї позиційної системи числень в іншу)

Для переводу цілих чисел з однієї системи числення з основою S в іншу з основою S1 потрібно це число послідовно ділити на основу S1 нової системи числення до тих пір, поки не отримаємо частку менше S1. Число у новій системі запишеться у вигляді залишків ділення, починаючи з останнього. Ця остання частка дає цифру старшого розряду у новій системі числення. Ділення виконують у вихідній системі числення.



Источник: http://5fan.ru/wievjob.php?id=31476

45. Метод переводу чисел діленням на основу в додатній степені (перевод чисел з однієї позиційної системи числень в іншу)

Звичайною для нас і загальноприйнятою є позиційна десяткова система числення. Як умовні знаки для запису чисел вживаються цифри.
Основою цієї системи є число десять. Основою системи числення називається число, яке означає, у скільки разів одиниця наступного розрядку більше за одиницю попереднього.

Загальновживана форма запису числа є насправді не що інше, як скорочена форма запису розкладу за степенями основи системи числення, наприклад

$130678=1*10^5+3*10^4+0*10^3+6*10^2+7*10^1+8$

Для переведення чисел із системи числення з основою p в систему числення з основою q з використанням арифметики старої системи числення з основою p потрібно:

- для переведення цілої частини:
    - послідовно число, записане в системі основою p ділити на основу нової системи числення, виділяючи остачі. Останні записані у зворотному порядку, будуть утворювати число в новій системі числення;
- для переведення дробової частини:
    - послідовно дробову частину множити на основу нової системи числення, виділяючи цілі частини, які й будуть утворювати запис дробової частини числа в новій системі числення.

Цим самим правилом зручно користуватися в разі переведення з десяткової системи числення, тому що її арифметика для нас звичніша.

46. Перевід правильних дробів множенням на основу системи (перевод чисел з однієї позиційної системи числень в іншу)

Правило. Для переводу правильного дробу (без цілої частини) необхідно, діючи в арифметиці
вихідної системи числення, помножити дріб, який переводиться на основу нової системи, у
результату відокремити цілу частину, а решту дробову частину знову помножити на основу і так до
отримання потрібного числа значущих цифр. Результат записувати як 0, ... і дробову частину в
порядку отримання.

Нехай є число 0.25. Переведемо його в двійкову систему:

0.25*2 = 0.5

0.5*2 = 1.0

0 * 2 = 0

Відповідно 0.25_(10) = 0.01_(2) - При переводі відсікаємо дробову частину, тобто 0.5 буде 0, а 1.0 буде 1

Далі перевіримо: 0.01_(2) = 2^(-1) * 0 + 2^(-2)*1 = 2^(-2) = ¼ = 0.25

Ті самі правила діють на будь-яку іншу систему числення.

Наприклад потрібно перевести число 206,116
Тоді, 206_(10)=11001110_(2) а 0,116 переводимо так:
.116 • 2 = 0.232
.232 • 2 = 0.464
.464 • 2 = 0.928
.928 • 2 = 1.856
.856 • 2 = 1.712
.712 • 2 = 1.424
.424 • 2 = 0.848
.848 • 2 = 1.696
.696 • 2 = 1.392
.784 • 2 = 0.784
Отримуємо число 206_(10)=11001110,0001110110_(2)

47. Кодування від'ємних чисел в ЕОМ

Прямий, зворотній і доповняльний коди В ЕОМ доцільно представляти знаки чисел з допомогою тих же символів, через які записується саме число. Для цього виділяється додатковий розряд, який називають знаковим і розташовують зліва від старшого розряду числа. Будемо позначати знакові розрядні числа як і відділяти їх крапками від цілої частини числа і комами від дробової. Для правильності виконання арифметичних операцій над числами, знаки яких закодовані числами, використовують спеціальні способи представлення чисел: прямий, зворотній і додатковий. Прямий код використовується для вводу-виводу інформації і в запам'ятовуючих пристроях. Додавання чисел в прямому коді (з однаковими знаками) не викликає труднощів. Однак додавання чисел з різними знаками в прямих кодах незручно, так як повинно бути

спеціальне обладнання для віднімання чисел і визначення знаку різниці. Операцію алгебраїчного додавання чисел можна звести до операцій додавання при використанні зворотних і додаткових кодів.

Представлення додатного числа в зворотному коді співпадає з його прямим кодом. Для отримання зворотного коду від'ємного числа в двійковій системі необхідно в знаковому розряді записати 1, а в інших розрядах одиниці замінити нулями, а нулі одиницями. Аналогічну заміну роблять при перетворенні зворотного коду від'ємного числа в прямий. На відміну від прямого коду, в зворотному не можна відкидати нулі після знакового розряду в цілій частині і нулі в кінці дробової частини від'ємного числа. Представлення додатного числа в доповнюючому коді співпадає з його прямим кодом. Правило формування додатного коду від'ємного числа формуються так: отримати зворотній код числа і додати 1 в молодший розряд числа. Перетворення доповнюючого коду від'ємного числа здійснюється або зворотнім шляхом (відняти 1 і перетворити в зворотній код) або утворити доповнюючий код до доповнюючого. Нуль, на відміну від прямого і зворотнього кодів, в доповнюючому коді має єдине представлення

48. Форми подання чисел в ЕОМ

В сучасних ЕОМ застосовуються два способу представлення чисел: з фіксованою крапкою і плаваючою крапкою.

В першому випадку місце коми, яка відділяє цілу частину числа від дробової, визначається на етапі конструювання ЕОМ. Зразу ж вказується кількість розрядів, які відводяться для зображення цілої і дробової частин. Причому кожному розряду комірки відповідає завжди один і той же розряд числа, що суттєво спрощує виконання арифметичних дій. Нехай, наприклад, комірка пам'яті машини має 24 двійкових розряду. Як ми знаємо, в комірку можна записати будь-яке машинне слово, тобто довільний набір з нулів і одиниць. Якщо це слово – число, то в конструкції машини може бути передбачено його представлення в формі з фіксованою комою. Наприклад, воно може бути таким: крайній зліва розряд – знаковий, потім наступні 9 розрядів відводяться під цілу частину і, на кінець , 14 розрядів, які залишилися, під дробову частину числа, тобто

кома тут завжди на одному і тому ж місці - після десятого розряду машинного слова (з врахуванням знакового розряду). Тоді найбільше число, яке можна представити, буде: (111111111,11111111111111)2. Видно, що воно менше, ніж 2^9. А найменше за модулем відмінне від нуля число дорівнює (000000000,00000000000001)2 = 2^-14 . Тобто, діапазон чисел, які можна записати в комірку пам'ті машини, тут такий: 2^-14 < |a| < 2^9

Форма з плаваючою крапкою. Форма представления вещественных (действительных) чисел, в которой число хранится в форме мантиссы и показателя степени.
Для того, щоб збільшити діапазон чисел, використовують другу форму запису чисел – з плаваючою комою. Будь-яке число в системі числення з основою Q можна записати так: a=A*Q^p . А називають мантисою числа, а P – порядком. Наприклад, в десятковій системі числення число 3,14 представимо у вигляді 3,14 = 0,314*10^1 . Тут мантиса дорівнює 0,314, а порядок 1. Очевидно, таке представлення далеко не однозначне. Число 3,14 записати так: 3,14=3,14*10^0 = 31,4*10^-1 = 0,0314*10^2 =... Порядок числа визначає положення коми в запису мантиси. При коректуванні порядку відповідним чином змінюється і положення коми – кома ніби "плаває". Звідси і назва методу представлення чисел.

Такая форма записи имеет недостаток: некоторые числа записываются неоднозначно (например, 0,0001 можно записать как $0{,}000001\times10^{2}$, $0{,}00001\times10^{1}$, $0{,}0001\times10^{0}$, $0{,}001\times10^{-1}$, $0{,}01\times10^{-2}$, и так далее), поэтому распространена (особенно в информатике) также другая форма записи — *нормализованная*, в которой мантисса десятичного числа принимает значения от 1 (включительно) до 10 (не включительно), а мантисса двоичного числа принимает значения от 1 (включительно) до 2 (не включительно), то есть 1 □ a < 10. В такой форме любое число (кроме 0) записывается единственным образом. Недостаток заключается в том, что в таком виде невозможно представить 0, поэтому представление чисел в информатике предусматривает специальный признак (бит) для числа 0.

49. Машинні алгоритми перетворення чисел

-Ліворуч двійково десяткове число записується розбите на декади.
-коди що записуються в БЦД зсуваються праворуч . якщо зі старшої декади в молодшу декаду перейде одиниця слід відняти 3 від тієї куди перейшла одиниця(додати в додатковому коді 1101).
-зсувати поки не закинчиться БЦД число
-всі числа, що вийшли з молодшої декади потрапляють у старший біт двійкового коду.
перевод числа 12(червона одиниця - та що перейшла зі старшої декади)

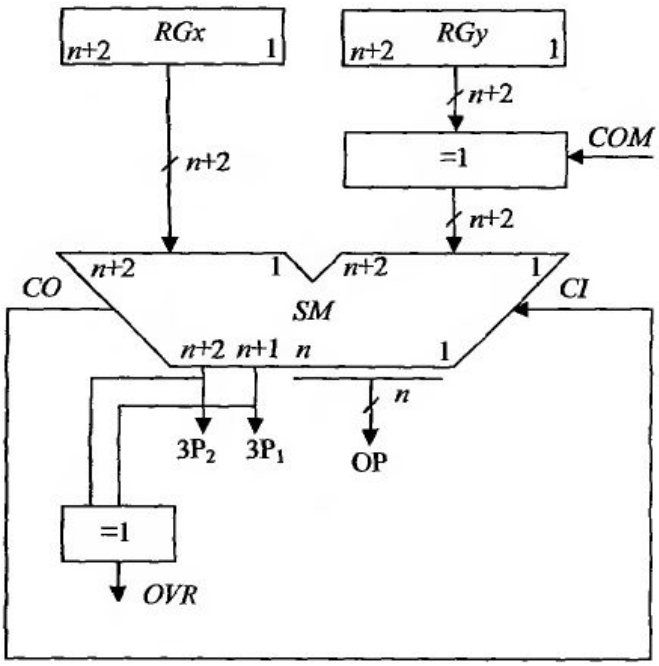
	0	0	0	1	0	0	1	0				
→		0	0	0	1	0	0	1	0			
-3					1	1	0	1				
		0	0	0	0	1	1	0				
→			0	0	0	0	1	1	0	0		
→				0	0	0	0	1	1	0	0	
→					0	0	0	0	1	1	0	0

50. Додавання чисел із знаком у машинних кодах [Misha Smaga]

При додаванні чисел в додатковому коді, одиниця переносу, що виникла, в знаковому розряді відкидається.
При додаванні чисел у зворотному коді одиниця переносу, що виникла, в знаковому розряді додається до молодшого розряду суми кодів.
Якщо результат арифметичних дій є кодом від'ємного числа, необхідно перетворити його в прямий код. При цьому зворотний код перетворюється в прямий заміною цифр у всіх розрядах, крім знакового на протилежні. Додатковий код перетворюється в прямий також, як і зворотний, з подальшим додаванням одиниці до молодшого розряду.

51. Додавання і віднімання чисел у зворотних кодах. Побудувати схему.

У ЗК операція віднімання замінюється операцією додавання, при цьому знаковий розряд і основні розряди обробляються як єдине ціле. Правильний знак суми утворюється автоматично в процесі підсумовування цифр знакових і основних розрядів з урахуванням переносів. Характерною рисою ЗК є циклічний перенос зі знакового розряду в молодший розряд суми, завдяки якому здійснюється корекція результату.



Операція віднімання виконується шляхом додавання зменшуваного до від'ємника, основні розряди якого інвертуються за допомогою елемента «ВИКЛЮЧНЕ АБО». Для цього на вхід COM подається 1. Під час додавання на цей вхід потрібно подати 0. Суматор формує основні розряди суми (ОР) та два знакові розряди ЗР1 та ЗР2. Корекція суми здійснюється за допомогою ланцюга циклічного переносу зі знакового розряду в молодший розряд суми CO -> CI. Цей ланцюг постійно замкнений, тому що перенос виникає тільки тоді, коли потрібно реалізувати корекцію суми. Щоб знайти переповнення розрядної сітки, використовується ознака переповнення OVR = ЗР1(+)ЗР2. За значення OVR = 0 переповнення розрядної сітки відсутнє, у випадку, коли OVR = 1 – наявне переповнення розрядної сітки.

52. Додавання і віднімання чисел у доповнювальних кодах. Побудувати схему.

У ДК операція віднімання, як і в ЗК, замінюється операціею додавання зменшуваного до від'ємного від'ємника. За підсумовування операндів у ДК автоматично отримуємо суму в ДК з урахуванням знаку. За застосування модифікованого коду корекції робити не треба при будь-якому сполученні знаків доданків. Факт переповнення розрядної сітки можна визначити за розбіжністю значень знакових розрядів. Старший знаковий розряд завжди зберігає знак результату.

$$\begin{array}{r} [A]_{\text{дк}}=00,10101 \\ +[B]_{\text{дк}}=00,01001 \\ \hline [C]_{\text{дк}}=00,11110. \end{array} \qquad \begin{array}{r} [A]_{\text{дк}}=00,10101 \\ +[B]_{\text{дк}}=11,10111 \\ \hline [C]_{\text{дк}}=00,01100. \end{array} \qquad \begin{array}{r} [A]_{\text{дк}}=00,01001 \\ +[B]_{\text{дк}}=11,01011 \\ \hline [C]_{\text{дк}}=11,10100. \end{array}$$

$$\begin{array}{r} [A]_{\text{дк}}=11,01011 \\ +[B]_{\text{дк}}=11,10111 \\ \hline [C]_{\text{дк}}=11,00010. \end{array} \qquad \begin{array}{r} [A]_{\text{дк}}=00,10101 \\ +[B]_{\text{дк}}=00,01110 \\ \hline [C]_{\text{дк}}=01,00011 \end{array} \text{ — додатне переповнення.}$$

В останьому випадку наявне додатне переповнення розрядної сітки та за застосуванням модифікованого коду старший розряд зберігає знак результату. Отже, отримане в результаті обчислень число є додатним.

$$\begin{array}{r} [A]_{\text{дк}}=11,01011 \\ +[B]_{\text{дк}}=11,10010 \\ \hline [C]_{\text{дк}}=10,11101 \end{array} \text{ — від'ємне переповнення.}$$

У цьому випадку наявне від'ємне переповнення розрядної сітки та за застосування модифікованого коду старший розряд зберігае знак результату. Отже, отримане в результаті обчислень число є від'ємним. Схема, що реалізуе операцію додавання і віднімання в ДК, аналогічна схемі для зворотніх кодів (схема з минулого питання), але в цьому випадку відсутній циклічний ланцюг корекції результату. За віднімання разом з одиничним сигналом на вхід СОМ подається одиниця на вхідний перенос СІ суматора. Це необхідно для зміни знака від'ємника, яке здійснюється інвертуванням усіх розрядів від'емника і додаванням 1 до його молодшого розряду. Під час додавання на входи СОМ і СІ потрібно подати 0.

53. Зсуви машинних кодів

Операция СДВИГ ВЛЕВО. Поразрядная операция «сдвиг влево» переносит содержимое каждого разряда первого операнда на то количество разрядов влево, которое задано вторым операндом, освобождающиеся разряды справа заполняются нулями. Результат операции содержит сдвинутое машинное слово, а сами операнды не изменяется. Операция СДВИГ ВПРАВО. Поразрядн3ая операция «сдвиг вправо» имеет некоторые особенности выполнения. По аналогии со сдвигом влево операция сдвига вправо на n разрядов интерпретируется как целочисленное деление на 2n. При этом заполнение освобождающихся старших разрядов производится таким образом, чтобы сдвиг соответствовал операции деления с учетом формы представления целого. Для беззнакового целого заполнение должно производиться нулями (логический сдвиг), а для целого со знаком - сопровождаться дублированием значения старшего знакового разряда (арифметический сдвиг). В последнем случае отрицательное число при сдвиге останется отрицательным

54. Арифметичний зсуви чисел поданих у зворотному коді.

Якщо під час зсуву від’ємного числа ліворуч за розрядну сітку виходить одиниця із знакового розряду то необхідно виконати корекцію $K = +2^{-k}$;

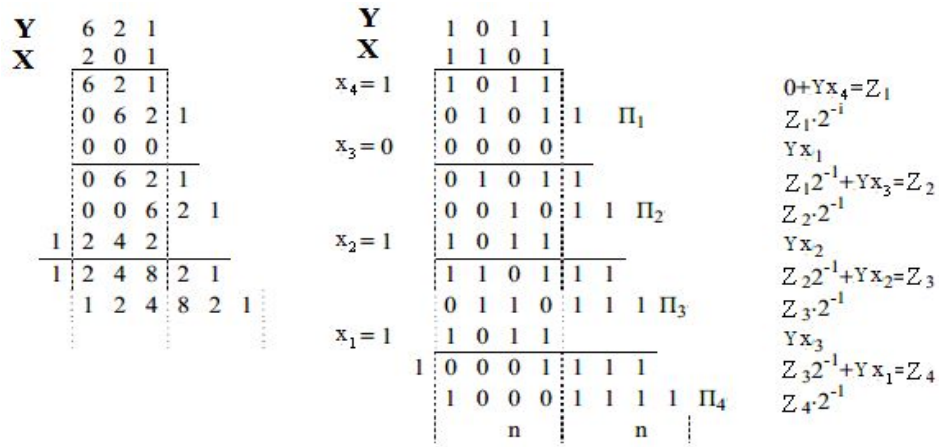
За зсуву праворуч від’ємного числа знаковий розряд переходить у поле основних розрядів і знову заповнюється тим самим значенням.

55. Арифметичний зсуви чисел поданих у доповнювальному коді.

56. Множення чисел (1- спосіб)

Множення з молодших розрядів множника та зсувом суми часткових доданків вправо.

У способі множення здійснюється з молодших розрядів множника, сума часткових добутків зсувається вправо, а множене залишається нерухомим. Наведемо приклад виконання множення за цим способом, зліва множимо два числа в десятковій формі, справа – в двійковій.

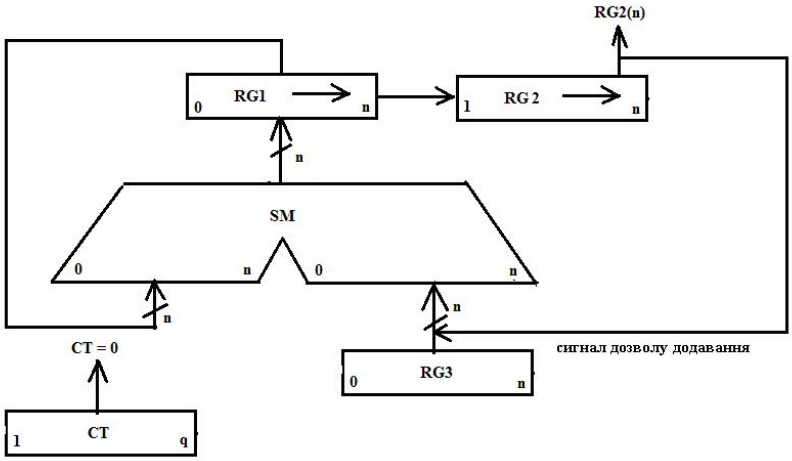


RG3 – регістр множеного.

RG2 – регістр множника.

RG1 – регістр добутку.

Перед початком множення X записується в RG2, а Y – в RG3



1 такт і-го циклу :

Аналізується значення RG2(n) – молодшого n-го розряду регістра RG2, в якому знаходиться чергова цифра множника. Вміст RG3 додається до суми часткових добутків, що знаходяться в регістрі RG1, якщо RG2(n) = 1, або не додається, якщо RG2(n) = 0.

2 такт і-го циклу :

Здійснюється зсув у регістрах RG1 та RG2, що еквівалентно множенню їх вмісту на 2^{-1}. За зсуву, цифра молодшого розряду регістру RG1 записується у вивільнюваний старший розряд регістра RG2. Після виконання n-циклів молодші розряди 2n-розрядного добутку будуть записані в регістр RG2, а старші – в регістр RG1

57. Множення чисел (2- спосіб) [Misha Smaga]

Множення з молодших розрядів множника та зсувом множеного вліво, при чому сума часткових добутків залишається нерухомою.

Даний спосіб множення вивчається в школі і його можна назвати шкільним методом множення.

Подаємо множення в наступному вигляді :

$Z = YX = ((\ldots((0+Y\,2^{-n}\,x_n) + Y\,2^{-n+1}\,x_{n-1}) + \ldots + Y\,2^{-1}\,x_1)$ Очевидно, що процес множення може бути зведений до n-кратного виконання циклу:

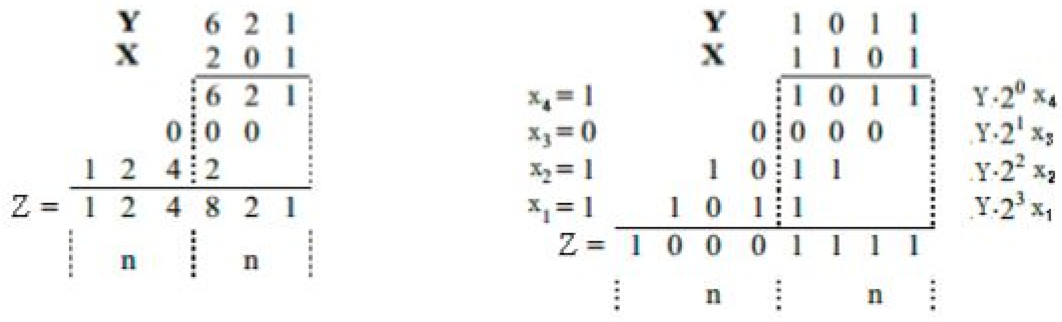
$$Z_i = Z_{i-1} + Y_i\,x_{n-i+1}$$

$$Y_i = 2\,Y_{i-1}$$

З початковим значенням і=1, $Y_0 = Y\,2^{-n}$, $Z_0 = 0$.

Перед початком множення 2-им способом, множник X записують у регістр RG2, а множене Y – в молодші розряди регістру RG3(тобто в регістрі RG3 установлюють $Y_0 = Y\,2^{-n}$). В кожному і-ому циклі множення додавання кодів RG3 і RG1 керує цифра RG2(n), а в регістрі RG3 здійснюється зсув вліво на 1-ин розряд, в результаті чого формується величина $Y_i = 2Y_{i-1}$. Оскільки сума часткових добутків у процесі множення нерухома, зсув у регістрі RG3 можна сполучити в часі з підсумовуванням (як правило, t_n > $t_з$). В цьому випадку t_i=n t_n . Завершення операції множення визначається за нульовим вмістом регістру RG2, що також приводить до збільшення швидкодії, якщо множник ненормалізований.

Наведемо приклад виконання множення за цим способом, зліва множимо два числа в десятковій формі, справа – в двійковій.



Операційна схема пристрою множення (рис.2) :

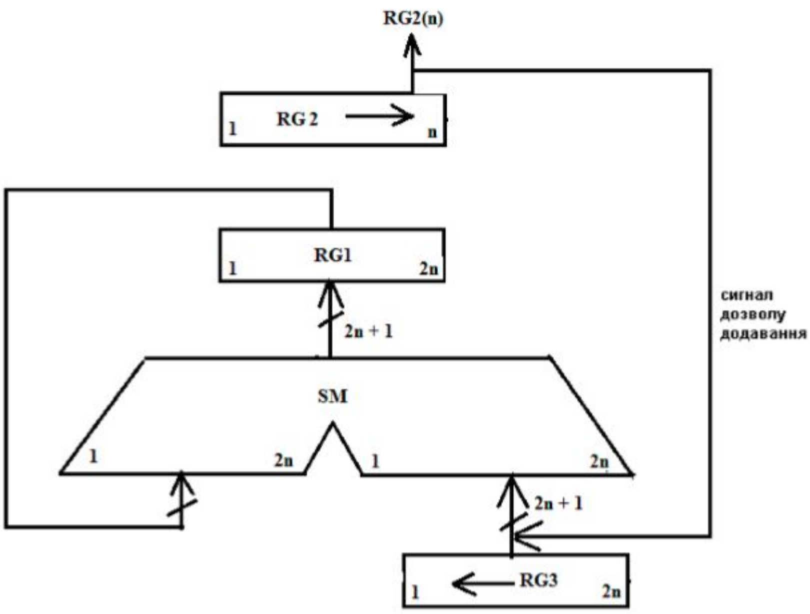


Рис.2

Алгоритм отримання результату даним способом наступний :

1) Вміст суматора(регістр RG1) обнуляється;

2) Множене RG3 множиться на наступний розряд множника;

3) Результат сумується з вмістом суматора;

4) Множене зсувається на 1 розряд вліво;

5) Множник (RG2) зсувається на 1-ин розряд вправо;

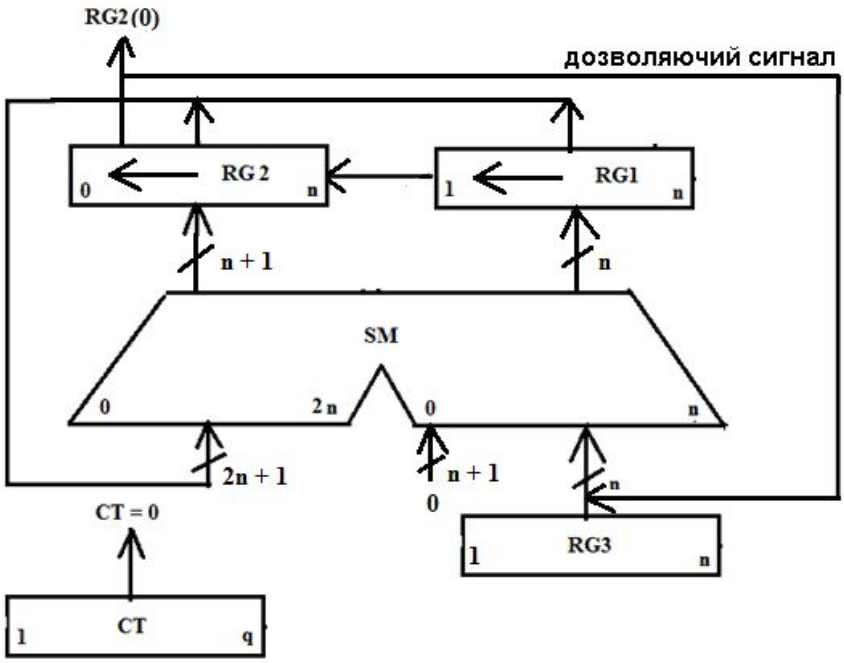
6) Пункти 2,3,4 повторюються n-1 разів;

7) Кінець циклу множення відбувається при нульовому значенні регістра RG2.

58. Множення чисел (3- спосіб)

Подаємо множення в наступному вигляді : Z =YX = ((…((0+Y2-nx1)2 + Y2–nx2)2+…+ Y2 -nxi)2 +…+Y2-nxn .

Суму часткових добутків в і-му циклі і є {1,n} можна одержати за виразом Zi = 2Zi-1 + Y2-nxi З початковими значеннями і = 1, Z0 = 0. Під час множення 3-ім способом вага молодшого розряду RG3 дорівнює 2-2n, тому код у регістрі RG3 являє собою значення Y2-n. На початку кожного циклу множення здійснюється лівий зсув у регістрах RG1 та RG2, а потім виконується додавання, яким керує RG2(0). У результаті підсумовування вмісту RG3 та RG1 може виникнути перенос у молодший розряд регістру RG2. У старшій частині суматора, на якому здійснюється підсумовування коду RG2 з нулями, відбувається поширення переносу. Збільшення довжини RG2 на один розряд усуває можливість поширення переносу в розряди множника. Після виконання n-циклів, молодші розряди добутку будуть знаходитись в регістрі RG1, старші – в RG2. Час множення третім способом визначається аналогічно першому способу і дорівнює ti = n(ti + tc). Перед початком множення X записуємо в старші розряди регістра RG2. Y записуємо в RG3. Підрахунок кількості циклів множення забезпечує лічильник СТ, відповідно до цього обирається його розрядність. Наведемо приклад виконання множення за цим способом, зліва множимо два числа в десятковій формі, справа – в двійковій.



59. Множення чисел (4- спосіб)

Множення з старших розрядів множника та зсувом множеного вправо при чому сума часткових добутків залишається нерухомою

Подаємо множення в наступному вигляді :

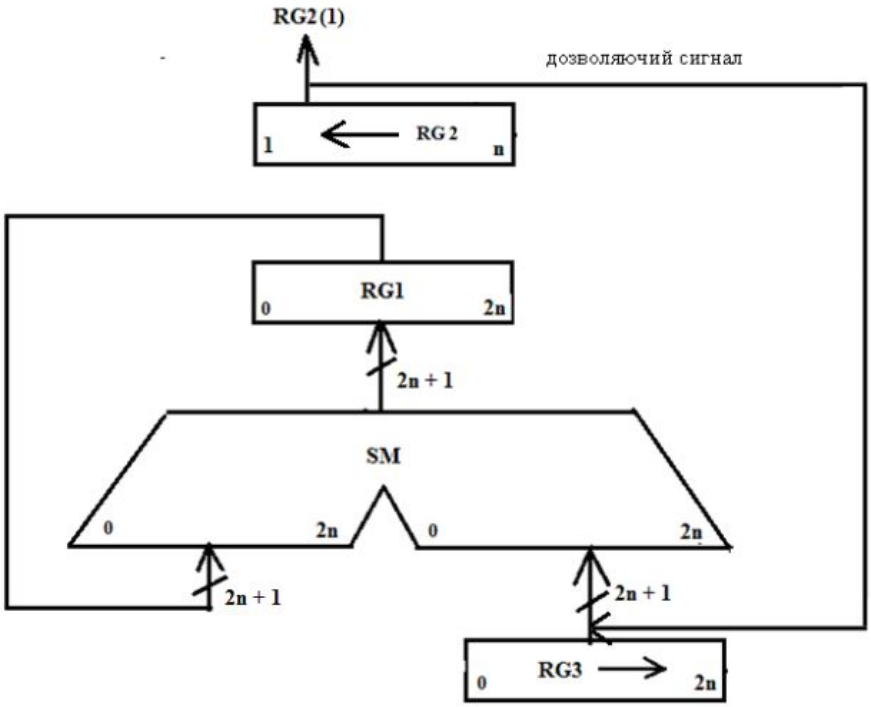
$Z = YX = ((\ldots((0+Y2^{-1}x_1) + Y2^{-2}x_2) + \ldots + Y2^{-i}x_i) + \ldots + Y2^{-n}x_n$

У цьому випадку процес множення може бути зведений до n- кратного виконання циклу Zi = Zi-1 + Yi-1xi , Yi = Yi-1 2-1 З початковими значеннями i = 1, Y0 = Y^2-1, Z0 = 0.

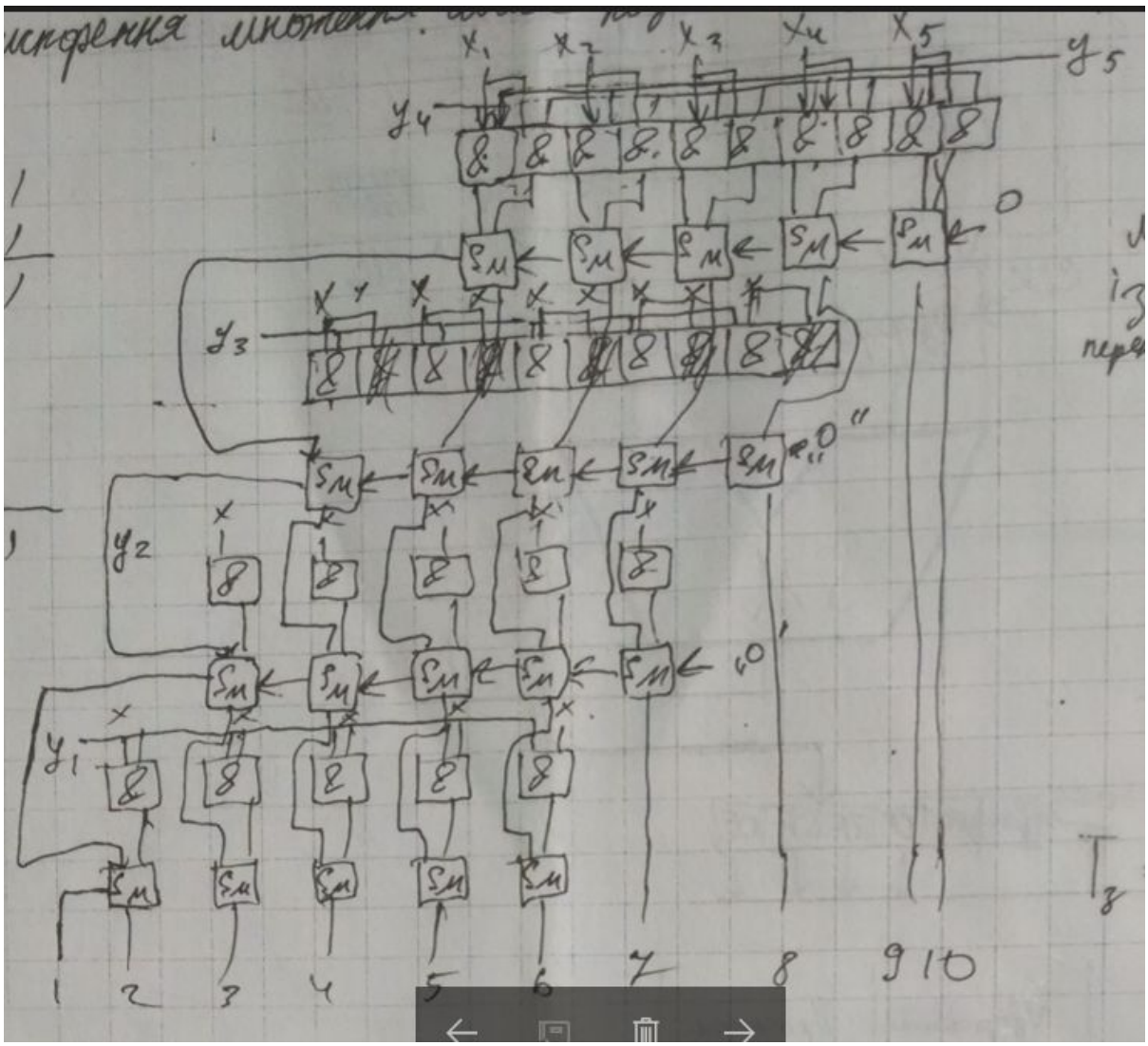
Y	6	2	1						
X	2	0	1						
	1	2	4	2					
	0	0	0	0	0				
	0	0	0	6	2	1			
	1	2	4	8	2	1			
		n			n				

Y	1	0	1	1						
X	1	1	0	1						
$x_1 = 1$	0	1	0	1	1					$Y \cdot 2^{-1}\ x_1$
$x_2 = 1$	0	0	1	0	1	1				$Y \cdot 2^{-2}\ x_2$
$x_3 = 0$	0	0	0	0	0	0	0			$Y \cdot 2^{-3}\ x_3$
$x_4 = 1$	0	0	0	0	1	0	1	1		$Y \cdot 2^{-4}\ x_4$
	1	0	0	0	1	1	1	1		
			n				n			

Множник записують у регістр RG2, а множене – в старші розряди регістру RG3(тобто в RG3 установлюють Y0 = Y2-1). У кожному циклі цифра RG2(0), що знаходиться в старшому розряді регістру RG2, керує підсумовуванням, а в RG3 здійснюється правий зсув на один розряд, що еквівалентно множенню вмісту цього регістра на 2^{-1}

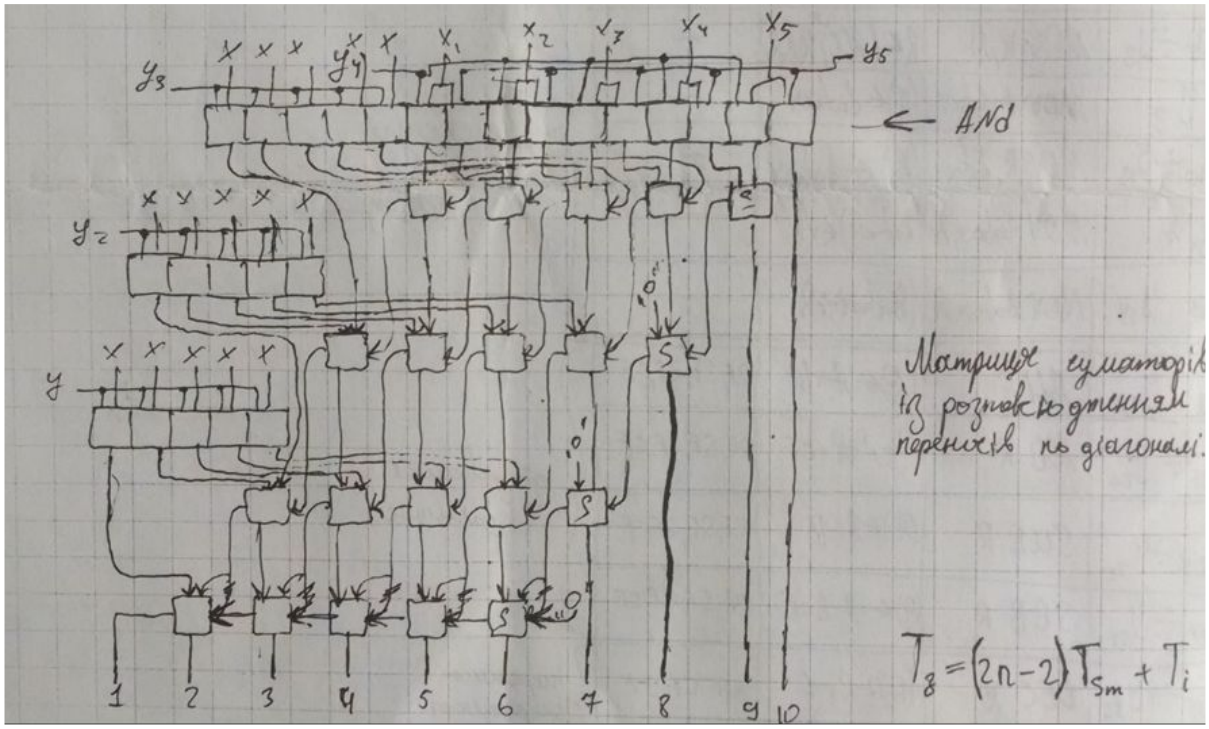

RG2(1)
дозволяючий сигнал
1
RG 2
n
RG1
0
2n
2n + 1
SM
0
2n
0
2n
2n + 1
2n + 1
RG3
0
2n

60. Прискорення множення чисел поданих паралельним кодом (схема матриці суматорів із розповсюдженням переносів по рядкам)



Затримка цієї схеми $T = (3n-4)T_{sm} + T_{and}$ де n - бітність чисел які перемножуються.

61. Прискорення множення чисел поданих паралельним кодом (схема матриці суматорів із розповсюдженням переносів по діагоналі)



матричні методи множення.

Коли множене і множник розташовані в регістрах машини, можна утворити відразу всі часткові добутки і здійснити їх одночасне додавання, використовуючи певну кількість суматорів. Узагальнена структура пристрою, що реалізує таке множення (рис. 3.6), містить: регістри РгА і РгВ, в яких зберігаються множене і множник, відповідно; блок елементів І, що забезпечує формування всіх часткових добутків; блок суматорів, у якому здійснюється одночасне додавання всіх часткових добутків.

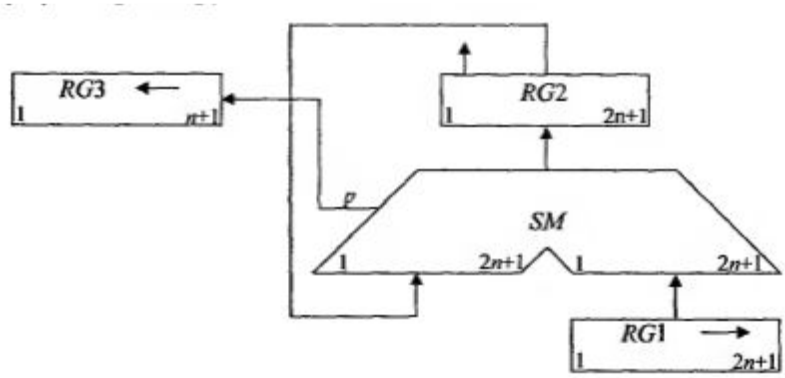
Матричні методи множення відрізняються саме організацією одночасного додавання.

Існує ряд методів множення, що засновані на додаванні груп часткових добутків з наступним об'єднанням сум разом з перенесеннями для одержання добутку. Наприклад, часткові добутки групуються по три і додаються із запам'ятовуванням перенесень за допомогою ланцюжка суматорів На кінці ланцюжка здійснюється додавання з розповсюдженням перенесень. Така роздільна обробка проміжних сум і перенесень вимагає так називаного "дерева суматорів"

Реалізація матричних методів виконання операції множення вимагає більшої кількості апаратури, ніж методів послідовного аналізу розрядів або груп розрядів множника, і дає більший виграш у часі. Однак у зв'язку зі значним розвитком мікроелектроніки обмеження щодо кількості апаратури стають усе менш суворими, тому матричні методи широко застосовують на практиці.

62. Ділення чисел без відновлення від'ємного залишку (з подвоєнням залишку)

Під час реалізації ділення даним способом, збільшується розрядність регістрів RG1, RG3 і суматор SM. Принцип побудови пристрою для ділення чисел зображенно на рис. В цьому випадку процес віднімання і додавання і зсуву можуть бути сполучені в часі. Цифра результату формується на виході переносу суматора SM(p).



Під час ділення чисел з фіксованою комою має бути передбаченна можливість фіксації переповнення розрядної сітки. Ознака переповнення виробляється, якщо в першому циклі обчислення отримана цифра результату із значенням 1.

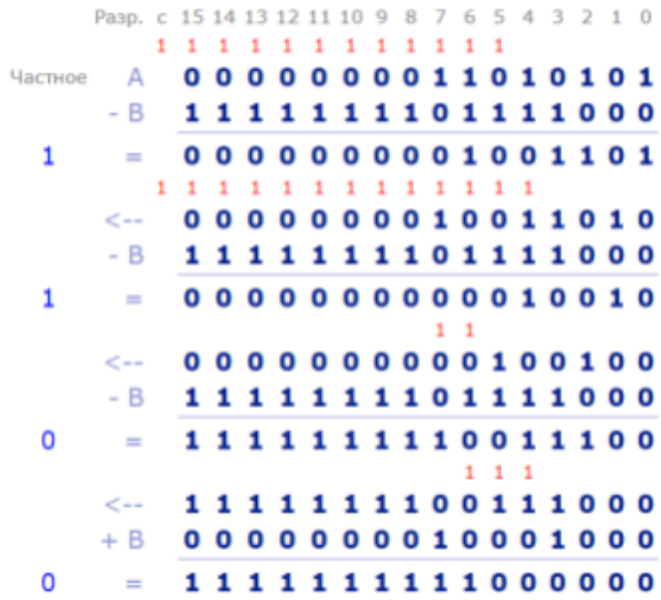
63. Ділення чисел без відновлення від'ємного залишку (з зменшенням вдічі дільника)

$$213_{10} = 11010101_2$$
$$17_{10} = 10001_2$$

A: 0000000011010101

B: 0000000010001000
$B_{доп}$1111111101111000

6) Процесс деления будет следующий:
6.1) Вычитаем из делимого **A** делитель **B** (т.е. прибавляем **-B**).
6.2) Анализируем знак полученного частичного остатка (15-й разряд). В регистр результата записываем "0" если остаток отрицательный и единицу в противном случае. Помним, что отрицательному числу соответствует наличие единицы в 15-м разряде и наоборот.
6.3) Сдвигаем частичный остаток на один разряд влево. При этом крайний правый (младший) разряд заполняется нулем, а знаковый разряд (15-й) в процессе сдвига не участвует.
6.4) Прибавляем к частичному остатку делитель **B** если остаток отрицательный либо вычитаем делитель в противном случае.
6.5) Анализируем знак полученного частичного остатка (15-й разряд). В регистр результата записываем "0" если остаток отрицательный и единицу в противном случае.
6.6) Действия описанные в пунктах 6.3-6.5 выполняем **k** раз (если k=0, то ни разу не выполняем).
Весь процесс деления выглядит следующим образом :



7) Определяем **остаток от деления**. Для этого анализируем последний частичный остаток. В нашем случае он равен **"1111111111000000"**.

7.1) Анализируем знак остатка (15-й разряд). В знаковом разряде содержится единица, значит требуется коррекция остатка. Для коррекции прибавим к нему делитель **B**.

7.2) Так как в процессе деления частичные остатки были сдвинуты 3 раза влево, то для получения верного значения последний полученный остаток необходимо сдвинуть 3 раза вправо (вернуть на место). После сдвига имеем:
0000000000001001
8) Определяем **знак результата**. Если знаки исходных операндов одинаковы, то результирующее частное положительно и наоборот. В нашем случае знаки совпадают, следовательно результирующее частное положительно.

Ответ: $\mathbf{11010101}_2$: $\mathbf{10001}_2$ = $\mathbf{1100}_2$ и $\mathbf{1001}_2$ в остатке.

или в десятичной системе счисления: 213_{10} : 17_{10} = 12_{10} и 9_{10} в остатке.
http://reshinfo.com/primer_delenije1.php

64. Додавання чисел з плаваючою комою.

1)A>0 ;B>0

A+B(зк) = A+B

2)A>0 ;B<0 ;A+B >0

A(зк)+B(зк) = A+2^(n+2) - 2^k +B=(A+B)+2^(n+2)-2^(-k)

корекція +2^k

Пр: A= 00.10101 (21/32) B=11.10110 (-9/32)

00.10101+

11.10110

100.01011+

00.00001

00.01100 (12/32)

3) A>0; B<0 ; |A| < |B|; A+B<0

A(зк) + B(зк) = A+2^(n+2) - 2^k + B = A+B + 2^(n+2)+2^(-k)=[A+B](зк)

Пр :

00,01001

11,01010

11,10011

4) A<0; B<0

A(зк) + B(зк) = A+B+2^(n+2)-2^(-k) = [A+B](зк) + 2^(n+2) - 2^(-k)

A=0.10101(-21) A(зк)= 11.01010+

B=0.01001 (-9) B(зк) = 11.10110

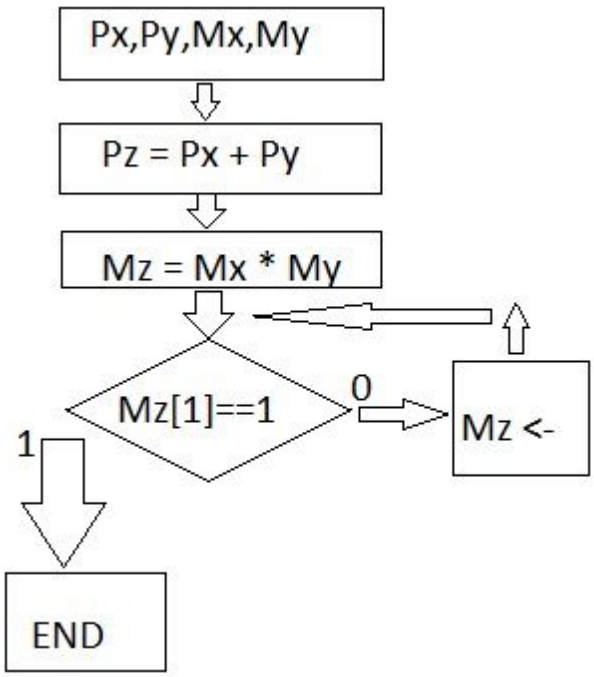
111.00000+

00.00001

11.00001

11.11110(-30)

65. Множення чисел з плаваючою комою.



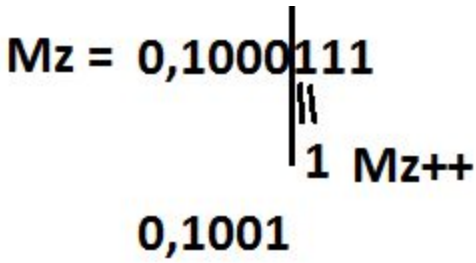
66. Ділення чисел з плаваючою комою.

X=2^Px * Mx

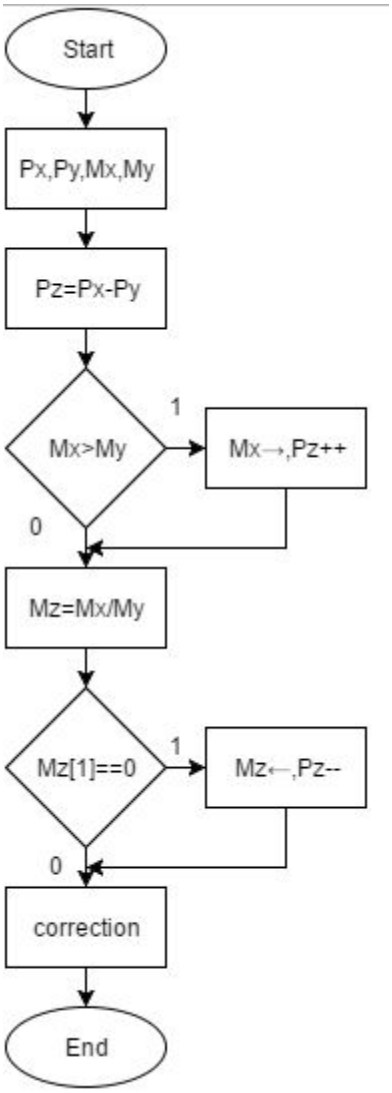
Y=2^Py * My

Z=2^Pz * Mz = X/Y

1) Pz=Px-Py
2) if Mx>My{Mx→, Pz++}
3) Mz=Mx/My
4) нормализация(if Mz[1]=0 {Mz ←, Pz--})



5) корекция


Start
Px,Py,Mx,My
Pz=Px-Py
Mx>My
1
Mx→,Pz++
0
Mz=Mx/My
Mz[1]==0
1
Mz←,Pz--
0
correction
End