**Экзамен ЦУиМП**

Оглавление

[1. Предметодье (как предисловие) 4](#_Toc131316580)

[2 Комбинационная логика 5](#_Toc131316581)

[2.1 Цифровые элементы (О, ТИ, АВ) 6](#_Toc131316582)

[2.2 Арифметические операции с числами 7](#_Toc131316583)

[2.3 Вещественные числа. Хранение вещественного числа в компьютере 8](#_Toc131316584)

[2.4 Перевод чисел 9](#_Toc131316585)

[2.5 Таблицы истинности и алгебраические выражения 10](#_Toc131316586)

[2.6 Карты Карно 11](#_Toc131316587)

[2.7 Метод Квайна 12](#_Toc131316588)

[2.8 Базис 2И-НЕ 13](#_Toc131316589)

[2.9 Базис 2ИЛИ-НЕ 14](#_Toc131316590)

[2.10 Шифратор (О, ТИ, АВ) 15](#_Toc131316591)

[2.11 Дешифратор 16](#_Toc131316592)

[2.12 ССИ 17](#_Toc131316593)

[2.13 Приоритетный шифратор (О?, ТИ) 18](#_Toc131316594)

[2.14 Дешифратор матричного типа 19](#_Toc131316595)

[2.15 Каскадное включение дешифраторов 20](#_Toc131316596)

[2.16 Мультиплексор (О, ТИ, АВ) 21](#_Toc131316597)

[2.17 Демультиплексор (О, ТИ, АВ) 22](#_Toc131316598)

[2.18 Мажоритарный элемент (О?, ТИ, АВ) 23](#_Toc131316599)

[2.19 Сумматор и полусумматор 24](#_Toc131316600)

[2.20 Полувычитатель и вычитатель 25](#_Toc131316601)

[2.21 Умножители. Умножитель 2х2, 3х3, 4х4 26](#_Toc131316602)

[2.22 Компаратор (О, ТИ, АВ) 27](#_Toc131316603)

[2.23 Последовательный сдвиговый регистр (Barrel shifter) 28](#_Toc131316604)

[2.24 АЛУ 29](#_Toc131316605)

[2.25 Статический сбой (static hazart) 30](#_Toc131316606)

[2.26 Вопросы на подумать 31](#_Toc131316607)

[2.26.1 ИСКЛ ИЛИ на 4 элементах (базис 2И-НЕ). ИСКЛ ИЛИ в базисе 2ИЛИ-НЕ 32](#_Toc131316608)

[2.26.2 Вычитатель на сумматоре 33](#_Toc131316609)

[2.26.3 Сложите 4-5 одноразрядных чисел. Есть восьмиразрядное число, определить количество единиц в нём 34](#_Toc131316610)

[2.26.4 Компаратор на сумматорах. 35](#_Toc131316611)

[2.26.5 Вариант 1 РК1 36](#_Toc131316612)

[2.26.6 Вариант 3 РК1 37](#_Toc131316613)

[2.26.7 Вариант 4 РК1 38](#_Toc131316614)

[2.26.8 Вариант 7 РК1 39](#_Toc131316615)

[3 Последовательстная логика (логика с памятью) 40](#_Toc131316616)

[3.1 RS-триггер, статический, асинхронный, одноступенчатый 41](#_Toc131316617)

[3.2 RS-триггер, синхронный, статический, одноступенчатый 42](#_Toc131316618)

[3.3 D-триггер (D flip-flop), синхронный, статический, одноступенчатый 43](#_Toc131316619)

[3.4 Динамический D-триггер, синхронный, двухступенчатый 44](#_Toc131316620)

[3.5 T-триггер (счётный) 45](#_Toc131316621)

[3.6 Асинхронный счётчик на D-триггерах. 46](#_Toc131316622)

[3.7 Динамический RS-триггер 47](#_Toc131316623)

[3.8 Вычитающий счётчик (асинхронный) 48](#_Toc131316624)

[3.9 Реверсивный счётчик 49](#_Toc131316625)

[3.10 Счётчик с предустановленными значениями 50](#_Toc131316626)

[3.11 Кольцевой счётчик (синхронный) 51](#_Toc131316627)

[3.12 Кольцевой счётчик Джонсона 52](#_Toc131316628)

[3.13 Кольцевой счётчик (ходит только одна единица) 53](#_Toc131316629)

[3.14 Синхронный счётчик вариант 1 54](#_Toc131316630)

[3.15 Синхронный счётчик вариант 2 55](#_Toc131316631)

[3.16 Регистры. Последовательно-параллельные 56](#_Toc131316632)

[3.17 Регистры. Параллельно-последовательные 57](#_Toc131316633)

[3.18 Регистры параллельно-параллельные 58](#_Toc131316634)

[3.19 Регистры. Циклический регистр. Выбор направления. Предустановка значений. 59](#_Toc131316635)

[3.20 Динамический одноступенчатый D-триггер 60](#_Toc131316636)

[3.21 Динамический D-триггер с асинхронным сбросом, установкой 61](#_Toc131316637)

[3.22 JK-триггер, синхронный, статический 62](#_Toc131316638)

[3.23 Динамический JK-триггер 63](#_Toc131316639)

[3.24 Динамический JK-триггер на D-треггерах 64](#_Toc131316640)

[3.25 T-триггер на JK-триггере 65](#_Toc131316641)

[3.26 Синхронный счётчик на JK-триггере 66](#_Toc131316642)

[3.27 Сброс триггера. Асинхронные, синхронные входы. 67](#_Toc131316643)

[3.28 Задачи на подумать. 68](#_Toc131316644)

[3.28.1 Счётчик в базисе 2И-НЕ 69](#_Toc131316645)

[3.28.2 Имеется кольцевой счётчик. Как загнать «1» или несколько «1» и двигать их по кругу? 70](#_Toc131316646)

# Предметодье (как предисловие)

*Я никогда не учу своих учеников. Я только даю условия, при которых они могут сами учиться*

(Эйнштейн)

*Без примеров невозможно ни правильно учить, ни успешно учиться*

(Пацанский паблик ВК)

Целью данной методы не является создать методу ради методы, хочется искренне всё понять, а это возможно только в том случае, если сделать всё самому. Поэтому при создании вашей части методы старайтесь «научить» читателя чему-то, заставить его попробовать реализовать что-то самостоятельно, ведь только в этом случае можно проникнуться всей красотой цифровой электроники и микропроцессоров, и сделать шаг к торжеству автоматизации рутинных задач и процессов.

Обозначения и сокращения:

* О – обозначение
* ТИ – таблица истинности
* АВ – алгебраические выражения
* ВД – временная диаграмма.

# Комбинационная логика

Комбинационная логика — это раздел цифровой логики, который занимается анализом и проектированием комбинационных схем.

Комбинационные схемы — это логические схемы, в которых выходные значения зависят только от текущих входных значений без учета предыдущих состояний или последовательности входных сигналов, что отличает их от последовательностных схем, в которых следующее состояние зависит от предыдущего.

Комбинационная логика используется для создания цифровых схем, которые выполняют определенные логические операции над входными данными и производят соответствующие выходные результаты. Эти схемы широко применяются в различных устройствах, включая компьютеры, микропроцессоры, телефоны, автомобильные системы и многие другие электронные устройства.

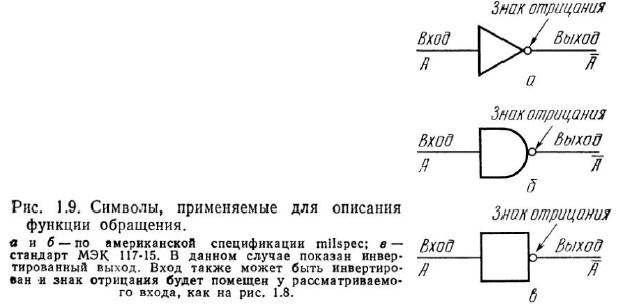
Основная цель комбинационной логики — это создание схем, которые могут выполнять определенные логические операции, такие как логическое И, ИЛИ, НЕ, XOR и другие, и комбинировать эти операции для получения желаемых выходных результатов. Комбинационные схемы также могут выполнять арифметические операции, сравнения, кодирование и декодирование информации, управление данными и т. д.

Преимущества использования комбинационной логики включают быстродействие, надежность, компактность и низкую стоимость. Она позволяет эффективно решать задачи обработки информации и управления в цифровом виде.

## Цифровые элементы (О, ТИ, АВ)

В системе МЭК (Международная Электротехническая комиссия) используется описание логической функции с помощью квадратной рамки, в которой находится символ данной функции.

1. Инвертор – думаю, здесь комментарии не нужны...ведь да? В алгебраических выражениях инверсия отображается чертой над логическим элементом/логической операцией: .



|  |  |
| --- | --- |
|  |  |
| 0 | 1 |
| 1 | 0 |

2. ИЛИ – «что-нибудь или всё». Пример обозначения приведён для 3ИЛИ, но это не имеет значения, так как мы можем сделать хоть 111111ИЛИ, все они будут собраны через элементы 2ИЛИ.

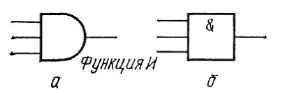
В алгебраических выражениях записывается следующим образом:



|  |  |  |
| --- | --- | --- |
| ­ |  |  |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

3. И – «всё или ничего». Пример обозначения приведён для 3И, но это не имеет значения, так как мы можем сделать хоть 111111И, все они будут собраны через элементы 2И.

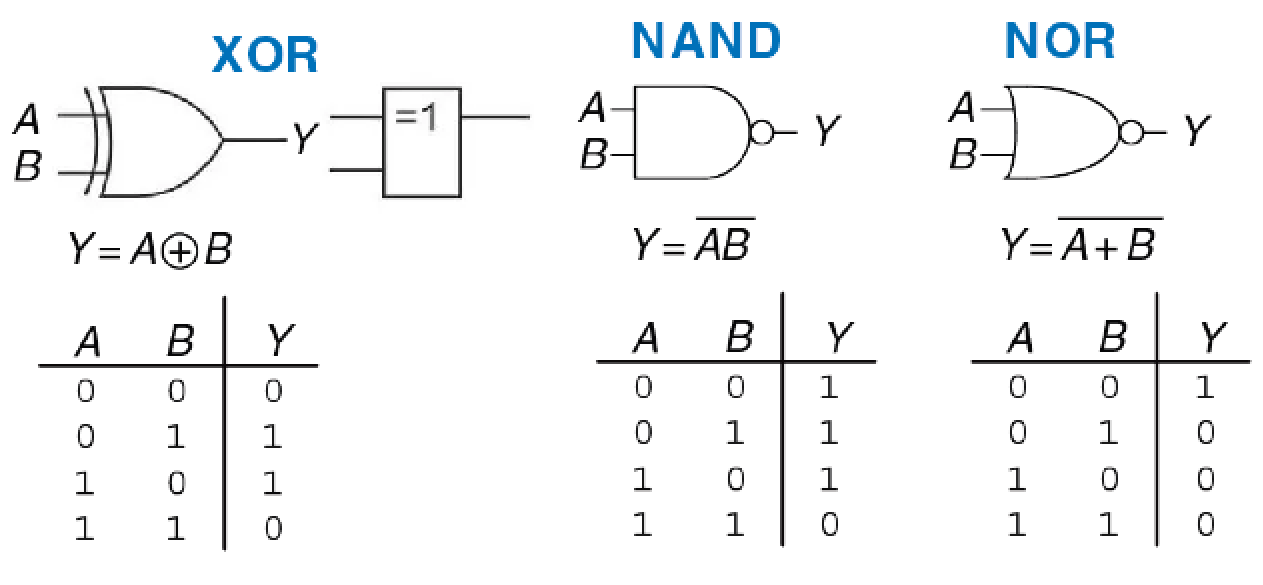
В алгебраических выражениях записывается следующим образом:



|  |  |  |
| --- | --- | --- |
| ­ |  |  |
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

4. XOR («Исключающее ИЛИ»), NAND (2И-НЕ), NOR(2ИЛИ-НЕ)

Из элементов 1-3 можно собрать и выдумать множество других элементов, примеры наиболее распространённых и их таблицы истинности приведены на рисунке ниже:



Исключающее ИЛИ в виде алгебраического выражения можно представить несколькими способами:

Последний вариант может использоваться в картах КАРНО, когда имеет место следующая таблица:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| b | 1 | 0 |
|  | 0 | 1 |

= – Исключающее ИЛИ-НЕ.

## Арифметические операции с числами

что такое позиционные и непозиционные коды? Прямой, обратный, доп код, перевод из одного в другой, для каких чисел какие коды существуют, умножение, сложение, вычитание бинарых чисел.

## Вещественные числа. Хранение вещественного числа в компьютере

Перевод вещественных чисел в бинарную систему счисления, хранение числа в компьютере (Задано число, нужно с точностью до Х знаков перевести в двоичную систему счисления. Как узнать число двоичных чисел после запятой, которых будет достаточно для заданной точности?)

## Перевод чисел

Hex, bin, dec. Почему hex код удобен?

## Таблицы истинности и алгебраические выражения

Основные законы алгебры логики (минимизации).

Комбинационные схемы — это логические схемы, в которых выходные значения зависят только от текущих входных значений без учета предыдущих состояний или последовательности входных сигналов. Комбинационная схема есть чёткая зависимость между входом и выходом.

Известно два способа задания логических функций: с помощью формулы и с помощью таблицы истинности. По формуле легко составляется таблица. На практике при конструировании различных электронных устройств часто возникает обратная задача – от таблицы истинности перейти к формуле, чтобы на ее основе построить функциональную схему.

Таблицу истинности можно записать в виде алгебраического выражения. Для этого существует 2 способа:

1) СДНФ, простыми словами, - «сумма произведений», в которых все произведения содержат ВСЕ логические переменные;

2) СКНФ, простыми словами, - «произведение сумм», в которых все суммы содержат ВСЕ логические переменные.

А если не простыми, то вот….

***Элементарной конъюнкцией*** называется конъюнкция нескольких переменных, взятых с отрицанием или без отрицания, причем среди переменных могут быть одинаковые (b).

***Элементарной дизъюнкцией*** называется дизъюнкция нескольких переменных, взятых с отрицанием или без отрицания, причем среди переменных могут быть одинаковые ().

Всякую дизъюнкцию элементарных конъюнкций назовем ***дизъюнктивной нормальной формой*** (ДНФ).

Всякую конъюнкцию элементарных дизъюнкций назовем ***конъюнктивной нормальной формой*** (КНФ).

***Совершенной дизъюнктивной нормальной формой*** (СДНФ) называется ДНФ, в которой нет одинаковых элементарных конъюнкций и все конъюнкции состоят из одного и того же набора переменных, в который каждая переменная входит только один раз (возможно, с отрицанием).

***Совершенной конъюнктивной нормальной формой*** (СКНФ) называется КНФ, в которой нет одинаковых элементарных дизъюнкций и все дизъюнкции состоят из одного и того же набора переменных, в который каждая переменная входит только один раз (возможно, с отрицанием).

**Алгебраические выражения по таблице истинности**

Построим случайную таблицу истинности:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |

Запишем алгебраические выражения:

1) СДНФ строится по «1», то есть находим «1» в столбце «» и записываем последовательность соответствующих ей (при этом, если в столбце любого из x стоит «1», то просто записываем , а если «0», то , то есть «не 0» = «1») в виде произведения. Данное действие повторяем для каждой «1» и складываем получившиеся элементарные конъюнкции:

макстерм

2) СКНФ строится по «0», то есть находим «0» в столбце «» и записываем последовательность соответствующих ей (при этом, если в столбце любого из x стоит «0», то просто записываем , а если «1», то , то есть «не 1» = «0») в виде суммы. Данное действие повторяем для каждой «1» и перемножаем получившиеся элементарные дизъюнкции:

минтерм

**Представим обратную задачу: записать таблицу истинности по алгебраическому выражению**

Запишем случайное алгебраическое выражение:

1) СДНФ

Всё просто: берём наши конъюнкции и заполняем строчки таблицы. Записав 1-ую, таблица имеет вид:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 |  |
| 1 | 0 | 0 |  |
| 0 | 1 | 0 |  |
| 1 | 1 | 0 |  |
| 0 | 0 | 1 |  |
| 1 | 0 | 1 |  |
| 0 | 1 | 1 |  |
| 1 | 1 | 1 | 1 |

2-ую:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 0 |  |
| 0 | 1 | 0 |  |
| 1 | 1 | 0 |  |
| 0 | 0 | 1 |  |
| 1 | 0 | 1 |  |
| 0 | 1 | 1 |  |
| 1 | 1 | 1 | 1 |

3-ую и 4-ую:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 0 |  |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 |  |
| 0 | 0 | 1 |  |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 1 |  |
| 1 | 1 | 1 | 1 |

Оставшиеся строчки заполняем «0»:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

2) СКНФ

Всё просто: берём наши дизъюнкции и заполняем строчки таблицы. Записав 1-ую, таблица имеет вид:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 |  |
| 0 | 1 | 0 |  |
| 1 | 1 | 0 |  |
| 0 | 0 | 1 |  |
| 1 | 0 | 1 |  |
| 0 | 1 | 1 |  |
| 1 | 1 | 1 |  |

2-ую:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 |  |
| 0 | 1 | 0 |  |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 |  |
| 1 | 0 | 1 |  |
| 0 | 1 | 1 |  |
| 1 | 1 | 1 |  |

3-ую и 4-ую:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 |  |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 1 |  |
| 0 | 1 | 1 |  |
| 1 | 1 | 1 |  |

Оставшиеся строчки заполняем «1»:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

**Основные законы минимизации**

1. Коммутативный (переместительный)

2. Ассоциативный (сочетательный)

3. Дистрибутивность (распределительный)

Доказательство последней строчки:

4. Теорема Де Моргана

– штрих Шефера

*–* стрелка Пирса

5. Закон поглощения

6. Склеивание

7. Соотношение, с помощью которого строится ИСКЛЮЧАЩЕЕ-ИЛИ на 4-ёх элементах

Доказательство:

Полезные выражения:

## Карты Карно

Причём тут код Грэя? Перевод бинарных чисел в код Грэя и в обратную сторону. Принцип составления карт Карно для СКНФ и СДНФ. Как можно объединять 1 (0) в ячейках? Что даст диагональное расположение 1 (0)?

## Метод Квайна

Принцип метода Квайна основан на использовании логических законов и тождеств для упрощения алгебраических выражений. Основная идея заключается в том, чтобы преобразовать выражение в эквивалентное, более простое выражение, используя логические свойства операций и логических законов.

Разберём последовательность операций метода на КНФ и ДНФ алгебраического выражения:

макстерм

* ДНФ

В 1 колонку таблицы запишем все макстермы. Во второй и последующих колонках с помощью операции склеивания будем уменьшать количество выражений до того момента, когда больше не сможем сократить.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 |  | 1-2: (1’) | 1’-2’: (1’’) |  |
| 2 |  | 1-5: (2’) | 3’-4’: c (2’’) |  |
| 3 |  | 2-3: (3’) | (3’’) |  |
| 4 |  | 2-5: (4’) |  |  |
| 5 |  | 3-5: (5’) |  |  |
|  |  | (6’) |  |  |

Далее запишем полученные в последнем столбце выражения в первый столбец новой таблицы. «Заголовками» остальных столбцов будут все изначальные макстермы. Ставим Х в ячейке, когда в макстерме содержится полученное упрощенное выражение из первого столбца. Сначала выбираем колонки, в которых стоит всего один Х. Так как таких колонов для из первого столбца много, то целесообразно выбрать ту, макстерм которой наипростейший, то есть последний столбец. Остальные колонки с Х, встречающиеся на пути при движении справа налево выбранного Х (показано черной стрелкой), вычеркиваются. Упрощённое выражение составляется из выражений первого столбца: если в строке имеется Х, то выписываем макстерм в конечное выражение

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | X | X | X |  | X |
|  |  |  |  | X |  |

* КНФ

Последовательность действий не изменяется и все операции осуществляются аналогичным образом только для минтермов

минтерм

|  |  |  |
| --- | --- | --- |
| 1 |  | 1-3: |
| 2 |  | 2-3: |
| 3 |  | 3-4: |
| 4 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | X |  | X |  |
|  |  | X | X |  |
|  |  |  | X | X |

Разберём ещё один случай:

|  |  |  |
| --- | --- | --- |
| 1 |  | 1-2: (1’) |
| 2 |  | 1-3: (2’) |
| 3 |  | 3-4: (3’) |
| 4 |  | 4-5: (4’) |
| 5 |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | X | X |  |  |  |
|  | X |  | X |  |  |
|  |  |  | X | X |  |
|  |  |  |  | X | X |

Таким образом, «живыми» остались все строчки, но в одном из столбцов имеется несколько «Х», и они никем не «убиваются», поэтому имеем два варианта записи упрощённого алгебраического выражения:

## Базис 2И-НЕ

Зачем нужны базисы? Основные цифровые элементы в базисе 2И-НЕ. Перевод АВ в базис 2И-НЕ.

## Базис 2ИЛИ-НЕ

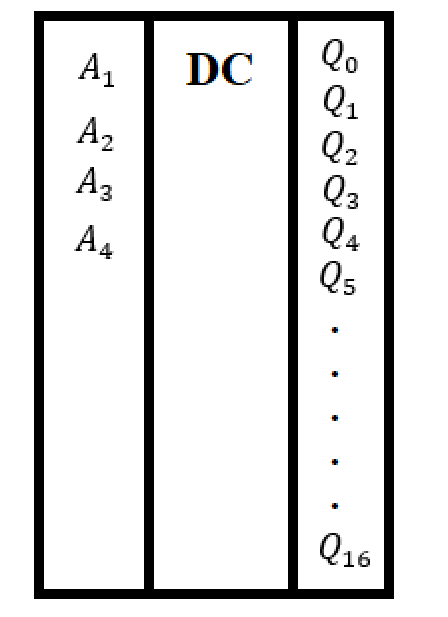
Зачем нужны базисы? Основные цифровые элементы в базисе 2ИЛИ-НЕ. Перевод АВ в базис 2ИЛИ-НЕ.

## Шифратор (О, ТИ, АВ)

Что это и зачем нужно? Можно ли использоваться комбинацию входных сигналов с несколькими 1? Обобщённая схема. Схема в базисе 2ИЛИ-НЕ, 2И-НЕ

## Дешифратор

Дешифратор (DC – decoder) – цифровое устройство, предназначенное для преобразования двоичного кода в десятичный. Обозначается следующим образом:



Количество входов и выходов дешифратора связаны следующим соотношением:

, где n – число выходов, k – число входов

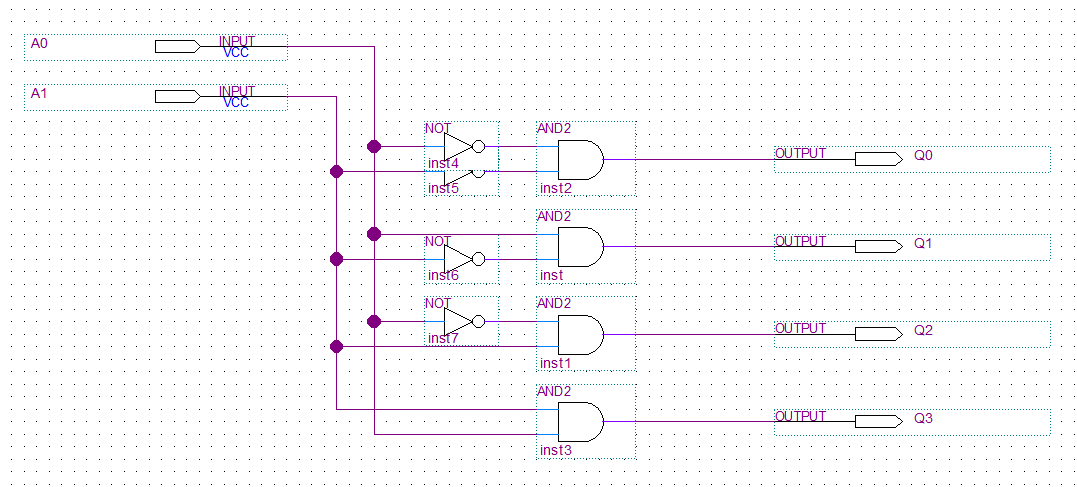
Логика работы: при подаче на входы двоичного числа на выходе с десятичным номером, соответствующим входному двоичному числу в десятичной системе, устанавливается логическая «1».

Составим таблицу истинности для дешифратора с 2-мя входами:

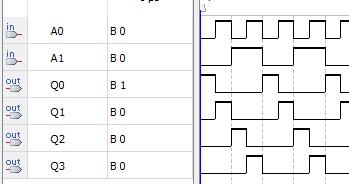
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

Из неё, очевидно:

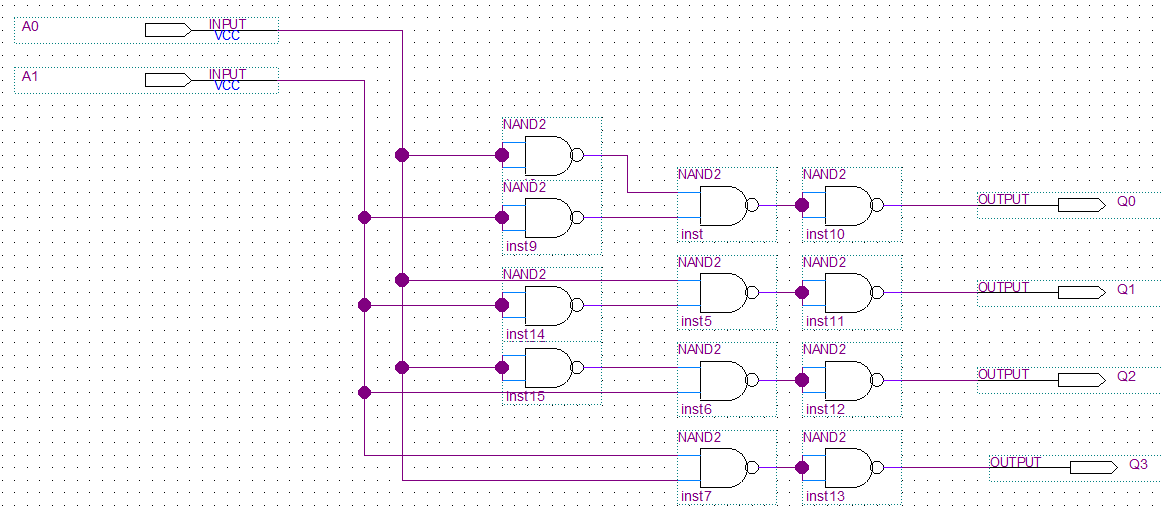
Соответственно схема имеет вид:

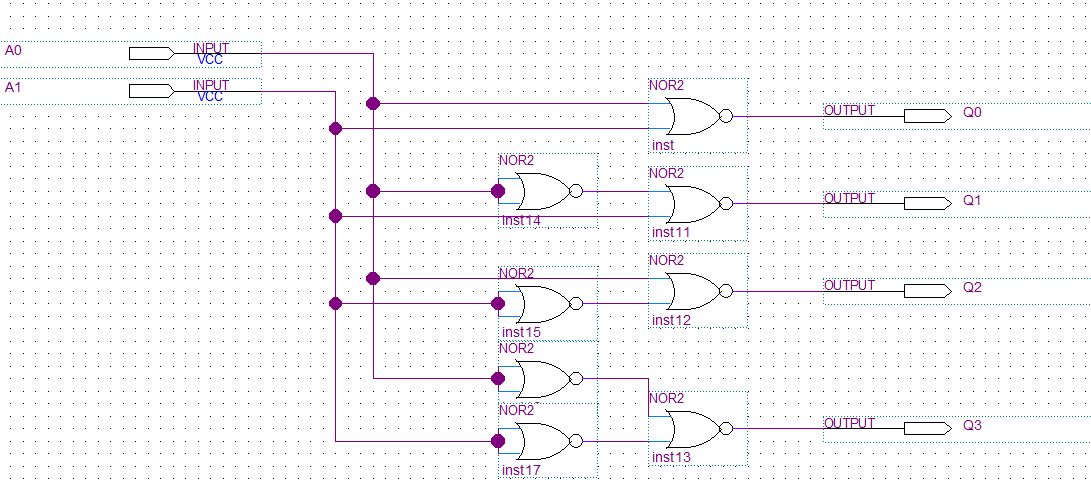


Временная диаграмма:



Представление схемы в базисе 2И-НЕ, 2ИЛИ-НЕ не представляет трудностей. Сделаем это через выражения и затем представим схему:





## ССИ

Устройство, ТИ, Схемы с ОК и ОА. Пример составления выражения для одного индикатора, его минимизации и перевод в базис (любой), схема для индикатора.

## Приоритетный шифратор

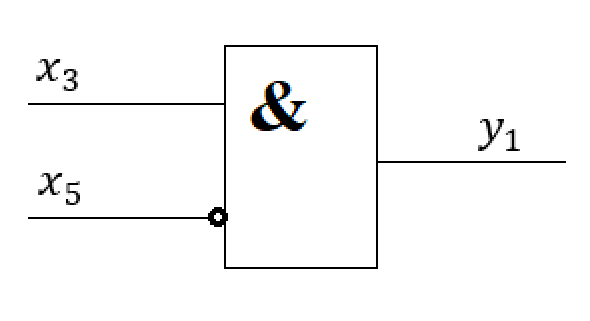
Приоритетный шифратор отличается от шифратора наличием дополнительной логической схемы выделения активного уровня старшего входа для обеспечения условия работоспособности шифратора (только один уровень на входе активный). Уровни сигналов на остальных низших входах схемой игнорируются. То есть говоря простым языком, если, например, на имеем логическую «1», то для нас не имеет значения, что находится на . Таблица истинности имеет следующий вид:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| X | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| X | X | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| X | X | X | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| X | X | X | X | X | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| X | X | X | X | X | X | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| X | X | X | X | X | X | X | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| X | X | X | X | X | X | X | X | 1 | 0 | 0 | 0 | 0 | 1 |
| X | X | X | X | X | X | X | X | X | 1 | 1 | 0 | 0 | 1 |

Х – любое состояние

Схема приоритетного дешифратора строится по следующим принципам:

* При получении высших комбинаций (например, 1010 для ) нас не интересуют низшие входы (), а имеют значения только высшие (,);
* Для исключения влияния входов, к примеру на вход , необходимо при построении схем для «выключать» их, когда на находится логическая «1». То есть, например, имеем помимо логической «1» на логическую «1» на . Тогда, на выходе может образоваться «1» из-за влияния , но если мы применим, например, элемент 2И, то мы исключим такое поведение:



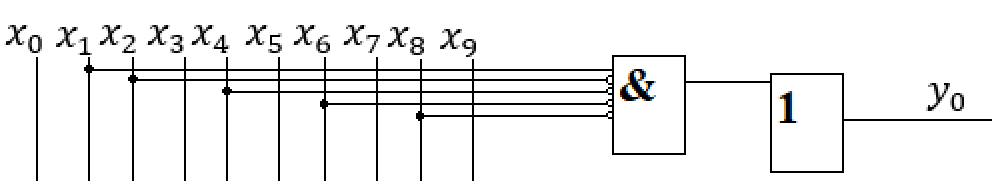
Таким образом, при появлении «1» на не имеет значения уровень , ведь он никак не повлияет на .

Соберём схему приоритетного шифратора для нашей таблицы истинности:

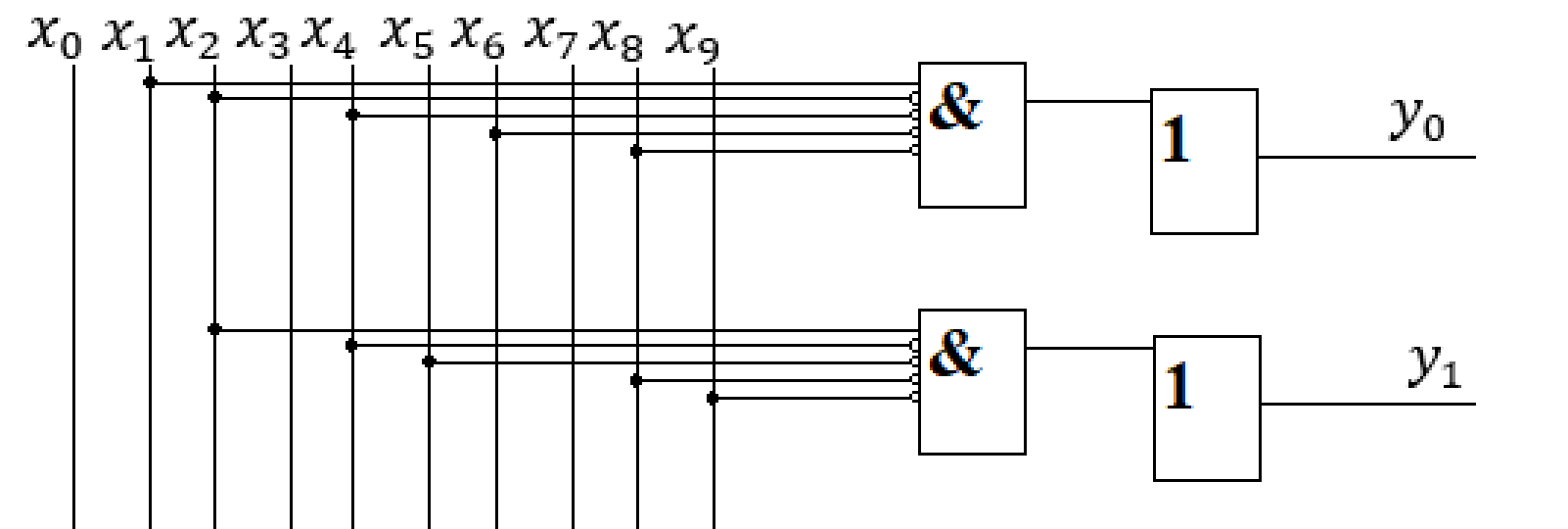
* При «1» на на находятся нули, поэтому схема не нужна
* При «1» на на комбинация 1000, то есть необходима «1» на . Нужна? Ну пожалуйста:



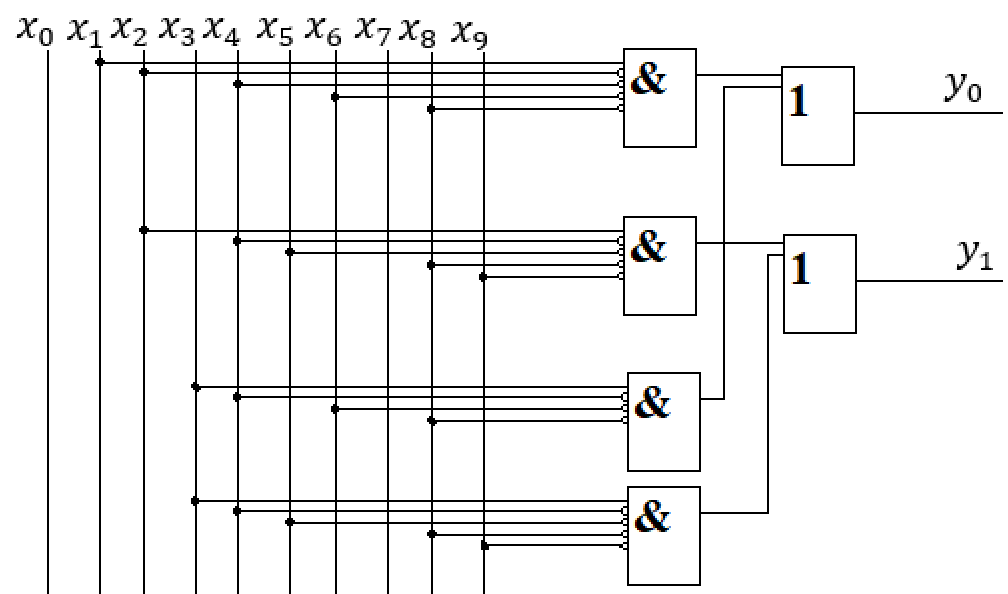
Но в таком случае «1» будет появляться и в тех случаях, когда «1» на и т.д., а её там быть не должно, ведь вход является низшим и значение на нём не должно влиять на высшие входы. Поэтому просто будем исключать влияние на , если на входах находится «1». Почему именно эти входы? Потому что в их выходных комбинациях отсутствует «1» на (смотреть таблицу истинности), когда на входах нам не нужно блокировать влияние «1» с , потому что в их выходных комбинациях и так есть «1» на . Таким образом, элемент для имеет следующий вид:



* Точно по таким же соображениям собираем схему для :

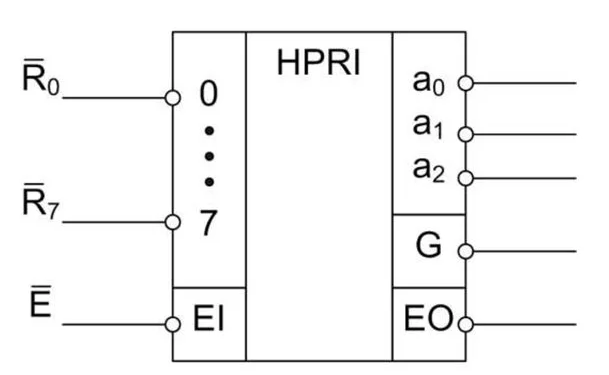


* :



Полная схема приведена в «priority\_encoder.bcd»

P.s. как обозначается - не нашёл сто процентной информации, но очень часто встречается на различным сайтах и картинкой с подписью HPRI:



## Дешифратор матричного типа

Что это, зачем и схема

## Каскадное включение дешифраторов

## Мультиплексор (О, ТИ, АВ)

Что это, зачем? Обобщённая схема, схема в базисах 2И-НЕ, 2ИЛИ-НЕ

## Демультиплексор (О, ТИ, АВ)

Что это, зачем? Обобщённая схема, схема в базисах 2И-НЕ, 2ИЛИ-НЕ

Демультиплексор – цифровое устройство, позволяющее передавать информацию с единственного информационного входа на один из выходов.

Таким образом, демультиплексор (DMX) имеет один вход для подачи информации, адресные входы для коммутации каналов, выходы. Количество адресных входов и выходов демультиплексора связны соотношением:

, где n – число выходов DMX, k – число адресных входов

Логика работы: при подаче на адресные входы двоичного числа на выходе с десятичным номером, соответствующим входному адресному двоичному числу в десятичной системе, устанавливается логический уровень информационного входа «D».

Обозначение:



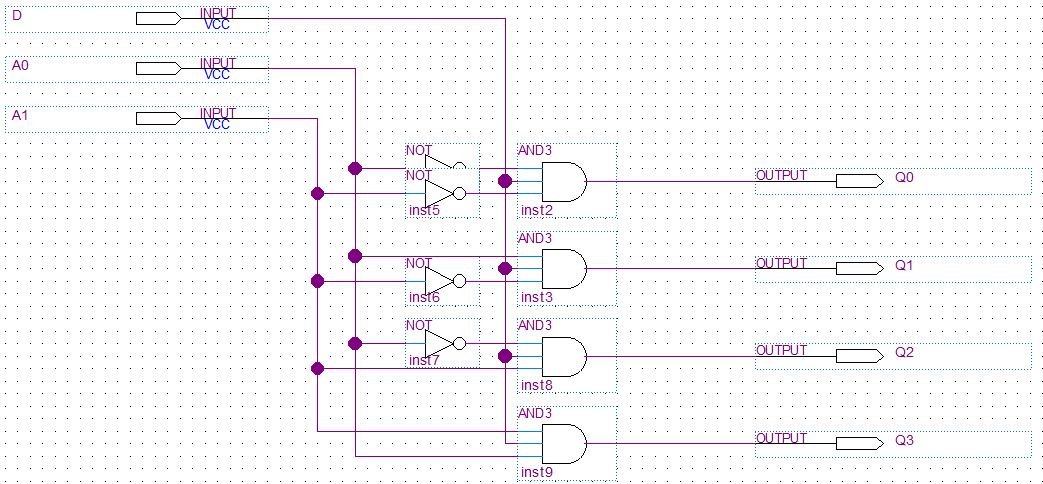
Если вы уже успели прочесть методичку с самого начала, то наверняка испытываете некоторое чувство дежавю на счёт прочитанной информации. Не волнуйтесь, всё хорошо! Причина этому следующая: демультиплексор = дешифратор + информационный вход «D». Какие-либо ещё отличия, кроме смысла применения, отсутствуют. Вот так вот….

Составим таблицу истинности для демультиплексора с 2-мя адресными входами:

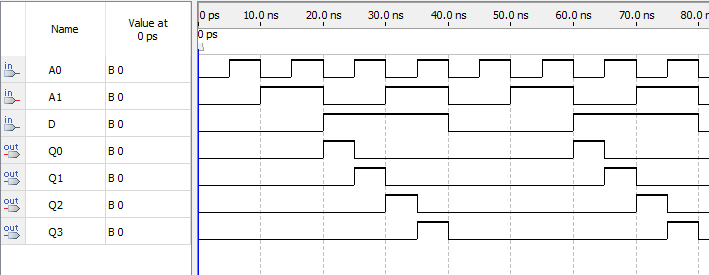
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Из неё, очевидно:

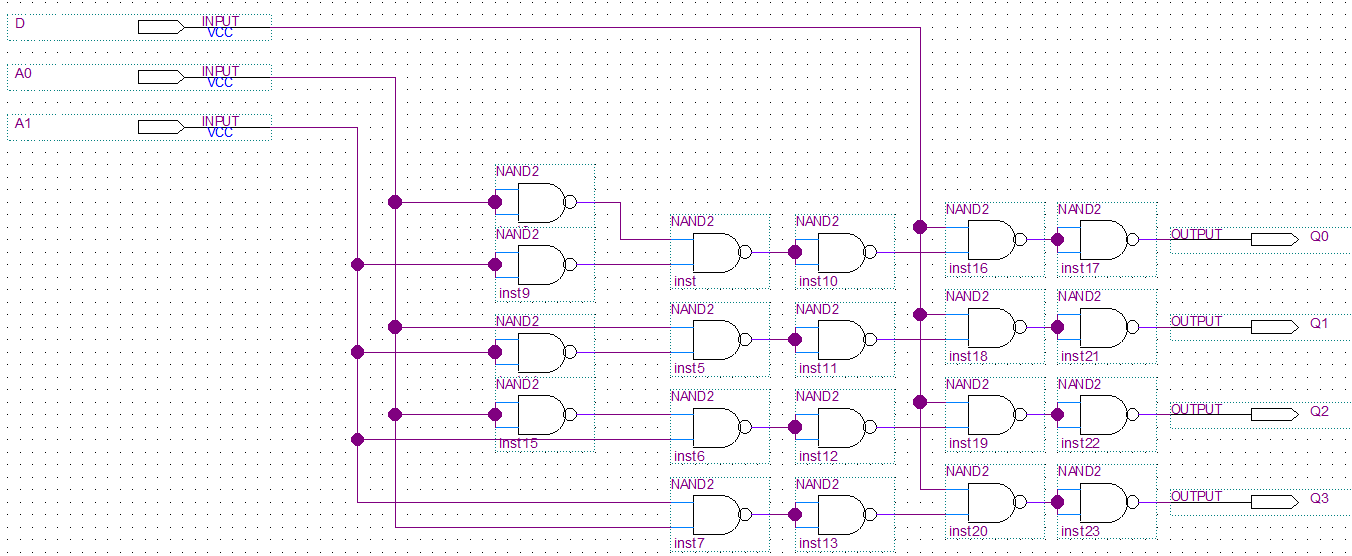
Соответственно схема практически совпадает со схемой дешифратора 2 на 4, а единственным наличием является дополнительный информационный вход «D»:

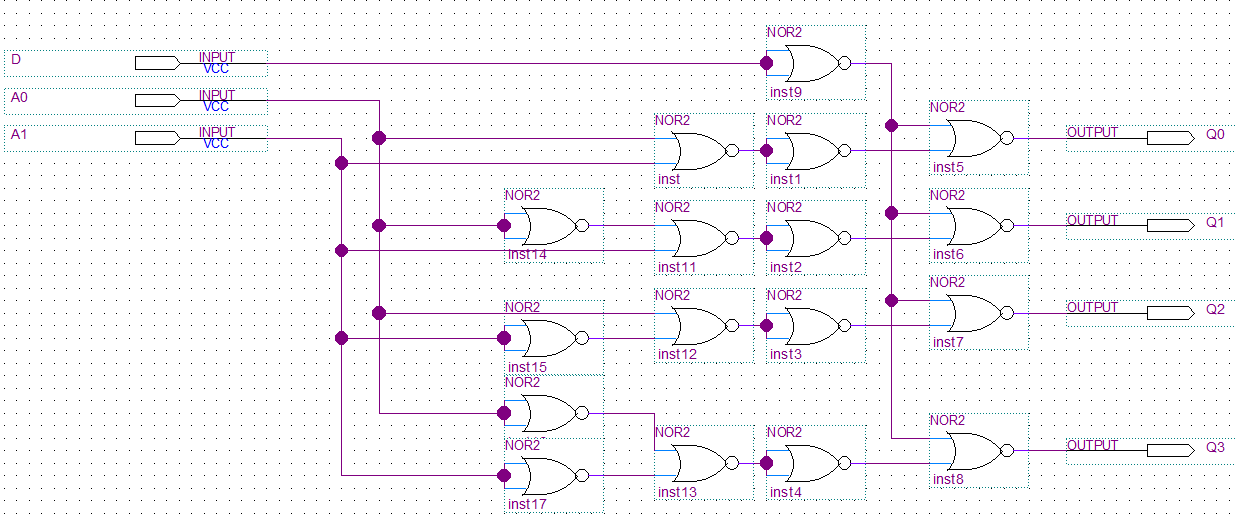


Временная диаграмма:



Представление схемы в базисе 2И-НЕ, 2ИЛИ-НЕ не представляет трудностей. Сделаем это через выражения и затем представим схему:





## Мажоритарный элемент (О?, ТИ, АВ)

Что это, зачем? Обобщённая схема, схема в 2И-НЕ, 2ИЛИ-НЕ.

## Сумматор и полусумматор

Главное отличие полусумматора от полного сумматора можно увидеть, рассмотрев сложение двух бинарных чисел:

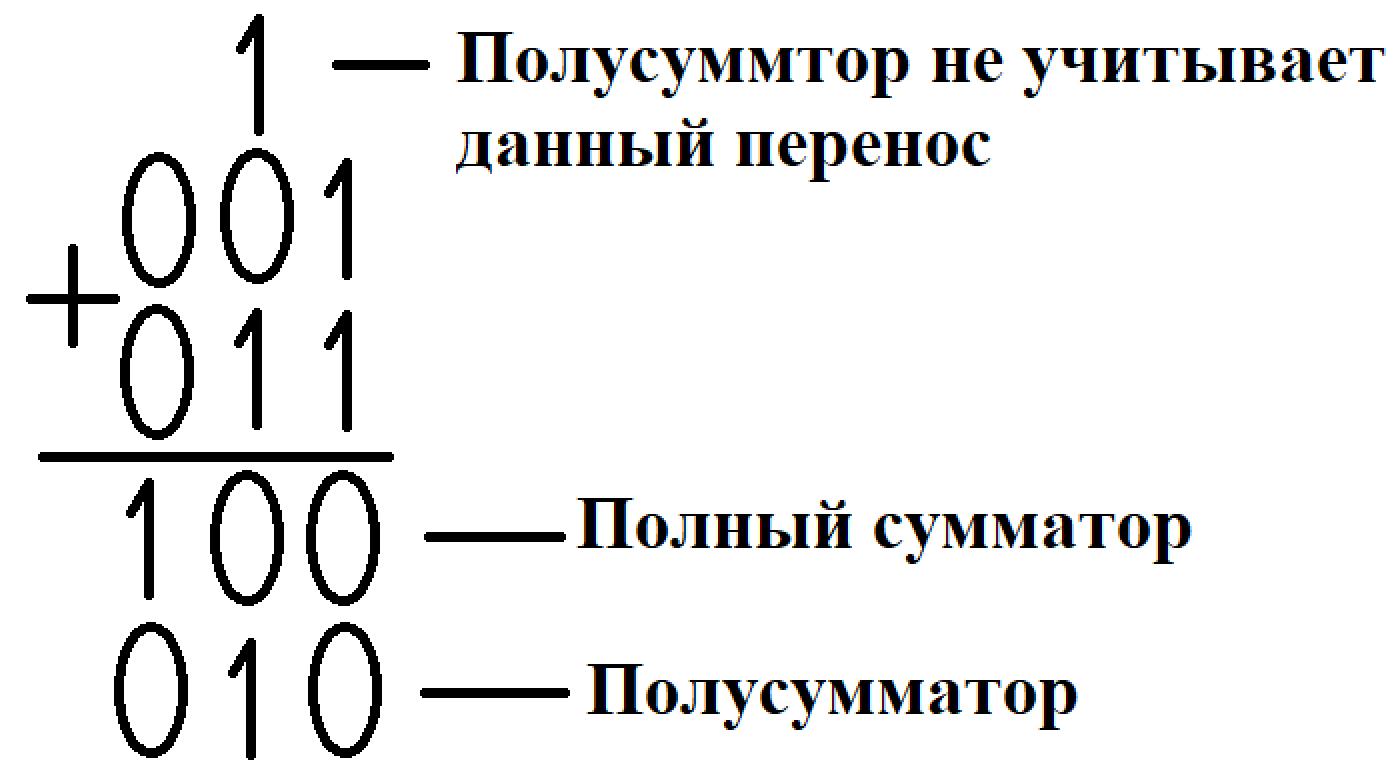
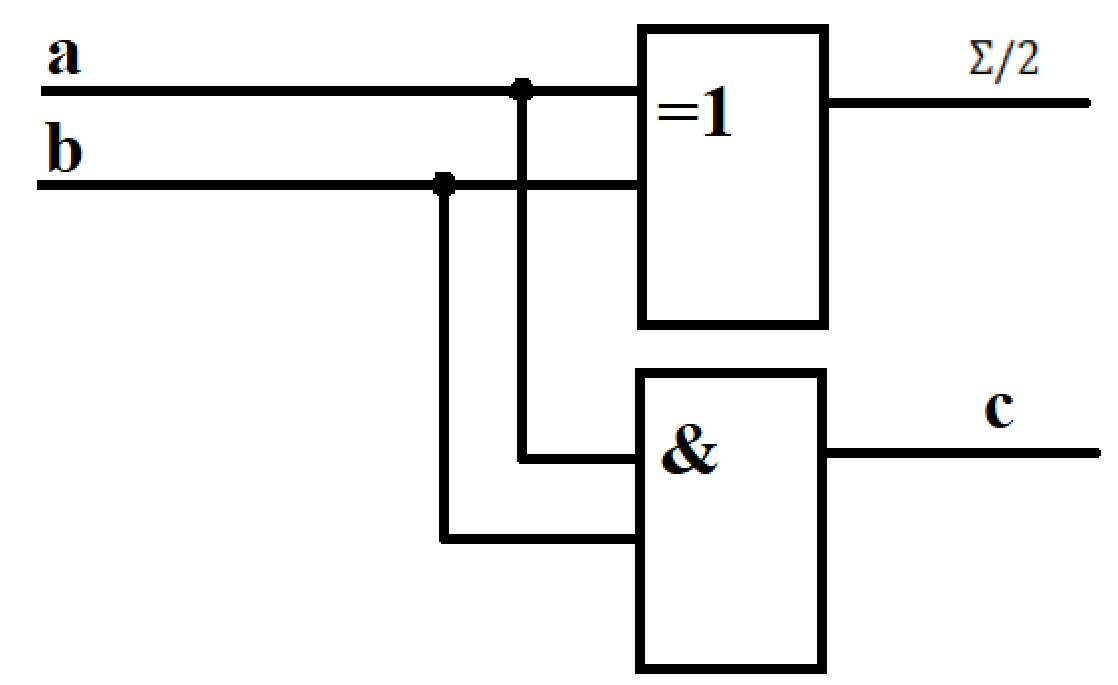


Таблица истинности полусумматора имеет следующий вид:

|  |  |  |  |
| --- | --- | --- | --- |
| a |  |  |  |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Рассмотрев столбец суммы полусuмматора (кек, хахахахахах)….Так, о чём это я…Рассмотрев столбец суммы , заметим, что «1» появляется только в случае, когда «1» присутствует лишь на одном из входов, что соответствует элементу ИСКЛЮЧАЩЕЕ-ИЛИ. Перенос с, в свою очередь, появляется лишь в случае «1» на обеих входах схемы, что соответствует логическому элементу 2И. Таким образом, схема полусумматора, а также выражение в СДНФ имеют следующий вид:

**

Обозначение полусумматора:

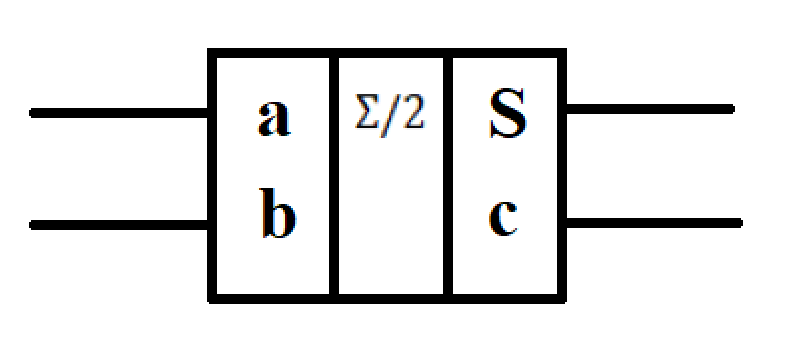
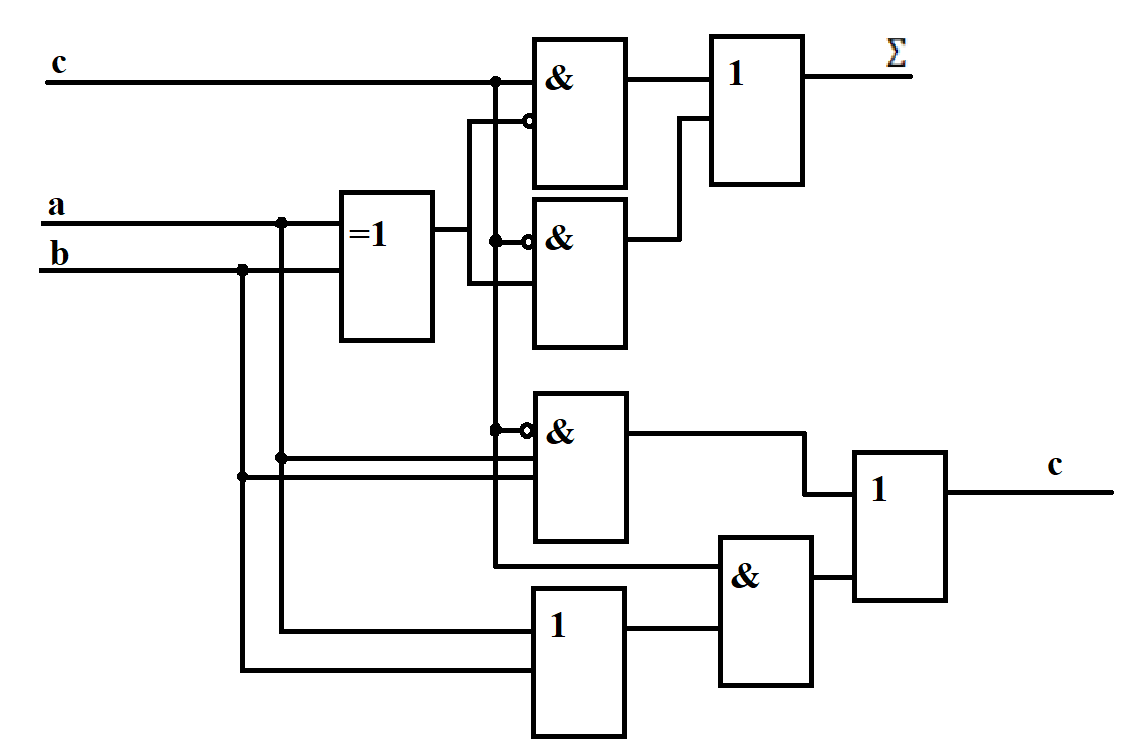


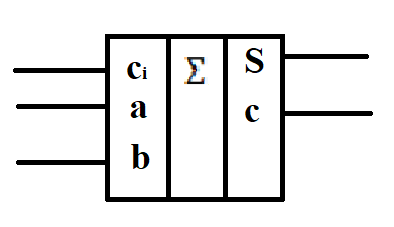
Таблица истинности сумматора имеет следующий вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a |  |  |  |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

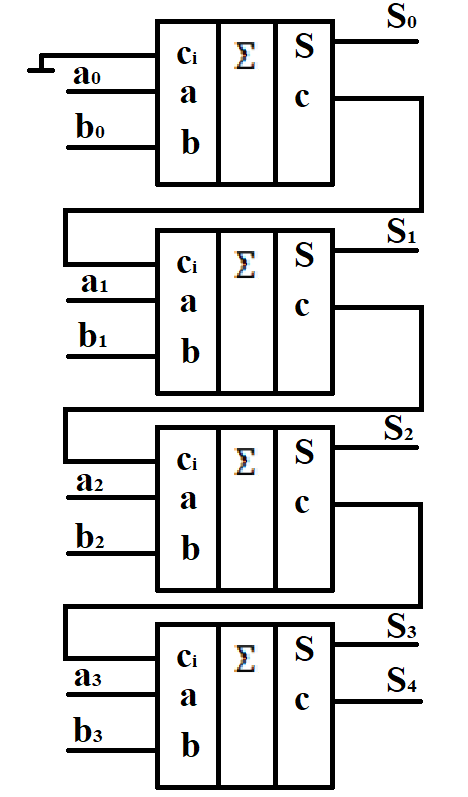
Первые 4 строки полностью повторяют таблицу истинности полусумматора. Заметим, что в строках, где перенос от предыдущего полусумматора (либо полного сумматора) равен «1», образует элемент ИСКЛЮЧАЮЩЕЕ-ИЛИНЕ, а перенос с образует элемент 2ИЛИ. Таким образом, выражения для сумматора и схема имеют следующий вид:



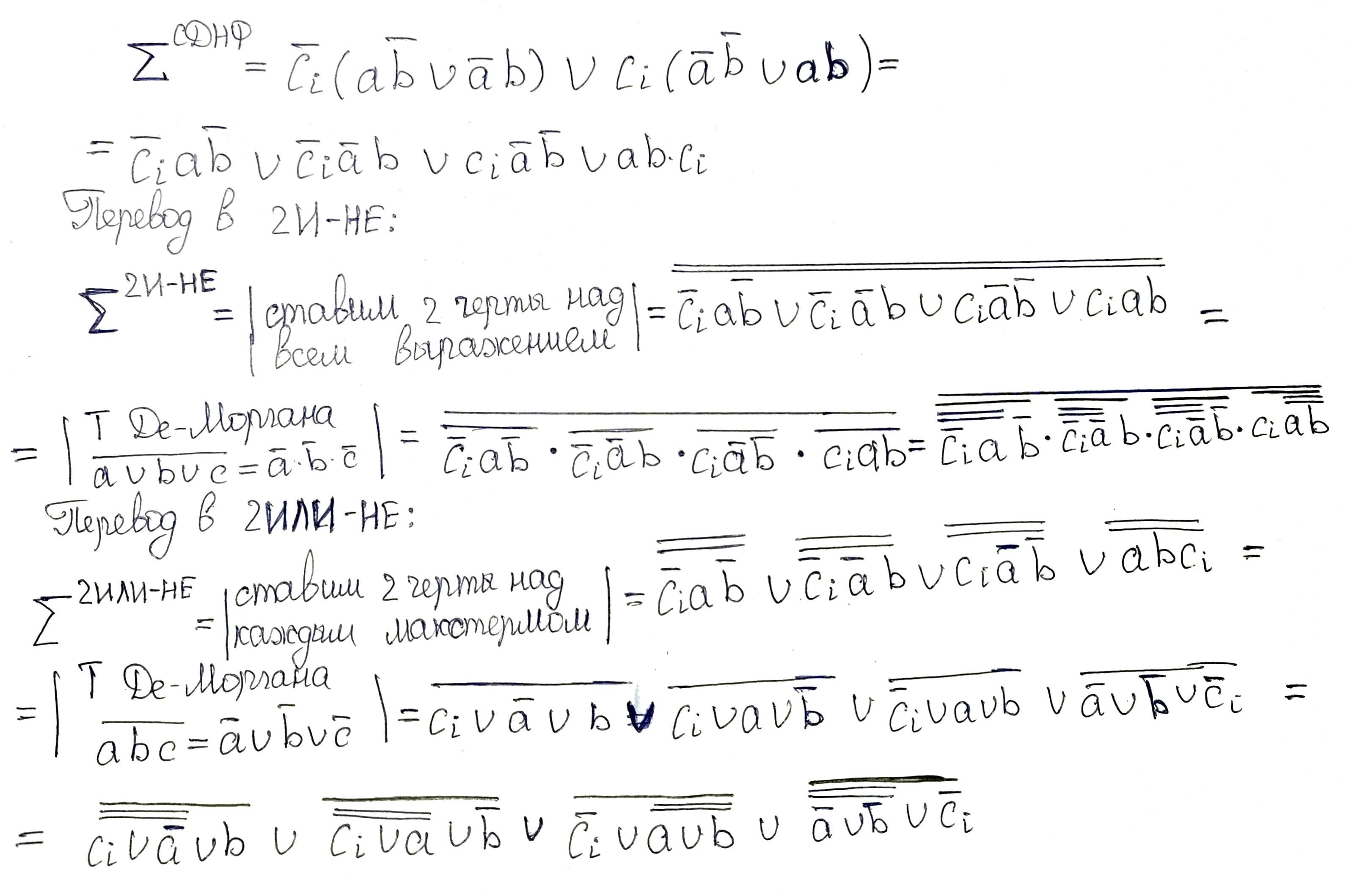
Обозначение сумматора:

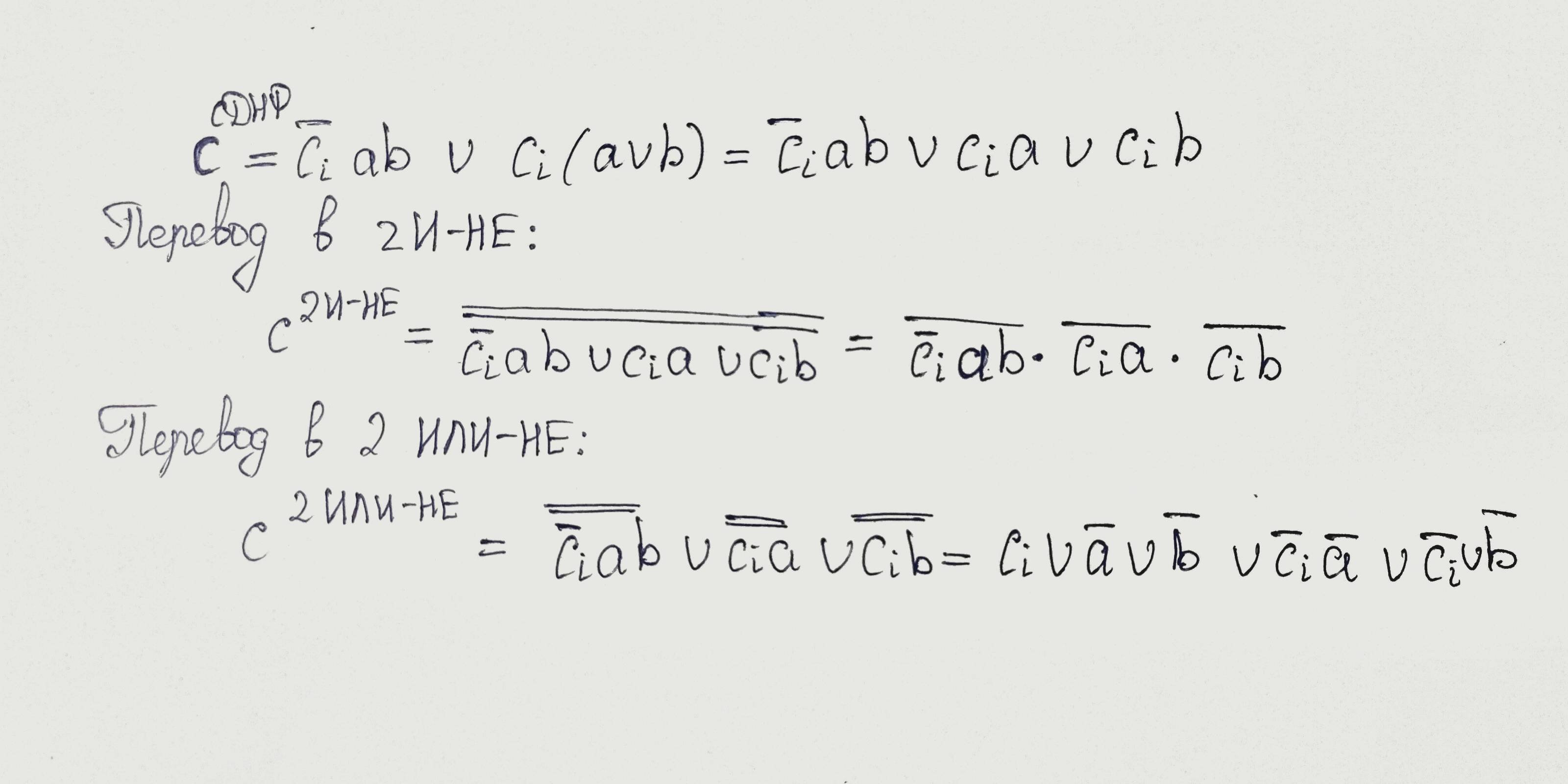


Каскадное включение сумматоров имеет следующий вид. Проверить, как они работают, думаю, можно самостоятельно:



Перевод выражений для сумматора в базисы 2И-НЕ и 2ИЛИ-НЕ:





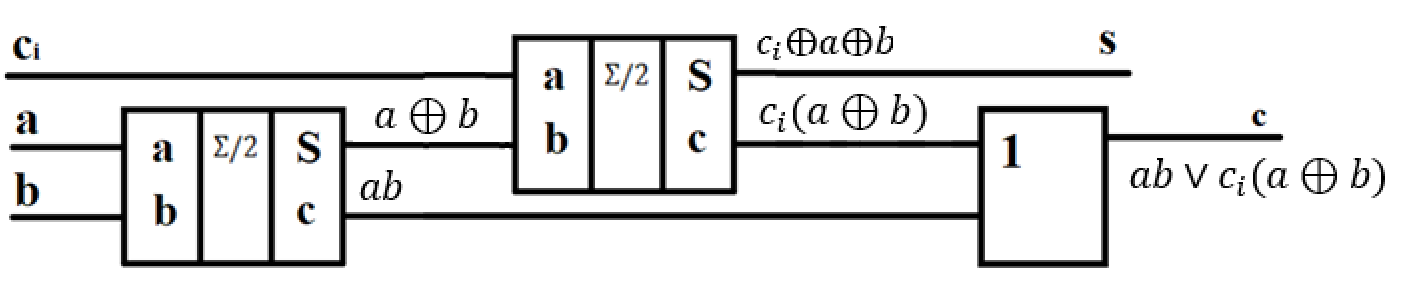
Схемы на 2И-НЕ и 2ИЛИ-НЕ находятся в файлах CNT\_2nand и CNT\_2nor.

Сумматор на полусумматорах. Вернёмся к выражениям для полного сумматора. Произведём для удобства следующую замену:

, тогда

Выражения для полусумматора, как было выведено ранее, имеют следующий вид:

Ну и теперь, основываясь на этих 4-ёх выражениях соберём схему сумматора на полусумматорах:



S полностью совпала, но С нет, как кажется на первый взгляд, но нужно заметить, что:

, ч.т.д.

## Полувычитатель и вычитатель

В чем разница (Н-р, с помощью вычитания двух бинарных чисел столбиком)? О, ТИ, АВ, обобщённая схема, схема в базисах 2И-НЕ, 2ИЛИ-НЕ. Вычитатель из полувычитателей? Каскадное включение вычитателей.

## Умножители. Умножитель 2х2, 3х3, 4х4

## Компаратор (О, ТИ, АВ)

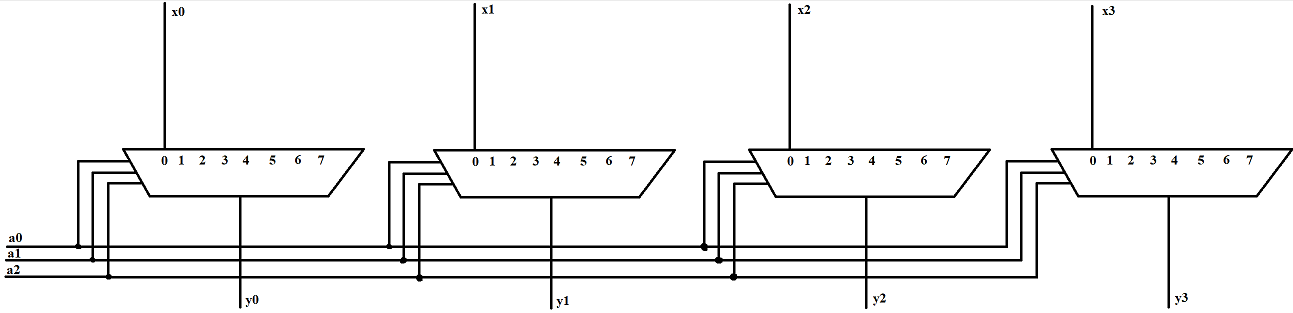
Зачем нужен, принцип работы? Последовательный компаратор. А есть параллельный (на лекциях ничего не написали, нужно почитать)?

## Быстрый сдвиговый регистр (Barrel shifter)

Устройство быстрого сдвига (Barrel shifter) представляет собой комбинационную логическую схему с n входами данных, n выходами данных и несколькими управляющими входами, сигналы на которых задают сдвиг между данными на входе и данными на выходе.

Называется быстрым, так как построен на мультиплексорах, то есть с использованием только комбинационной логики (триггеры, а значит и последовательностная логика, отсутствуют).

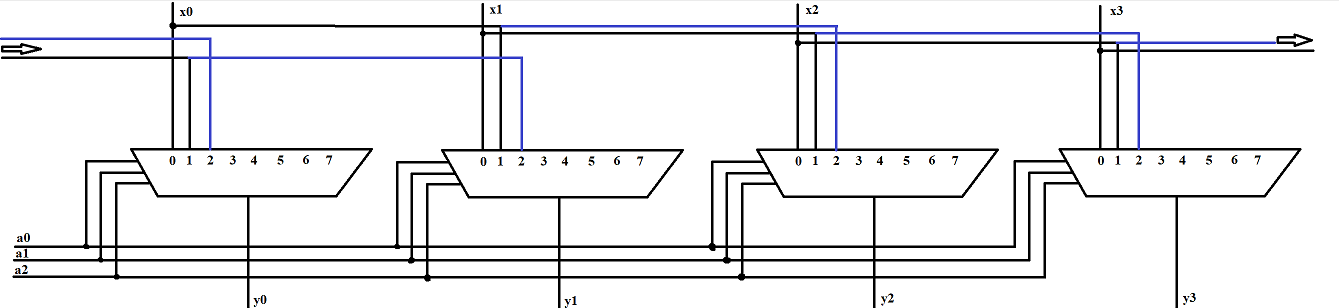
Допустим, будем сдвигать 4-ёх битовое число. Для этого нам понадобятся 4 мультиплексора с 3 адресными входами. На «изначальную» схему на 0 входы мультеплексоров поразрядно поступает наше двоичное число (). На выходах образуется двоичное число после сдвига. Предположим, что (адресные входы). Тогда на выход регистра наше число пройдет без изменений.



Если же установить адресную комбинацию , то число должно быть сдвинуто вправо на «1» разряд. Для этого необходимо соединить со входом «1» второго мультиплексора, – со входом «1» третьего мультиплексора, – со входом «1» четвёртого мультиплексора, – со входом «1» первого мультиплексора:



Если установить адресную комбинацию , то число должно быть сдвинуто вправо на «2» разряда. Для этого необходимо соединить со входом «2» третьего мультиплексора, – со входом «2» чётвертого мультиплексора, – со входом «2» первого мультиплексора, – со входом «2» второго мультиплексор.



По такой логике соединяем все входы, пока не сдадим экзамен Семер…пока входы мультиплексоров не закончатся. Полная схема представлена в Barrel\_shifter.bdf.

## АЛУ

Написали про них мало, но возможно стоит поискать более подробную общую информацию, красиво все описать, а не как в лекции одной картинкой, в которую еле всё поместилось и ниче не понятно.

## Статический сбой (static hazard)

Пример на какой-то схеме, н-р, как в лекции, но если будут другая, то будет только +

В цифровой электронике статический сбой (static hazard) относится к временному сбою или ошибке, которая может возникнуть в выходном сигнале цифровой схемы из-за несоответствия времени или дисбаланса входных сигналов схемы. Эти помехи могут возникать, когда несколько входных сигналов меняются одновременно или когда в схеме присутствуют задержки, которые не сбалансированы должным образом.

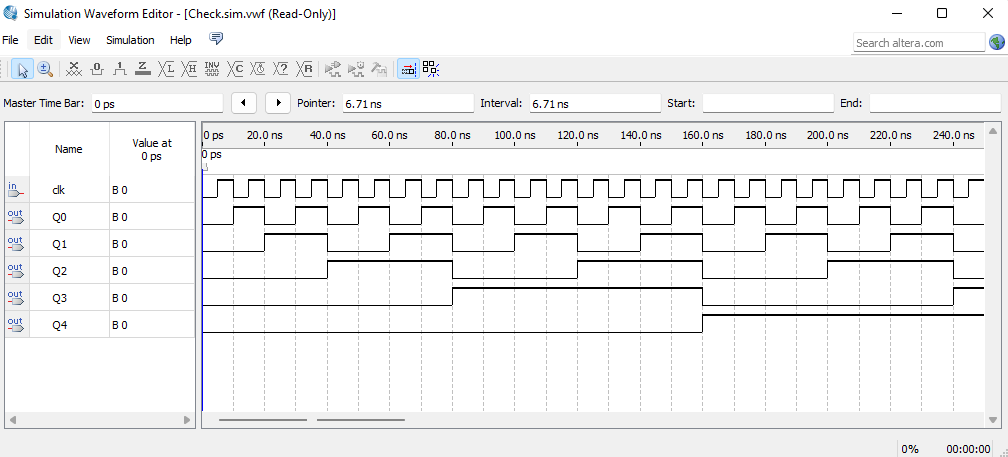
Статические помехи могут привести к неправильным или непредсказуемым значениям выходного сигнала, что может вызвать проблемы в цифровых системах. Они особенно проблематичны в синхронных схемах, где время является критическим фактором. Возникновение помехи, вызванной статической ошибкой, может нарушить намеченное логическое поведение схемы, что приведет к неправильной обработке данных или неправильной функциональности.

Статистический сбой «1» - сбой, при котором должна быть «1», но появляется «0».

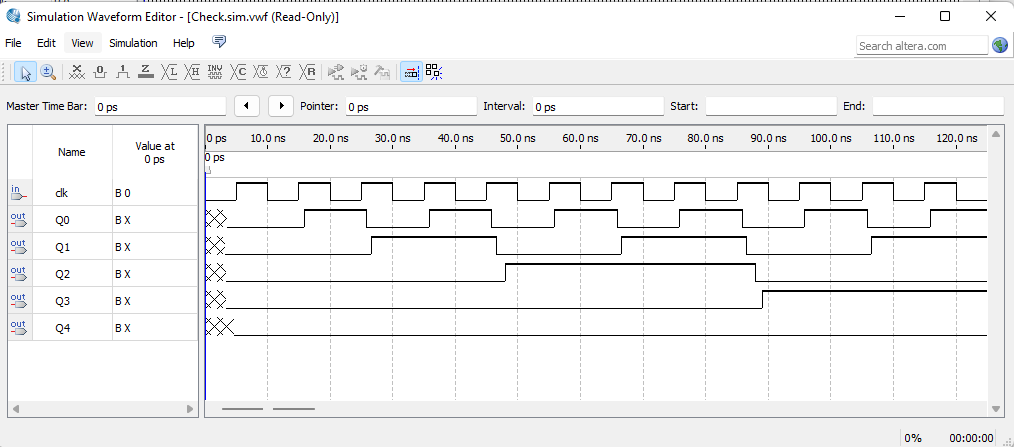
Статистический сбой «0» - сбой, при котором должен быть «0», но появляется «1».

Рассмотрим статистический сбой на примере асинхронного счётчика на T-триггерах по переднему фронту (T\_counter.bdf).

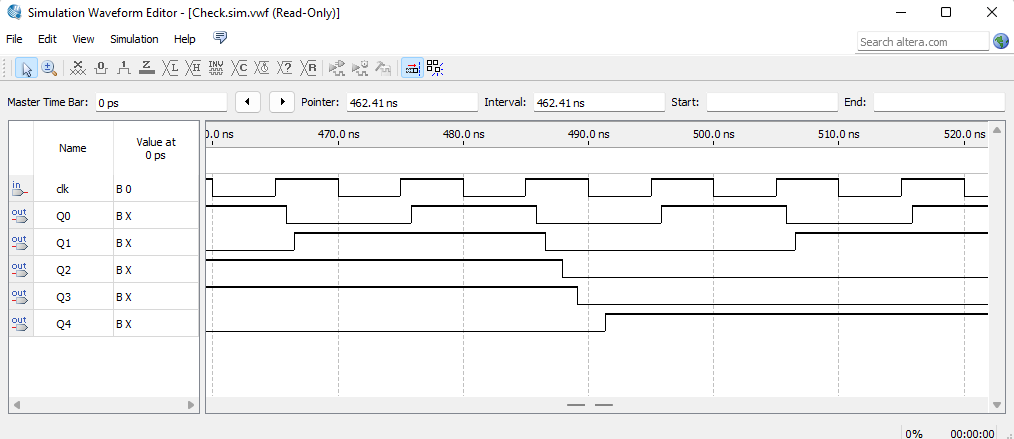
Временная диаграмма идеальной (не имеющей временных задержек) схемы выглядит следующим образом:



Временная диаграмма схемы с временными задержками выглядит следующим образом:



Как видим из диаграммы, в отличие от идеального случая у нас возникает некая временная задержка установления сигнала на выходе при прохождении T-триггера. В связи с тем, что счётчик асинхронный (сигнал на выходе следующего триггера тактируется от сигнала на выходе предыдущего), то временная задержка «накапливается» и может выливаться в следующие казусы:

**

После 15 («01111») на выходах должно установиться число 16 («10000»), но в результате временных задержек на выходах происходит некая суета, и лишь после жесткого тусича сигналов наконец устанавливается 16.

## Вопросы на подумать

Просьба не сразу выдавать решение, а сначала написать некий алгоритм рассуждений, на основе которого можно самому попытаться построить цифровую схему.

При решении вариантов можно не делать задания, которые уже присутствовали в вопросах до этого, если только конечно вы не знаете какой-то изощрённый и отличающийся от представленного в методе ранее способа решения.

### ИСКЛ ИЛИ на 4 элементах (базис 2И-НЕ). ИСКЛ ИЛИ в базисе 2ИЛИ-НЕ

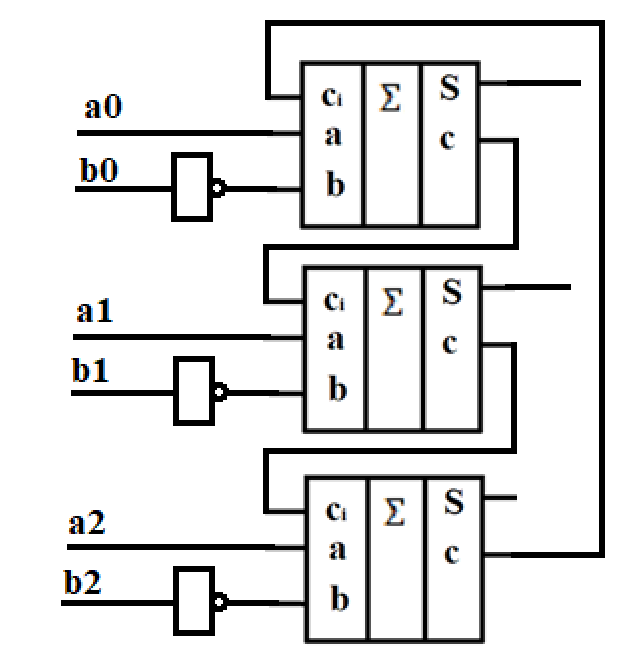
### Вычитатель на сумматоре

Вычитатель на сумматоре строится следующим образом: чтобы не реализовывать нудную операцию вычитания с переносами, можно перевести вычитаемой число в дополнительный код, то есть представить его в отрицательном виде. Например:

Для начала переведём 8 в обратный код:

Это ещё не дополнительный код, так как нужно прибавить «1», но повременим с этим и вот почему:

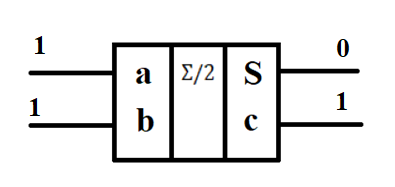
То есть результат сложения 12 в прямом коде и 8 в обратном дал и образовалась 1 переноса в старший разряд, которую мы можем использовать для перевода 8 в дополнительный код. То есть сейчас к нужно прибавить эту потерянную единицу:

**

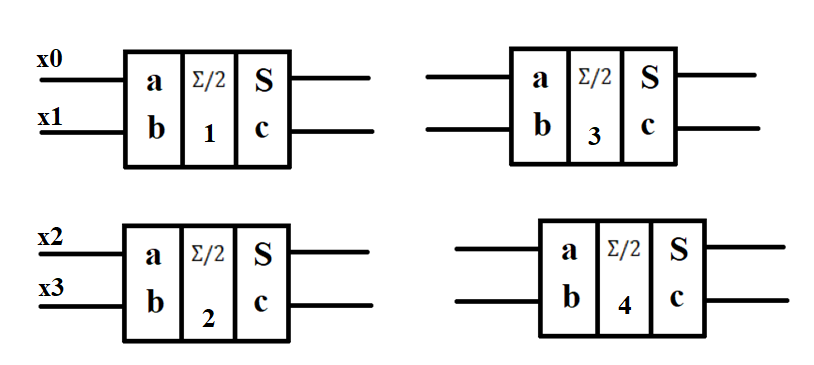
### Сложите 4-5 одноразрядных чисел. Есть восьмиразрядное число, определить количество единиц в нём

Задача решается с использованием полусумматоров. Для её решения нужно осознать две вещи:

1) Складываем «1», то есть , следовательно, на первой ступени полусумматоров выход «S» будет означать младший разряд (нулевой), а выход C, в свою очередь, уже первый разряд.

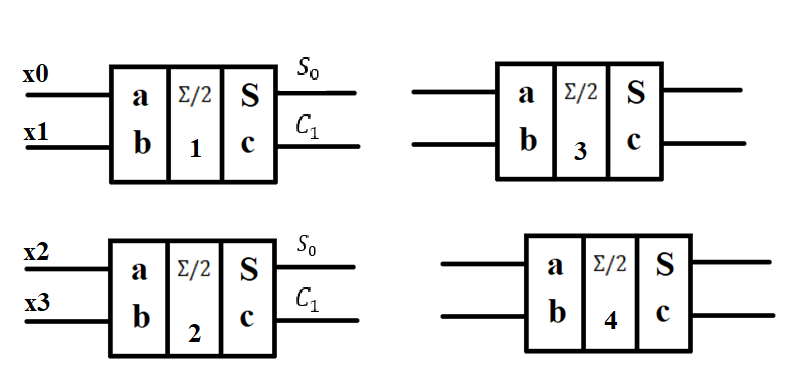


2) Не путать какие разряды получаются на выходах полусумматора. Для примера сложим разряды 4-ёх разрядного числа:

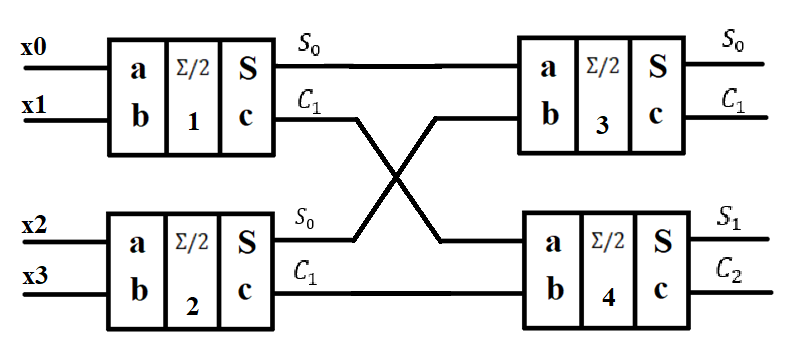


На выходе полусумматора «1» получаем сумму (0 означает, что это будет нулевой разряд полученного числа) и перенос (1 означает, что данная двоичное число уже относится к первому разряду получаемого числа).

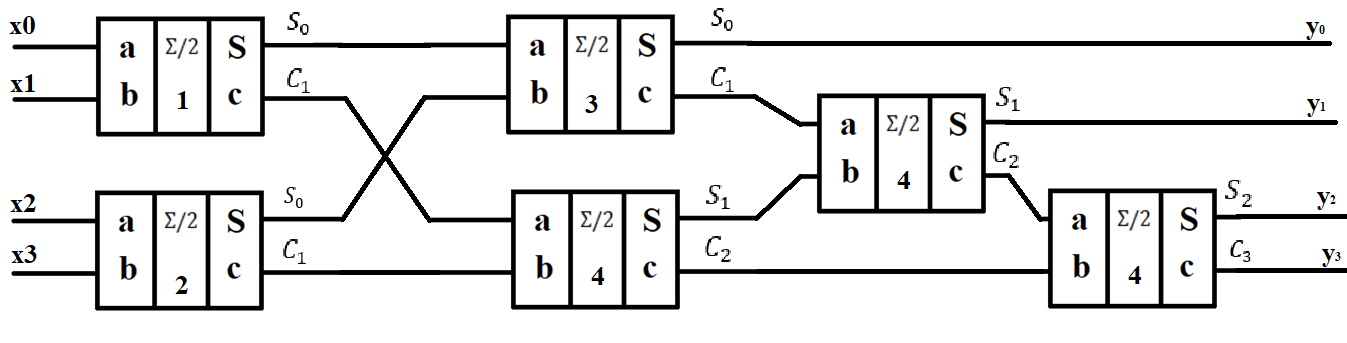
На выходе полусумматора «2» аналогично: (0 означает, что это будет нулевой разряд полученного числа) и перенос (1 означает, что данная двоичное число уже относится к первому разряду получаемого числа).



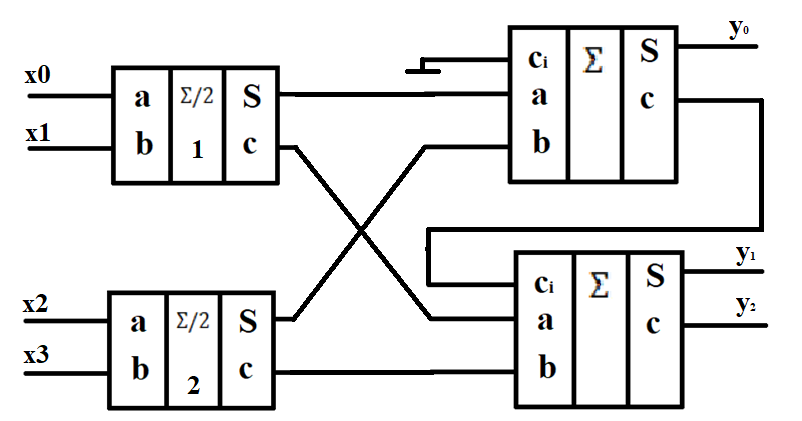
Далее действуем по той же логике: нулевые разряды складываем с нулевыми, а первые с первыми. Тогда, на выходах полусумматора «3» получаем и , а на выходах полусумматора «4» и :



Складываем и для получения первого разряда, при этом образуется перенос , который необходимо с «4» полусумматора:

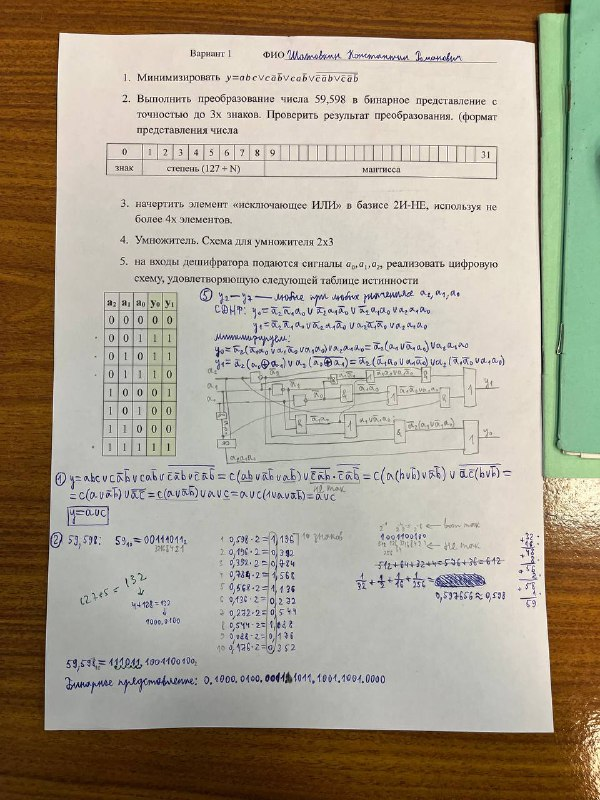
**

Более компактно схему можно получить, используя полные сумматоры:

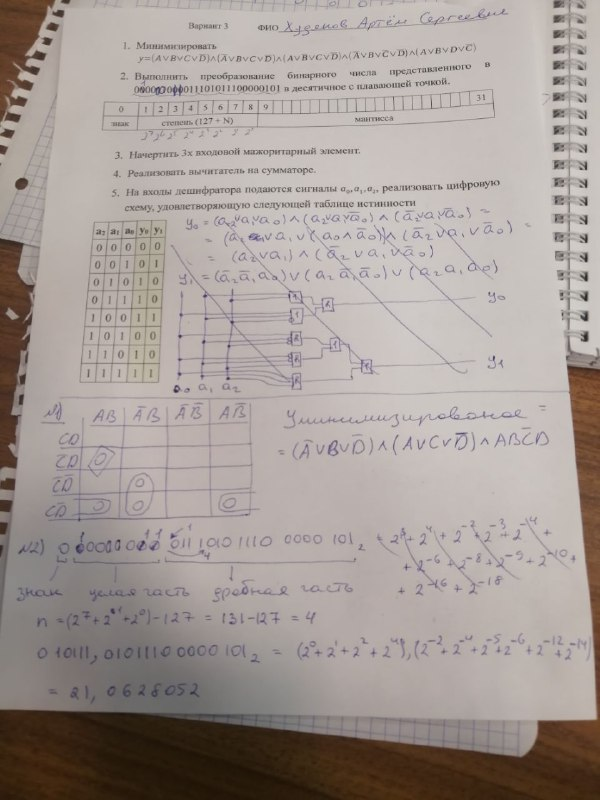


### Компаратор на сумматорах.

### Вариант 1 РК1



### Вариант 3 РК1



1. Минимизировать:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 1 | 1 | 1 | 1 |
|  | 0 | 1 | 1 | 1 |
|  | 1 | 0 | 1 | 1 |
|  | 0 | 0 | 1 | 0 |

2. Выполнить преобразование бинарного числа, представленного в

в десятичное с плавающей точкой.

Решаем обратную задачу. У нас имеется:

мантисса

степень

знак

Переведём степень в десятичный код:

, следовательно 131 – 127 = 4

Сдвинем 4 знака в мантиссе и вставим «1» в начале числа:

1)

2) 10111,01011100000101

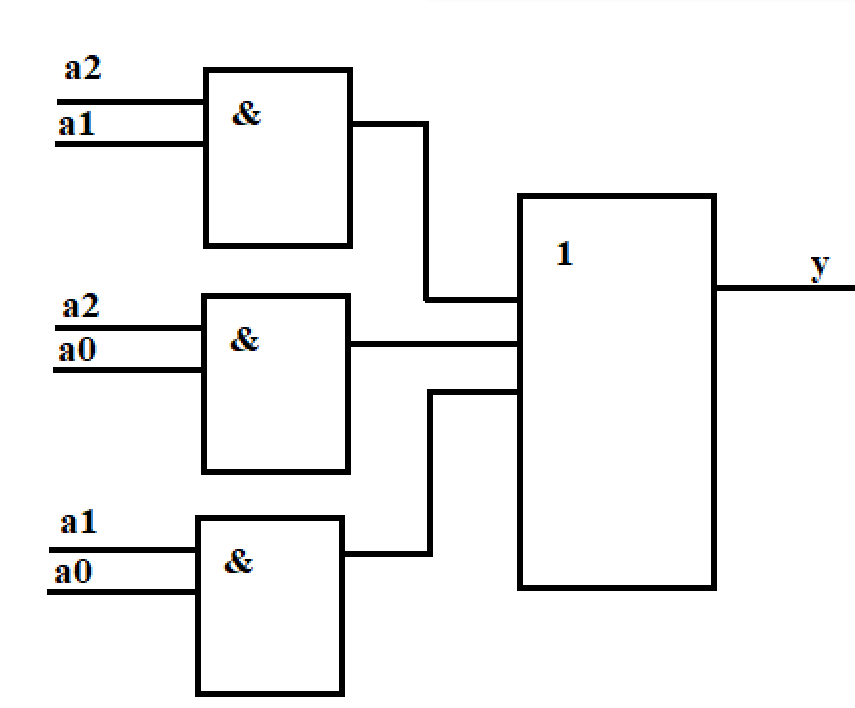
Переведём число в десятичное с плавающей точкой:

Таким образом, ответ 23,36.

3. Начертить 3-ёх входовой мажоритарный элемент.

Построим ко СДНФ аблица истинности:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | y |
| 1 | 1 | x | 1 |
| 1 | x | 1 | 1 |
| x | 1 | 1 | 1 |



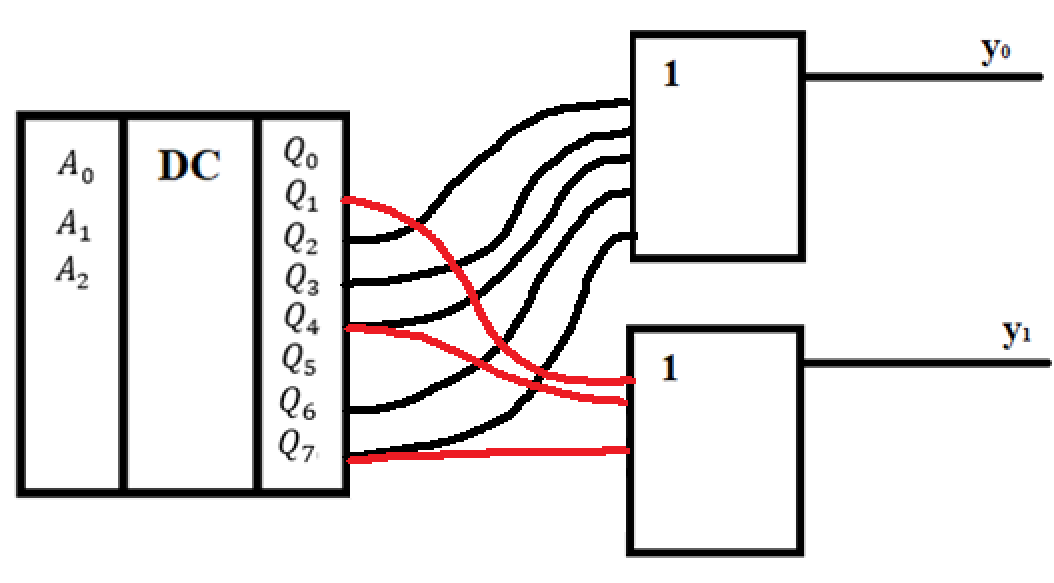
4. [Вычитатель на сумматоре.](#_Вычитатель_на_сумматоре)

5. На входы дешифратора подаются , реализовать цифровую схему, удовлетворяющую следующей таблице истинности:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Примем что у нас уже есть готовый дешифратор и нам нужно присобачить к выходам схему, которая будет удовлетворять условию.

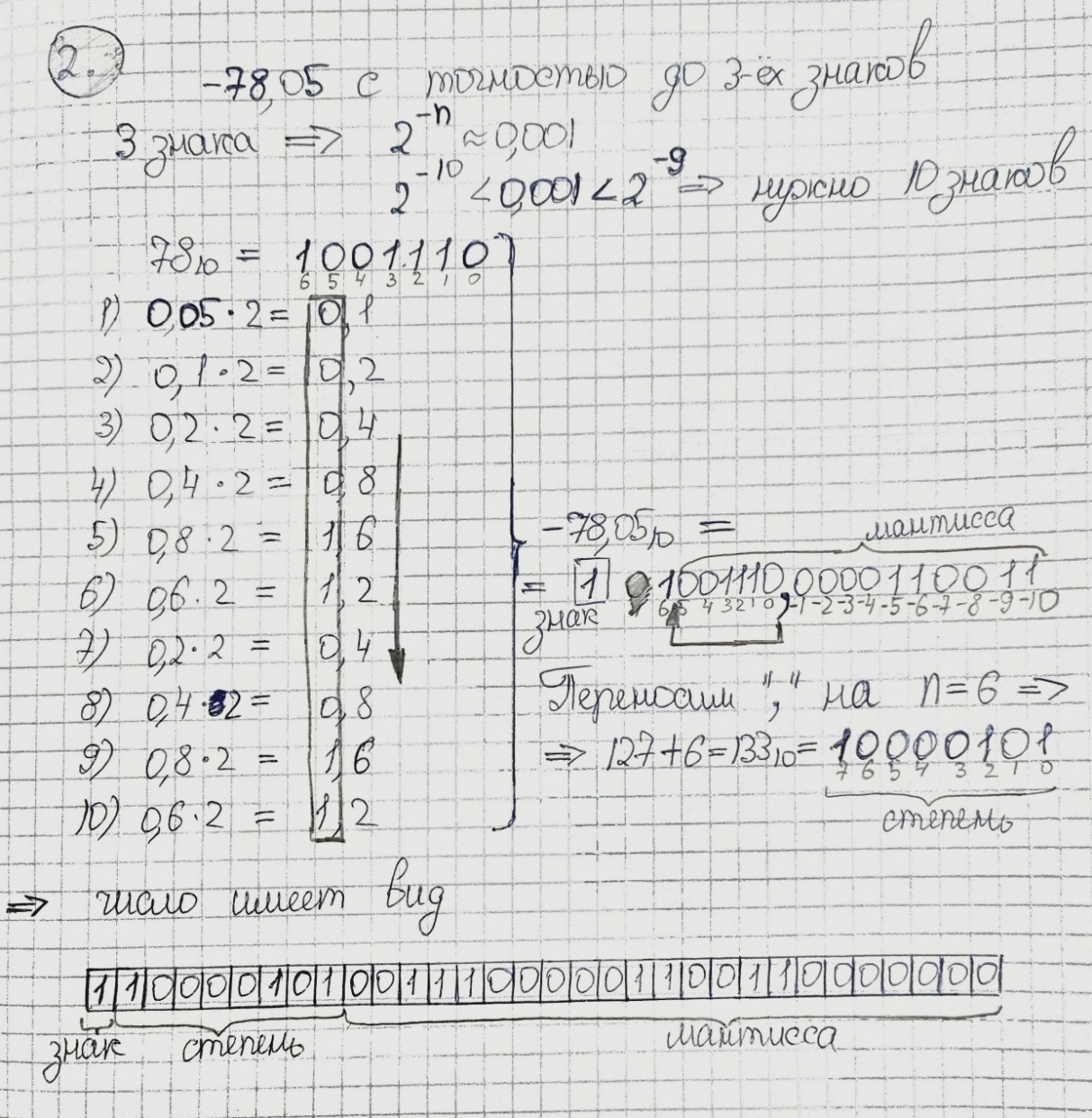
Схема: берем элементы OR5 и OR3 и подключаем к ним соответствующие выходы дешифратора.



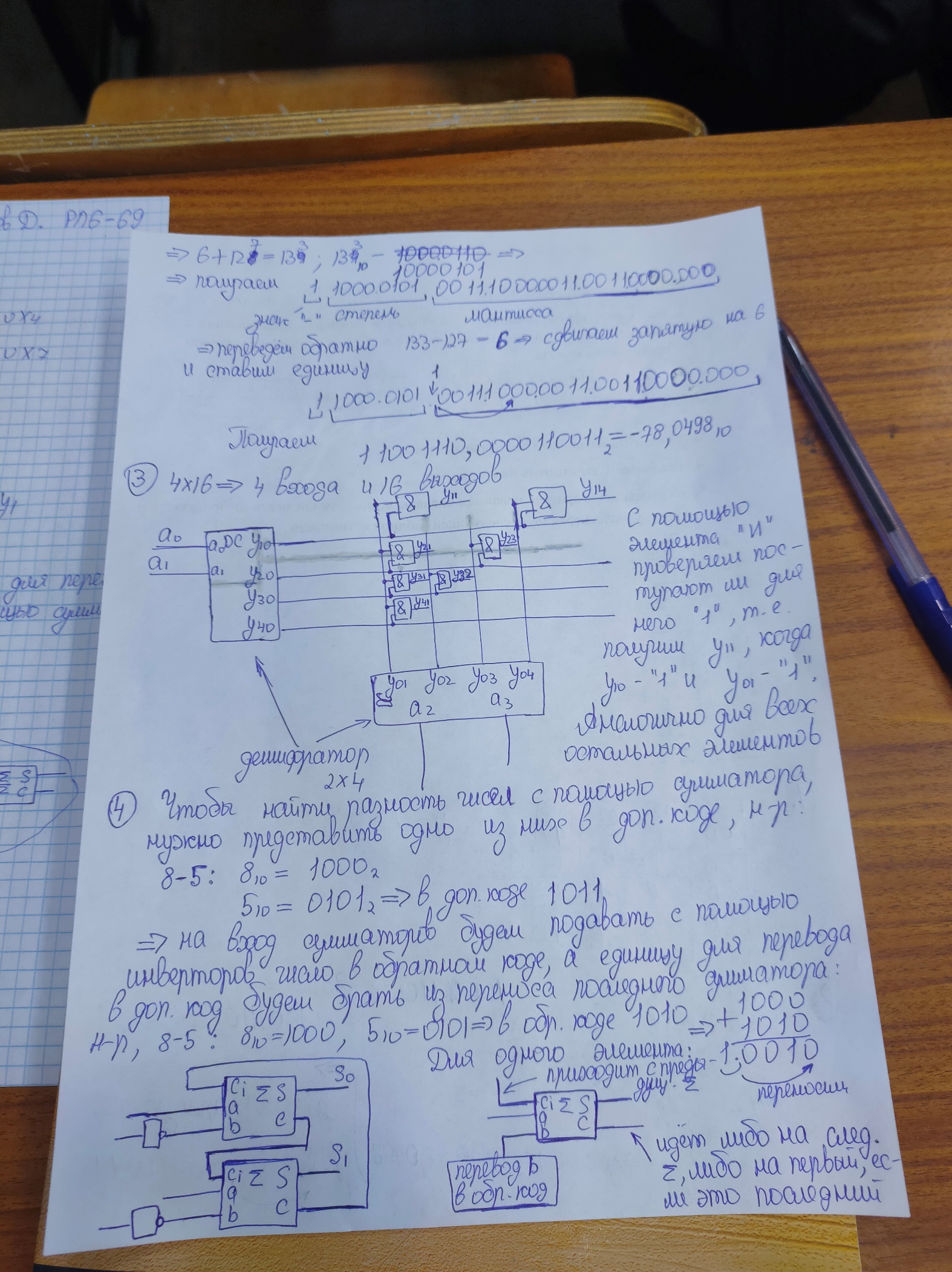
### Вариант 4 РК1

1. Минимизируем с помощью алгебраических операций:

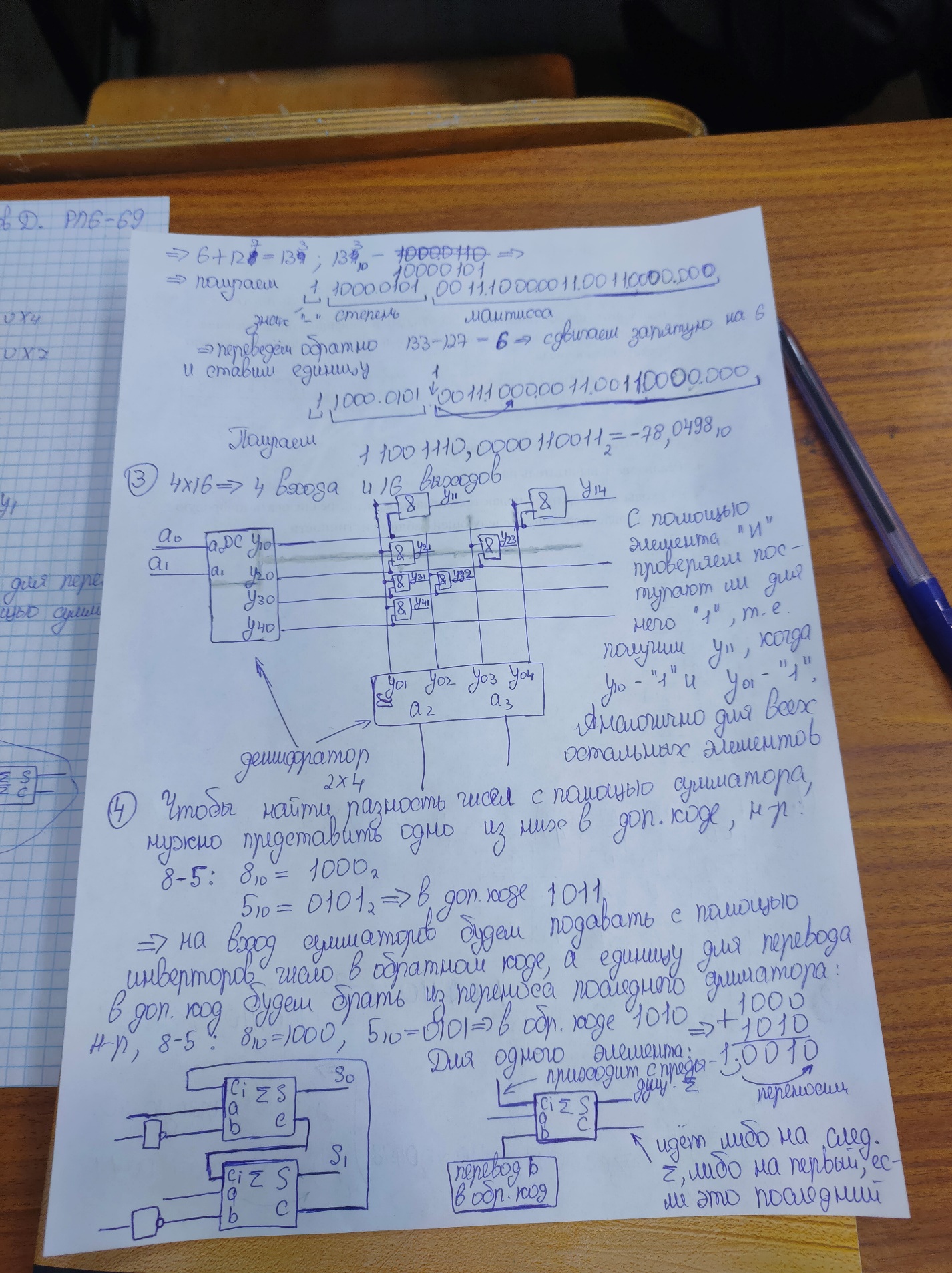
2. Выполнить преобразование числа -78,05 с бинарное представление с точностью до 3-х знаков.

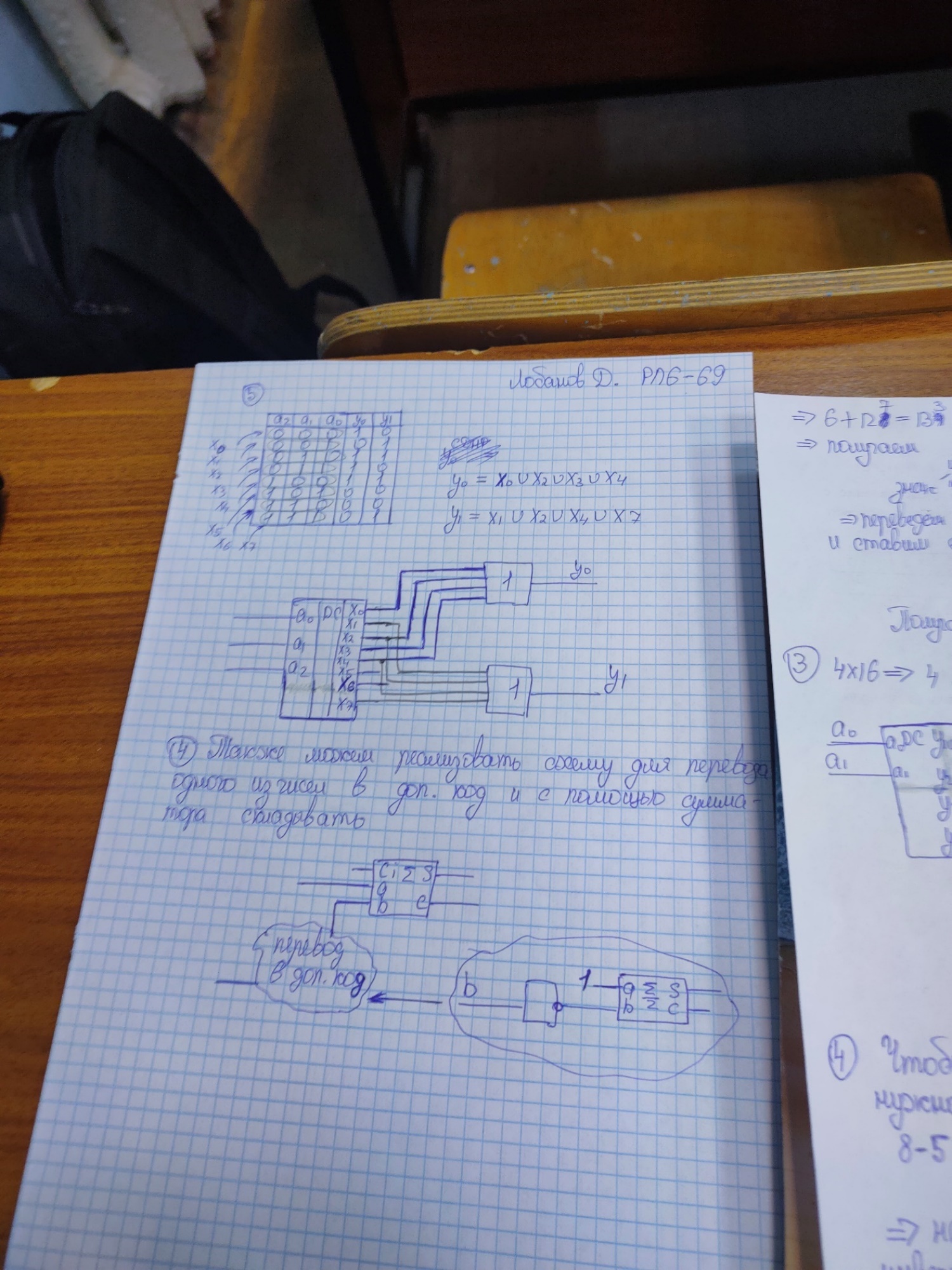


3. Начертить матричный дешифратор 4\*16; [\*тык\*](#_Дешифратор_матричного_типа+)



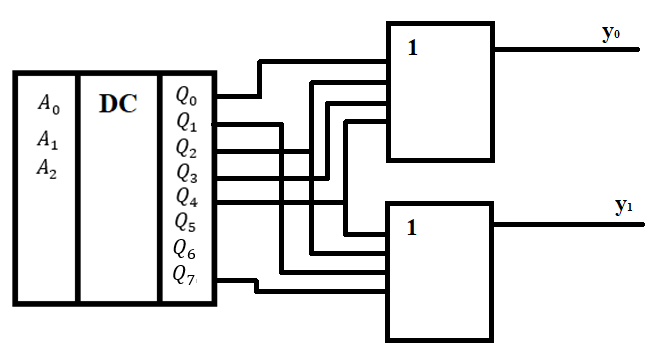
4. Реализовать вычитатель на сумматоре:



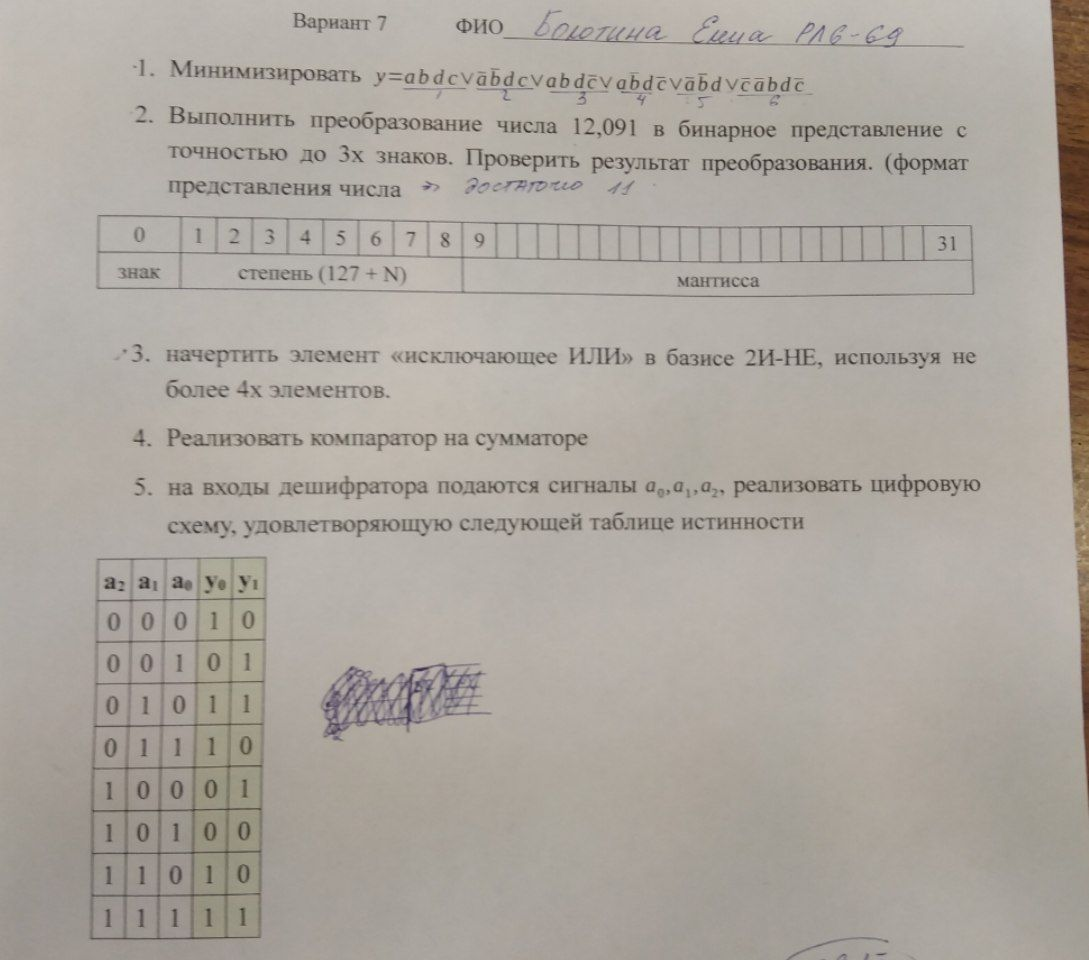


5. На входы дешифратора подаются . Реализовать цифровую схему, удовлетворяющую следующей таблице истинности:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |



### Вариант 7 РК1



# Последовательстная логика (логика с памятью)

В чём отличие от комбинационной логики? Шо це таке?

## RS-триггер, статический, асинхронный, одноступенчатый

Пояснить статический, асинхр. О, ТИ, ВД. Построить на 2И-НЕ и 2ИЛИ-НЕ. В чём отличие? Как избежать запрещённого состояния?

## RS-триггер, синхронный, статический, одноступенчатый

Пояснить статический, синхр. О, ТИ, ВД. Построить на 2И-НЕ и 2ИЛИ-НЕ. В чём отличие? Как избежать запрещённого состояния?

## D-триггер (D flip-flop), синхронный, статический, одноступенчатый

О, ТИ, ВД. Всегда синхронный! Статический? Реализция D-триггера методом «чёрного ящика». На 2И-НЕ и 2ИЛИ-НЕ.

## Динамический D-триггер, синхронный, двухступенчатый

Переключение по заднему фронту, по переднему? О, ВД.

## T-триггер (счётный)

О, ВД.

## Асинхронный счётчик на D-триггерах.

Что значит Асинхронный? О, ВД. Как реализовать сброс счётчика?

## Динамический RS-триггер

ВД, О, схема. По переднему фронту, по заднему?

## Вычитающий счётчик (асинхронный)

Асинхронный? О, ВД

## Реверсивный счётчик

Принцип, ВД.

## Счётчик с предустановленными значениями

Принцип, ВД.

## Кольцевой счётчик (синхронный)

Принцип, ВД

## Кольцевой счётчик Джонсона

Принцип, ВД, как можно использовать для создания дешифратора?

## Кольцевой счётчик (ходит только одна единица)

Принцип, ВД, самовосстановление счётчика

## Синхронный счётчик вариант 1

Зачем нужны? Принцип работы, ВД. Реальные ВД для синхронного и асинхронного счетчиков, чтобы показать преимущества синхронного.

## Синхронный счётчик вариант 2

## Регистры. Последовательно-параллельные

## Регистры. Параллельно-последовательные

## Регистры параллельно-параллельные

## Регистры. Циклический регистр. Выбор направления. Предустановка значений.

## Динамический одноступенчатый D-триггер

Страшная схема

## Динамический D-триггер с асинхронным сбросом, установкой

Страшная схема со сбросом и с установкой

## JK-триггер, синхронный, статический

ВД, О

## Динамический JK-триггер

## Динамический JK-триггер на D-треггерах

## T-триггер на JK-триггере

## Синхронный счётчик на JK-триггере

## Сброс триггера. Асинхронные, синхронные входы.

## Задачи на подумать.

Просьба не сразу выдавать решение, а сначала написать некий алгоритм рассуждений, на основе которого можно самому попытаться построить цифровую схему.

### Счётчик в базисе 2И-НЕ

### Имеется кольцевой счётчик. Как загнать «1» или несколько «1» и двигать их по кругу?