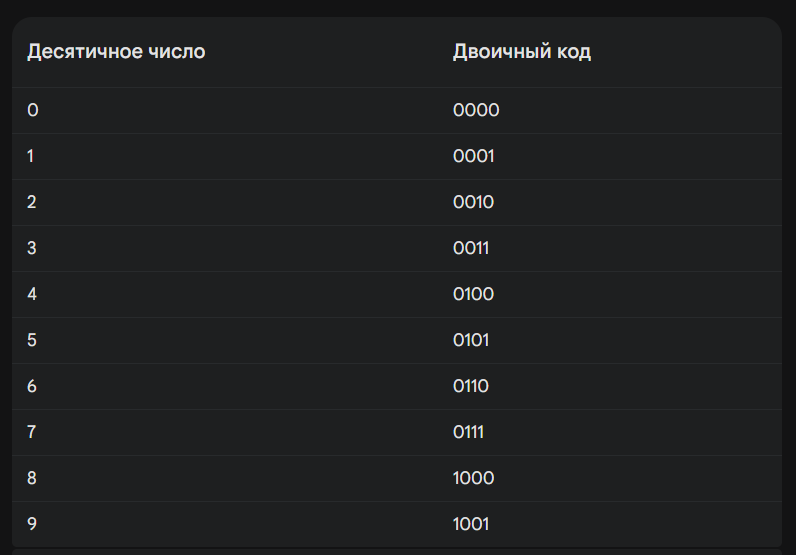
**Билет 1**

**1. Двоично — десятичная система счисления.**

**Двоично-десятичная система (ДДС, также называемая BCD, от англ. Binary-Coded Decimal)** – это способ представления десятичных чисел с помощью двоичных цифр. В отличие от стандартной двоичной системы, где используются только 0 и 1, в ДДС каждое десятичное число (от 0 до 9) кодируется своим уникальным 4-битным двоичным кодом



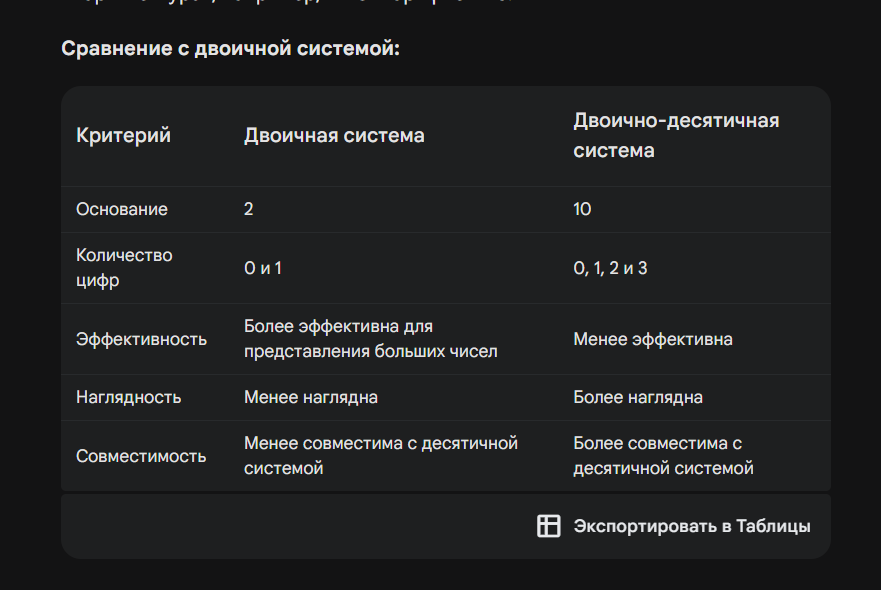
Например, десятичное число 45 в BCD представляется как 0100 0101, где 0100 — это 4, а 0101 — это 5.

### Преимущества:

1. **Удобство при работе с десятичными числами**: Легко преобразовать и интерпретировать числа, так как каждая цифра десятичного числа представлена отдельно.
2. **Простота операций с десятичными числами**: Легко выполнять арифметические операции, так как они выполняются по десятичным правилам.

### Недостатки:

1. **Эффективность использования памяти**: Требуется больше битов для представления числа по сравнению с обычной двоичной системой. Например, число 45 в обычной двоичной системе — это 101101, что занимает всего 6 битов, в то время как в BCD требуется 8 битов.
2. **Сложность реализации**: Арифметические операции могут быть менее эффективными, так как требуют дополнительных шагов для обработки каждой десятичной цифры.

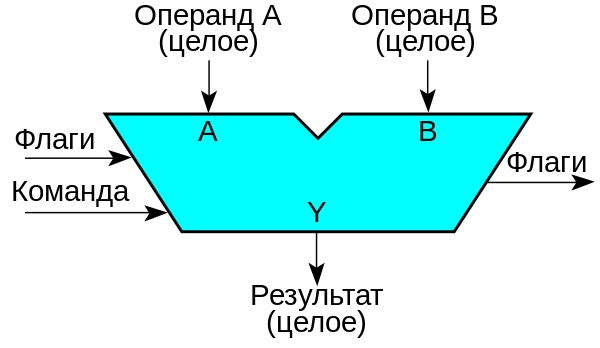


В целом, двоично-десятичная система является более простой и понятной альтернативой стандартной двоичной системе, но при этом она менее эффективна.

**2. 4-х разрядное арифметико логическое устройство. Графическое обозначение.**

**4-х разрядное арифметико-логическое устройство (АЛУ)** - это цифровая схема, предназначенная для выполнения **арифметических и логических операций** над **4-х разрядными двоичными числами**.

АЛУ является **основным компонентом** процессора и используется в различных приложениях, таких как **вычисления, обработка данных и управление**.



(Ебать как не факт что это нужно)

### **Работа ALU:**

ALU в зависимости от управляющих сигналов Op0 и Op1 может выполнять различные операции над операндами A и B. Например, для операции сложения установлены управляющие сигналы Op0=0 и Op1=0, для вычитания Op0=1 и Op1=0 и так далее. Результат операции появляется на выходе Result, а флаги состояния обновляются в соответствии с результатом.

### **Реализация АЛУ**

АЛУ может быть реализована **с помощью различных схем**, таких как:

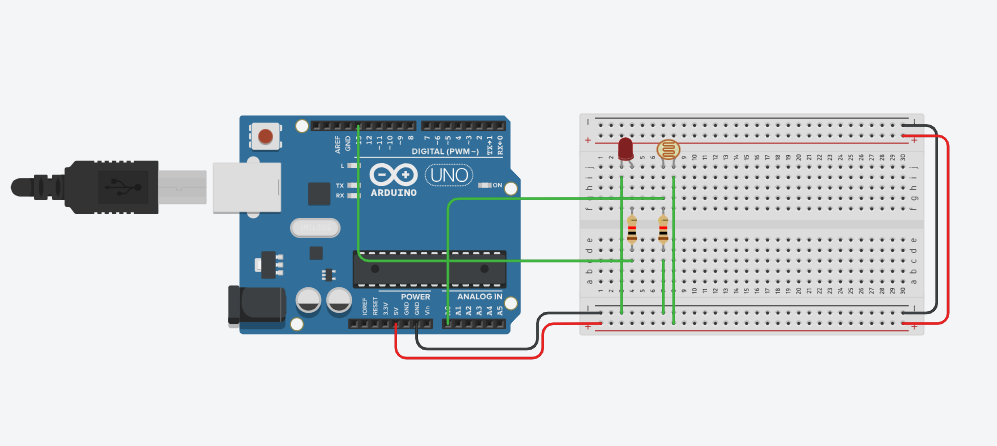
* **Сумматоры:** Сумматоры используются для выполнения сложения и вычитания.
* **Мультипликаторы:** Мультипликаторы используются для выполнения умножения.
* **Делители:** Делители используются для выполнения деления.
* **Логические вентили:** Логические вентили используются для выполнения логических операций.

Ссылка на Gemini:

https://docs.google.com/document/d/191B-NKt8nNp6Kr4jnse6\_JHAUsHC9DuutuDSAt9c8V8/edit?usp=sharing

Ссылка на ChatGpt: https://chatgpt.com/share/475a49b5-9d9c-4895-9096-bb5956618d75

3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, фоторезистора и светодиода. Написать программный код для работы схемы. При затемнении фоторезистора, светодиод включается



// Определение пинов

const int photoResistorPin = A0; // Фоторезистор подключен к аналоговому входу A0

const int ledPin = 13; // Светодиод подключен к цифровому выходу 13

// Переменные для хранения значений

int photoResistorValue = 0; // Значение с фоторезистора

int threshold = 500; // Пороговое значение для определения затемнения

void setup() {

// Настройка пина светодиода как выхода

pinMode(ledPin, OUTPUT);

// Настройка пина фоторезистора как входа

pinMode(photoResistorPin, INPUT);

// Инициализация последовательного порта для отладки

Serial.begin(9600);

}

void loop() {

// Чтение значения с фоторезистора

photoResistorValue = analogRead(photoResistorPin);

// Отладка: вывод значения фоторезистора в последовательный порт

Serial.print("Фоторезистор значение: ");

Serial.println(photoResistorValue);

// Проверка, если значение фоторезистора ниже порогового значения

if (photoResistorValue < threshold) {

// Включение светодиода

digitalWrite(ledPin, HIGH);

} else {

// Выключение светодиода

digitalWrite(ledPin, LOW);

}

// Задержка для стабильности чтения

delay(100);

}

Ссылка на проект:

https://www.tinkercad.com/things/jaWY8Q9Ys1Z-bilet-1?sharecode=YR-gxek9Q4TQVV82-npJ6NDvHr5mtGdrec7X4ahh\_vM

**Билет 2**

**1.Двоично — шестнадцатиричная система счисления**

Двоично-шестнадцатеричная система счисления – это способ представления шестнадцатеричных чисел с помощью двоичных цифр. В отличие от стандартной шестнадцатеричной системы, где используются цифры от 0 до 9 и буквы A-F, в двоично-шестнадцатеричной системе каждое шестнадцатеричное число (от 0 до F) кодируется своим уникальным 4-битным двоичным кодом



**Преимущества двоично-шестнадцатеричной системы:**

* **Компактность:** Для представления одного шестнадцатеричного числа требуется всего 4 бита, что меньше, чем в стандартной шестнадцатеричной системе (где для представления 16 чисел требуется 4 бита для каждой цифры).
* **Простота преобразования:** Легко преобразовать числа из двоичной системы в шестнадцатеричную и обратно.
* **Наглядность:** Легко понять, как двоичные цифры кодируют шестнадцатеричные значения.

**Недостатки двоично-шестнадцатеричной системы:**

* **Ограниченная распространенность:** Не так распространена, как двоичная или шестнадцатеричная системы, что может затруднить использование в некоторых приложениях.

### **Применение:**

Шестнадцатиричная система широко используется в компьютерной технике и программировании для представления адресов памяти, цветов в веб-дизайне (например, #FF5733), и при отладке программ, где часто встречаются длинные двоичные числа.

Таким образом, двоично-шестнадцатиричная система счисления — это удобный способ представления двоичных данных, который обеспечивает компактность и легкость чтения для людей, работающих с компьютерами и цифровыми системами.

Ссылка на Gemini: https://docs.google.com/document/d/1fFZ2s1SQ4mYEXp2M8yCrsJOGVFoUo\_8xPGBLKXIs0wo/edit?usp=sharing

Ссылка на ChatGpt:

https://chatgpt.com/share/475a49b5-9d9c-4895-9096-bb5956618d75

**2. Микроконтроллер. Описание.**

**Микроконтроллер** — это интегральная микросхема, предназначенная для управления различными устройствами. Он представляет собой компактную вычислительную систему, включающую в себя процессор, память и периферийные устройства.

### Основные компоненты микроконтроллера:

1. **Процессор (CPU)**:
   * Центральный процессор является «мозгом» микроконтроллера, выполняющим команды программы.
   * Он обрабатывает данные и управляет другими компонентами микроконтроллера.
2. **Память**:
   * **Постоянная память (ROM, Flash)**: используется для хранения прошивки, то есть программы, которую микроконтроллер выполняет.
   * **Оперативная память (RAM)**: используется для временного хранения данных, необходимых во время выполнения программы.
   * **Энергонезависимая память (EEPROM)**: используется для хранения данных, которые должны сохраняться при выключении питания.
3. **Периферийные устройства**:
   * **Цифровые и аналоговые входы/выходы (I/O)**: используются для взаимодействия с внешними устройствами, такими как датчики и приводы.
   * **Таймеры и счетчики**: позволяют выполнять операции по времени, такие как измерение длительности событий или создание задержек.
   * **Коммуникационные интерфейсы**: включают в себя UART, SPI, I2C и другие, которые позволяют микроконтроллеру обмениваться данными с другими устройствами.

**Микроконтроллеры программируются** с помощью специальных языков программирования, таких как C, C++, Assembly.

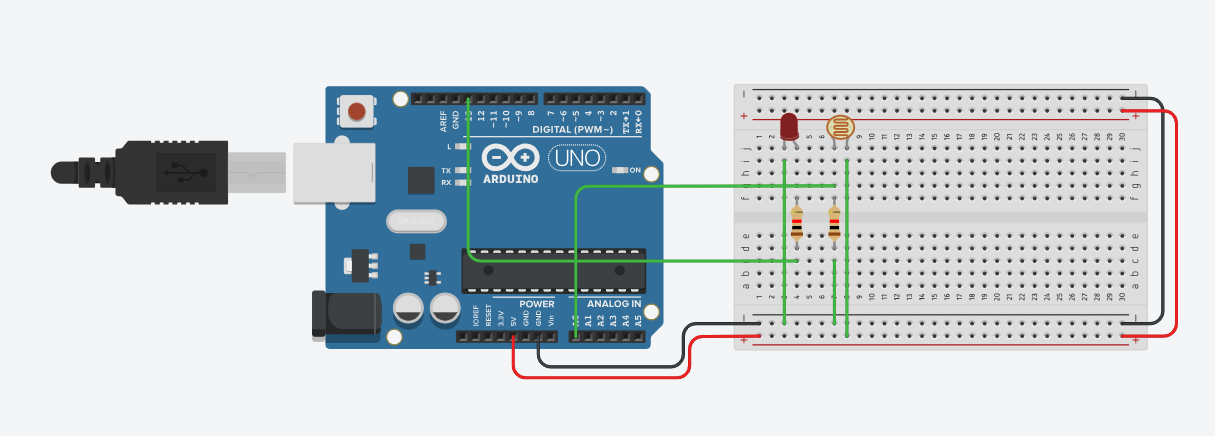
Микроконтроллеры являются ключевыми компонентами в современном мире электроники и встроенных систем. Они обеспечивают мощность и гибкость, необходимые для управления сложными устройствами, при этом оставаясь компактными и энергоэффективными. Благодаря своим многочисленным возможностям и применению, микроконтроллеры играют важную роль в повседневной жизни и технологическом прогрессе.

Ссылка на Gemini: https://docs.google.com/document/d/1\_gJJVZEg1zSngkYwmVHOm6xK7kT8iaUfqxPIJAqS6h8/edit?usp=sharing

Ссылка на ChatGpt:

https://chatgpt.com/share/475a49b5-9d9c-4895-9096-bb5956618d75

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, фоторезистора и светодиода. Написать программный код для работы схемы. При затемнении фоторезистора, светодиод выключается.**



// Определение пинов

const int photoResistorPin = A0; // Фоторезистор подключен к аналоговому входу A0

const int ledPin = 13; // Светодиод подключен к цифровому выходу 13

// Переменные для хранения значений

int photoResistorValue = 0; // Значение с фоторезистора

int threshold = 500; // Пороговое значение для определения затемнения

void setup() {

// Настройка пина светодиода как выхода

pinMode(ledPin, OUTPUT);

// Настройка пина фоторезистора как входа

pinMode(photoResistorPin, INPUT);

// Инициализация последовательного порта для отладки

Serial.begin(9600);

// Включение светодиода при старте

digitalWrite(ledPin, HIGH);

}

void loop() {

// Чтение значения с фоторезистора

photoResistorValue = analogRead(photoResistorPin);

// Отладка: вывод значения фоторезистора в последовательный порт

Serial.print("Фоторезистор значение: ");

Serial.println(photoResistorValue);

// Проверка, если значение фоторезистора ниже порогового значения

if (photoResistorValue < threshold) {

// Выключение светодиода

digitalWrite(ledPin, LOW);

} else {

// Включение светодиода

digitalWrite(ledPin, HIGH);

}

// Задержка для стабильности чтения

delay(100);

}

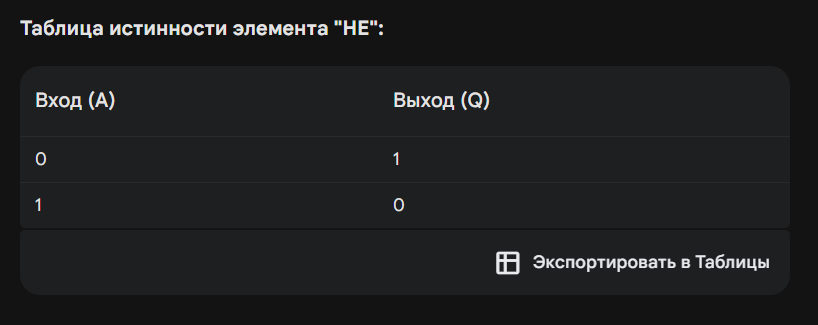
Ссылка на Проект:

https://www.tinkercad.com/things/hqG1gO36bxu-bilet-2?sharecode=92VcRIRL4ykdqROAABkLGNbx0PTLUH-Znu76PYouksw

**Билет 3**

**1.Логический элемент «НЕ». Графическое обозначение, таблица истинности.**

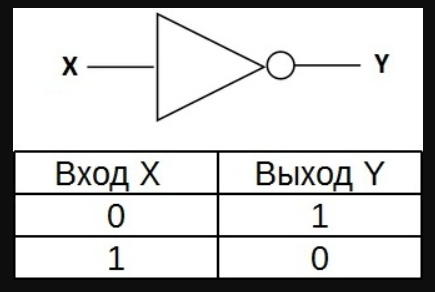
Логический элемент "НЕ" (инвертор) - это базовый элемент цифровой электроники, который выполняет операцию логического отрицания.



**Описание работы:**

Элемент "НЕ" имеет один вход (A) и один выход (Q). На выходе элемента (Q) всегда находится значение, противоположное значению на входе (A).

* Если на вход (A) подается напряжение логического нуля (0), то на выходе (Q) будет напряжение логической единицы (1).
* Если на вход (A) подается напряжение логической единицы (1), то на выходе (Q) будет напряжение логического нуля (0).



Ссылка на Gemini:

https://docs.google.com/document/d/1PEH-bUyzrgr6Zar8BQdseucdjw85Sm4Qe71lURzcgkU/edit?usp=sharing

Ссылка на ChatGpt:

https://chatgpt.com/share/475a49b5-9d9c-4895-9096-bb5956618d75

**2.Дещифратор на 2 входа. Графическое обозначение, таблица истинности**

**Дешифратор на 2 входа** – это электронная схема, предназначенная для преобразования двоичного кода, состоящего из 2 бит, в унитарный код, где на выходе будет активен только один из 4 выходов, соответствующий данному двоичному коду.

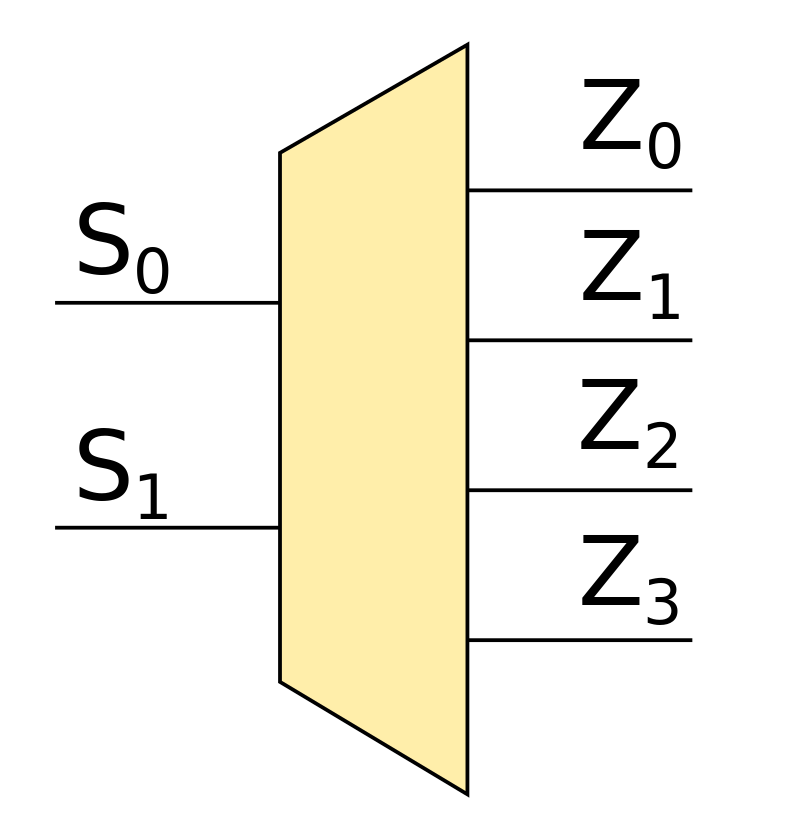
### 1. Описание работы

Дешифратор на 2 входа имеет 2 входных бита (A0 и A1) и 4 выходных канала (Q0, Q1, Q2, Q3).

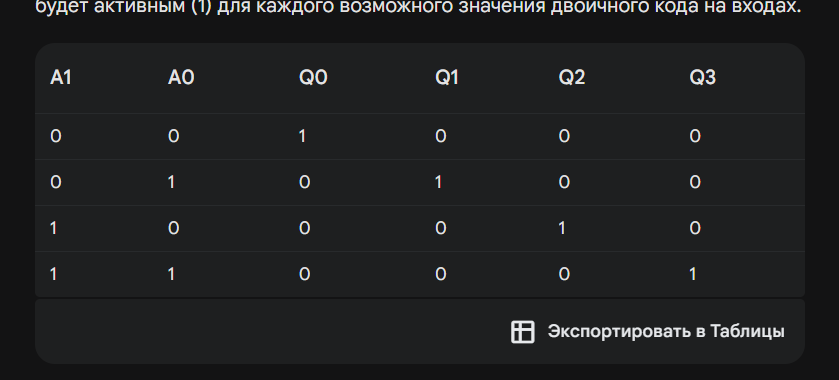
* **Входные биты:**
  + A0 - Младший значащий бит (LSB)
  + A1 - Старший значащий бит (MSB)
* **Выходные каналы:**
  + Q0 - Активен, если A0 = 0 и A1 = 0
  + Q1 - Активен, если A0 = 0 и A1 = 1
  + Q2 - Активен, если A0 = 1 и A1 = 0
  + Q3 - Активен, если A0 = 1 и A1 = 1

На каждом выходном канале может быть активным либо высокий уровень напряжения (логическая единица, 1), либо низкий уровень напряжения (логический ноль, 0). В зависимости от реализации дешифратора, активный уровень на выходе может быть как 1, так и 0.

**2. Графическое обозначение**



**Таблица истинности**



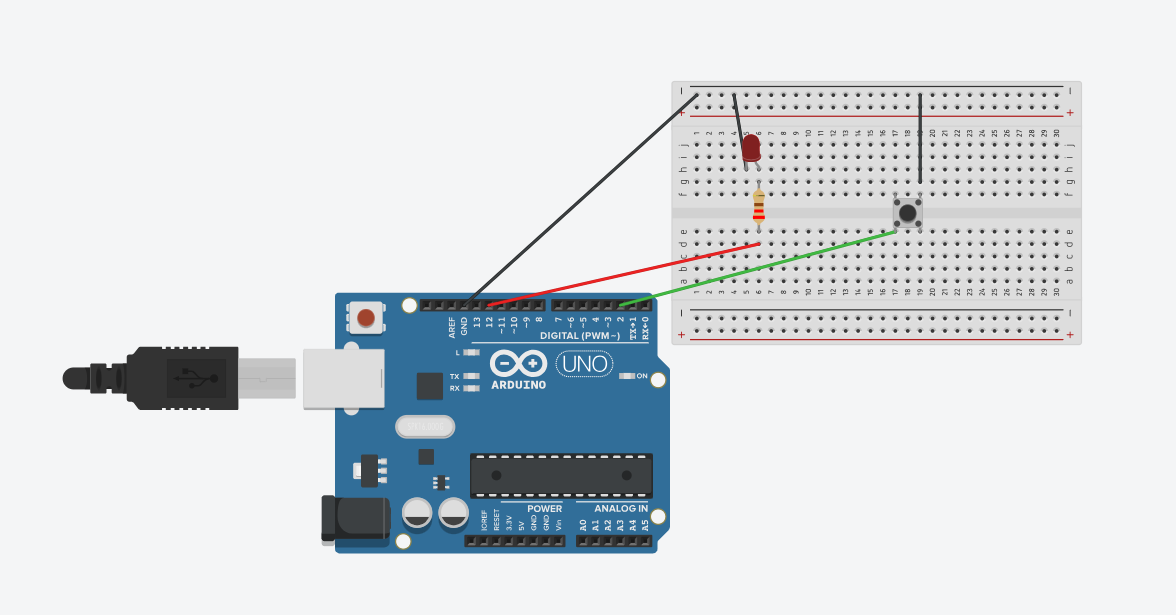
Ссылка на Gemini:

https://docs.google.com/document/d/1RQW\_YHKbCvrD5-CazAcuuFuU1Z3QPiS9qu0\_ft4k\_kM/edit

Ссылка на ChatGpt:

https://chatgpt.com/share/475a49b5-9d9c-4895-9096-bb5956618d75

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одной тактовой кнопки и светодиода. Написать программный код для работы схемы. При нажатии кнопки, светодиод включается.**



// constants won't change. They're used here to set pin numbers:

const int buttonPin = 2; // the number of the pushbutton pin

const int ledPin = 12; // the number of the LED pin

// variables will change:

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

// initialize the LED pin as an output:

pinMode(ledPin, OUTPUT);

// initialize the pushbutton pin as an input:

pinMode(buttonPin, INPUT\_PULLUP);

}

void loop() {

// read the state of the pushbutton value:

buttonState = digitalRead(buttonPin);

// check if the pushbutton is pressed.

if (buttonState == LOW) {

// turn LED on:

digitalWrite(ledPin, HIGH);

} else {

// turn LED off:

digitalWrite(ledPin, LOW);

}

}

Ссылка на Проект:

https://www.tinkercad.com/things/kSn0h6J7SKL-bilet-3?sharecode=fYWzaiOL0X7ZJQzUwYyjRVlRRlN-c3H7Q0LiMz-36xc

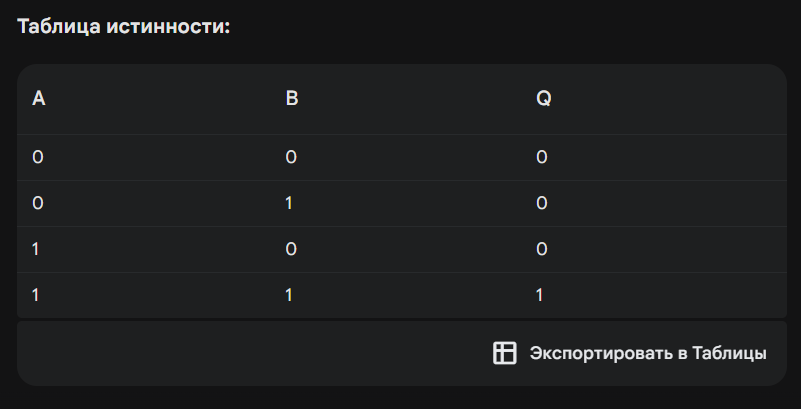
**Билет 4**

**Логический элемент «2 И». Графическое обозначение, таблица истинности.**

**Логический элемент «2 И» (И)**, также известный как **конъюнктор**, является базовым элементом цифровой электроники, который выполняет операцию логического умножения.

**Описание работы:**

* Элемент «2 И» имеет два входа (A и B) и один выход (Q).
* На выходе (Q) элемента будет значение логической единицы (1) только в том случае, если на обоих входах (A и B) одновременно будут значения логической единицы (1).
* Если на одном из входов (или на обоих) будет значение логического нуля (0), то на выходе (Q) будет значение логического нуля (0).



Ссылка на Gemini:

https://docs.google.com/document/d/1RQW\_YHKbCvrD5-CazAcuuFuU1Z3QPiS9qu0\_ft4k\_kM/edit

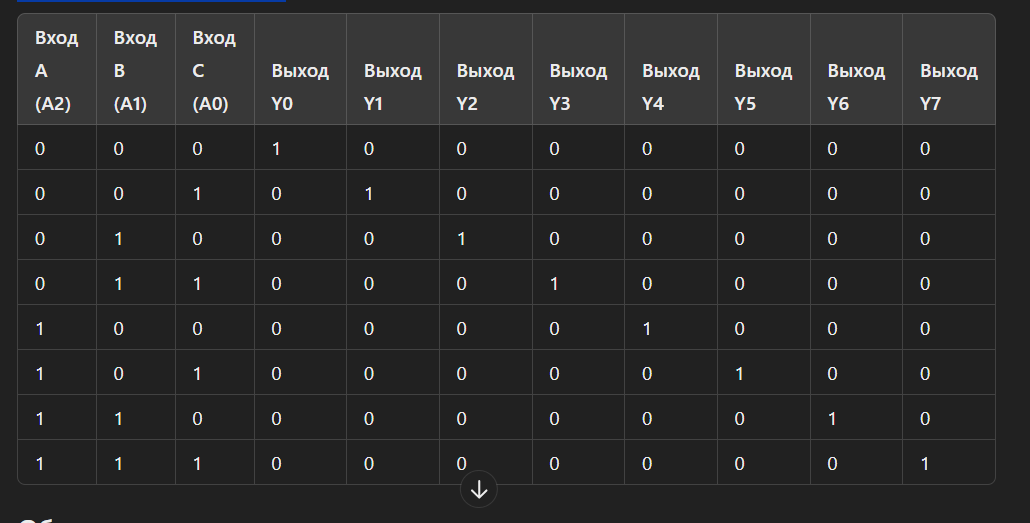
**2.Дещифратор на 3 входа. Графическое обозначение, таблица истинности.**

**Дешифратор** (декодер) — это цифровое логическое устройство, которое преобразует кодированную информацию из одного формата в другой. Дешифратор на 3 входа принимает 3-битный двоичный код и активирует один из 8 выходов, соответствующих этому коду.

### **Принцип работы:**

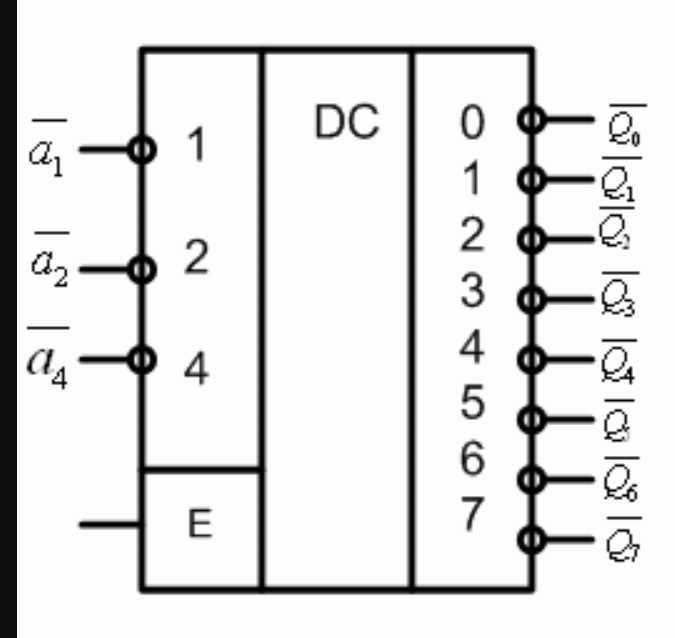
Дешифратор с 3 входами имеет 3 входных линии и 8 выходных линий. В зависимости от комбинации входных сигналов, только один из выходов будет активным (логическая 1), а остальные — неактивными (логический 0).

**Таблица истинности:**



**Графическое обозначение**

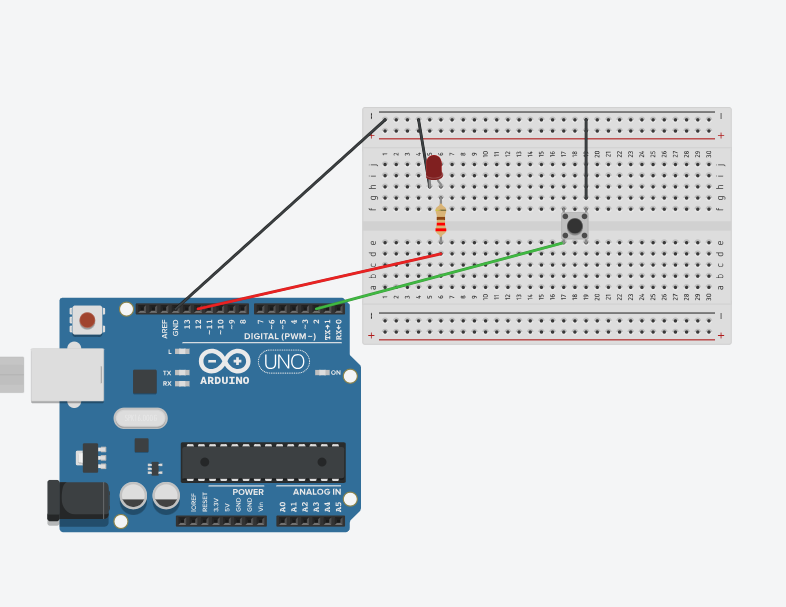
Дешифратор часто обозначается как прямоугольник с входами и выходами, где на входах указываются двоичные коды, а на выходах — соответствующие линии.



Ссылка на ChatGpt:

https://chatgpt.com/share/475a49b5-9d9c-4895-9096-bb5956618d75

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одной тактовой кнопки и светодиода. Написать программный код для работы схемы. При нажатии кнопки, светодиод выключается.**



// constants won't change. They're used here to set pin numbers:

const int buttonPin = 2; // the number of the pushbutton pin

const int ledPin = 12; // the number of the LED pin

// variables will change:

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

// initialize the LED pin as an output:

pinMode(ledPin, OUTPUT);

// initialize the pushbutton pin as an input:

pinMode(buttonPin, INPUT\_PULLUP);

}

void loop() {

// read the state of the pushbutton value:

buttonState = digitalRead(buttonPin);

// check if the pushbutton is pressed.

if (buttonState == LOW) {

// turn LED on:

digitalWrite(ledPin, LOW);

} else {

// turn LED off:

digitalWrite(ledPin, HIGH);

}

}

Ссылка на Проект:

https://www.tinkercad.com/things/5GglAEFM4ab-bilet-4?sharecode=2VhiIVE8XWnOgPy5wZKUiJW5bzDPRlyzC5EVWjzkyzw

**Билет 5**

я больше не буду писать ответы на первый вопрос, они все одинаковы если знать стандартные логические операторы и то как они работают то нормально ответ можно придумать

**2.Дещифратор на 4 входа. Графическое обозначение, таблица истинности.**

Дешифратор на 4 входа (или декодер на 4 входа) является комбинационным логическим устройством, которое имеет 4 входа и 16 выходов. Каждый из 16 выходов активируется только при определенной комбинации входных сигналов. Обычно это означает, что только один из 16 выходов будет активен в любой момент времени.

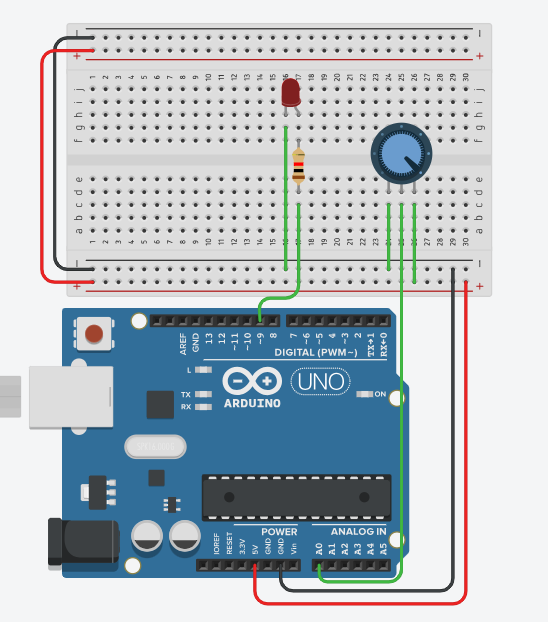
Например, если входные сигналы I3 = 0, I2 = 1, I1 = 0, I0 = 1, то только выход O5 будет активным (равен 1), а все остальные выходы O0-O15 будут равны 0.

Ссылка на чат потому что таблица пиздец большая : https://chatgpt.com/share/9c3416c8-2236-4b23-901a-849882c8c86d

**Унитарный код** - это код, в котором **только один выход** имеет значение **1**, а все остальные **0**.

**Дешифратор на 4 входа** используется для **декодирования адресов** в различных цифровых устройствах. ( я не знаю насколько этот ответ полный поскольку тут можно просто про дешифраторы рассказать ну или стих блять я ебу , вот еще копилот но он тоже самое написал https://sl.bing.net/eEj1VD4CBw)

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, резистора переменного сопротивления и светодиода. Написать программный код для работы схемы. При вращении ручки резистора переменного сопротивления, изменяется яркость свечения светодиода**



// Определение пинов

const int potentiometerPin = A0; // Переменный резистор подключен к аналоговому входу A0

const int ledPin = 9; // Светодиод подключен к цифровому выходу 9 (должен поддерживать ШИМ)

// Переменные для хранения значений

int potentiometerValue = 0; // Значение с переменного резистора

int brightness = 0; // Значение яркости светодиода

void setup() {

// Настройка пина светодиода как выхода

pinMode(ledPin, OUTPUT);

// Настройка пина переменного резистора как входа

pinMode(potentiometerPin, INPUT);

// Инициализация последовательного порта для отладки

Serial.begin(9600);

}

void loop() {

// Чтение значения с переменного резистора

potentiometerValue = analogRead(potentiometerPin);

// Преобразование значения переменного резистора в значение яркости (0-255)

brightness = map(potentiometerValue, 0, 1023, 0, 255);

// Установка яркости светодиода

analogWrite(ledPin, brightness);

// Отладка: вывод значения яркости в последовательный порт

Serial.print("Яркость: ");

Serial.println(brightness);

// Задержка для стабильности чтения

delay(10);

}

Ссылка на проект: https://www.tinkercad.com/things/9HqonQOfFWn-bilet-5?sharecode=Fgp11woZe\_ssMl113FAyQn5hTRXTpBGq9g-dnvJvcgs

**Билет 6**

**2.Дещифратор для семисегментного индикатора. Графическое обозначение, таблица истинности.**

**Дешифратор для семисегментного индикатора** – это электронная схема, которая преобразует двоичный код, обычно в формате BCD (двоично-десятичный код), в сигналы, необходимые для управления семисегментным индикатором.

Семисегментный индикатор состоит из семи светодиодов, расположенных в форме цифры. Дешифратор принимает на вход 4-битный двоичный код (BCD) и активирует соответствующие сегменты индикатора, отображая заданную цифру.

### **Работа дешифратора**

* **Вход:** 4-битный двоичный код (BCD)
* **Выход:** 7 сигналов для управления сегментами семисегментного индикатора
* **Функция:** Преобразует BCD-код в сигналы для отображения цифры на индикаторе

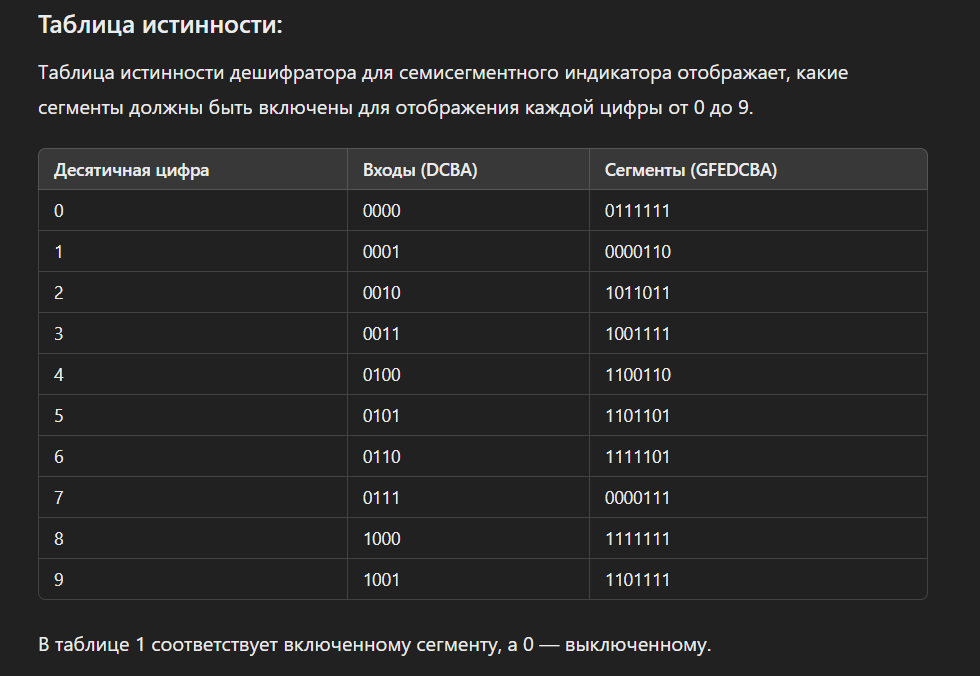
**Пример:**

* **Вхід:** 0101 (BCD)
* **Вихід:** 0 0 0 1 1 0 0 (активированы сегменты b, c, d, f, g)
* **Отображение:** цифра "5"

### **Типы дешифраторов**

Существует несколько типов дешифраторов для семисегментных индикаторов, которые отличаются по количеству входов, выходов, функциям и технологии реализации.

* **Дешифраторы BCD-семисегментные:** 4-битных BCD-входа и 7-сегментных выходов.
* **Дешифраторы двоично-семисегментные:** 3-битных двоичных входа и 7-сегментных выходов.
* **Дешифраторы с дополнительными функциями:** декодирование букв, управление точкой, гашение неиспользуемых сегментов.



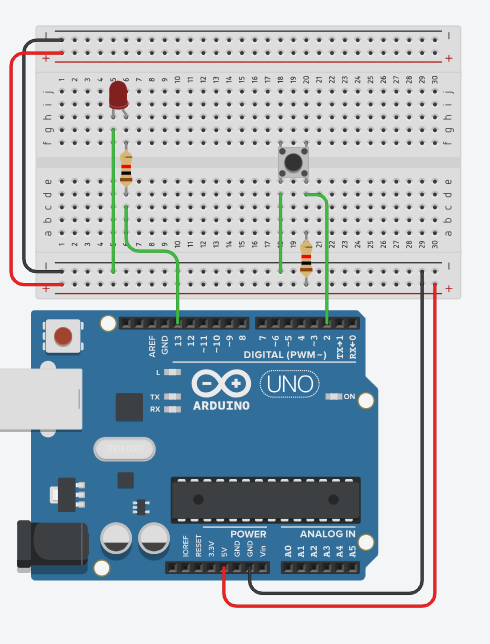
Ссылка на Gemini :

https://docs.google.com/document/d/1Fk8LnEiZUFwlf8-zIaoxdg40WmXzbgKVjpLD87QLAbU/edit?usp=sharing

Ссылка на Copilot:

https://sl.bing.net/eTVPrixo8rY

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одной тактовой кнопки и светодиода. Написать программный код для работы схемы. При нажатии кнопки, светодиод включается, при следующем нажатии кнопки светодиод выключается.**



// Определение пинов

const int buttonPin = 2; // Кнопка подключена к цифровому входу 2

const int ledPin = 13; // Светодиод подключен к цифровому выходу 13

// Переменные для хранения состояния кнопки и светодиода

int buttonState = 0; // Текущее состояние кнопки

int lastButtonState = 0; // Предыдущее состояние кнопки

bool ledState = LOW; // Состояние светодиода

void setup() {

// Настройка пина кнопки как входа

pinMode(buttonPin, INPUT);

// Настройка пина светодиода как выхода

pinMode(ledPin, OUTPUT);

// Инициализация состояния светодиода

digitalWrite(ledPin, ledState);

}

void loop() {

// Чтение текущего состояния кнопки

buttonState = digitalRead(buttonPin);

// Проверка, если кнопка была нажата (переход из LOW в HIGH)

if (buttonState == HIGH && lastButtonState == LOW) {

// Инвертирование состояния светодиода

ledState = !ledState;

// Установка нового состояния светодиода

digitalWrite(ledPin, ledState);

}

// Сохранение текущего состояния кнопки для следующего цикла

lastButtonState = buttonState;

}

Ссылка на проект :

https://www.tinkercad.com/things/7LpGveNkjv8-bilet-6?sharecode=VyNb7mPf02\_z\_nvrvH3lAKq-KQUc1xe3IU6mke-W0aE

**Билет 7**

**2.Шифратор. Графическое обозначение, таблица истинности.**

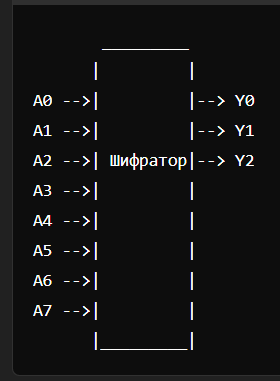
**Шифратор** (или кодер) — это цифровое устройство, которое преобразует активный входной сигнал в кодированный выходной сигнал. Шифратор имеет несколько входов и обычно меньшее количество выходов. Он работает в противоположном направлении относительно дешифратора.

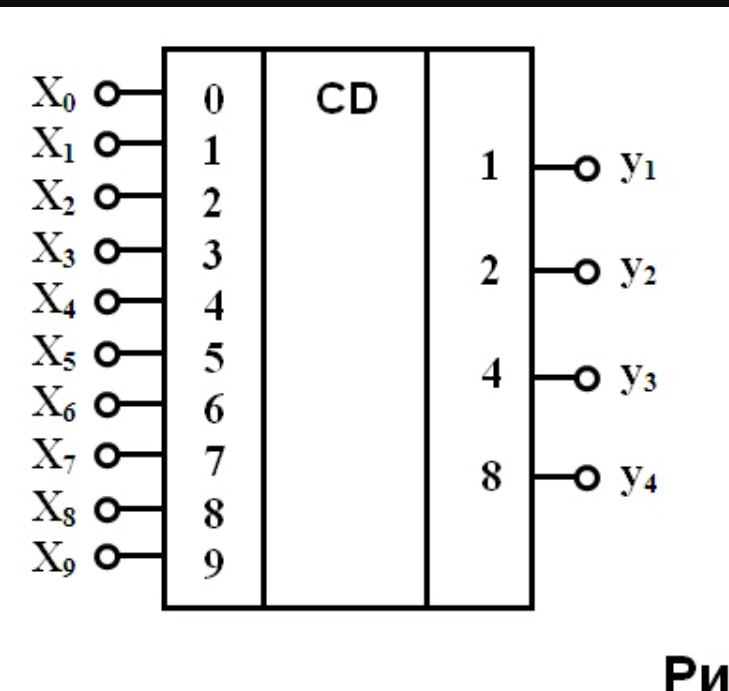
В общем случае, шифратор принимает на вход **сигнал**, представленный в одном коде (например, двоичном, десятичном, позиционном), и преобразует его в **выходной сигнал**, представленный в другом коде (например, двоичном, позиционном, n-ичном).

### Функции шифратора:

* **Кодирование:** Преобразование входного сигнала в выходной код.
* **Сжатие данных:** Уменьшение количества битов, необходимых для представления информации.
* **Обнаружение и исправление ошибок:** Включение дополнительных битов в выходной код для обнаружения и исправления ошибок при передаче данных.
* **Стандартизация:** Преобразование входного сигнала в стандартный формат, совместимый с другими устройствами.

Шифратор обычно изображается как прямоугольник с множеством входов и меньшим числом выходов. На входах указываются числа или символы, которые шифратор может принять, а на выходах — кодированные значения.







### Логические уравнения:

Для 8-3 шифратора логические уравнения для выходов могут быть записаны следующим образом:

* Y2=A4+A5+A6+A7Y2 = A4 + A5 + A6 + A7Y2=A4+A5+A6+A7
* Y1=A2+A3+A6+A7Y1 = A2 + A3 + A6 + A7Y1=A2+A3+A6+A7
* Y0=A1+A3+A5+A7Y0 = A1 + A3 + A5 + A7Y0=A1+A3+A5+A7

Ссылка на Gemini :

https://docs.google.com/document/d/1NyHmW9nEn3KxmGG4ya9kAYAVj3wa4qBqrArn43C0lcw/edit?usp=sharing

Ссылка на Chat gpt :

https://chatgpt.com/share/475a49b5-9d9c-4895-9096-bb5956618d75

Ссылка на Copilot :

https://sl.bing.net/kxJoz7WPZtc

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одной тактовой кнопки и двух светодиодов. Написать программный код «Т - триггер» для работы схемы**

Переключатель и т триггер это одно и тоже

Ссылка на проект :

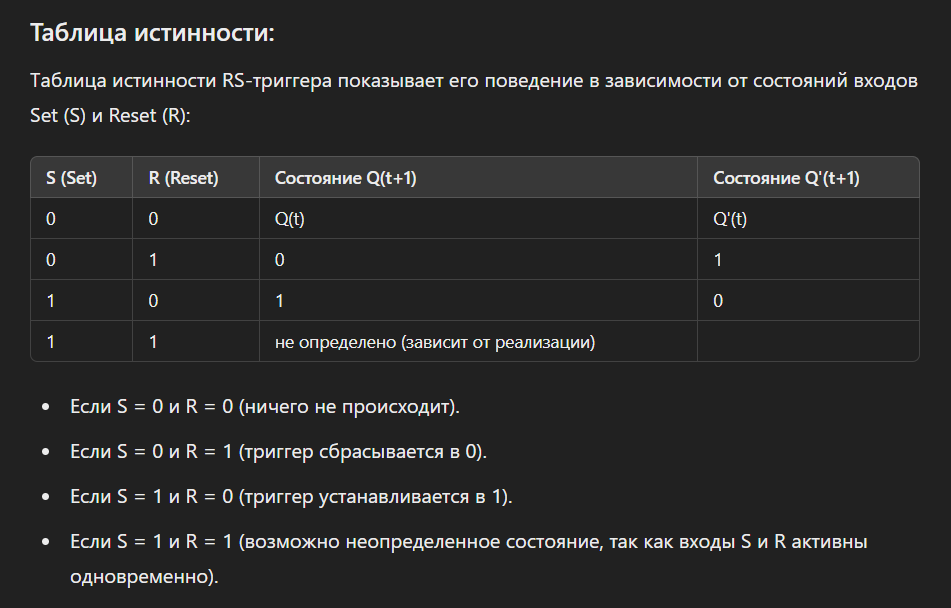
https://www.tinkercad.com/things/7LpGveNkjv8-bilet-6?sharecode=VyNb7mPf02\_z\_nvrvH3lAKq-KQUc1xe3IU6mke-W0aE

**Билет 8**

**RS-триггер** (Reset-Set триггер) — это базовый цифровой элемент, который состоит из двух входов: Set (установка) и Reset (сброс). Он имеет два стабильных состояния и используется для хранения одного бита информации.

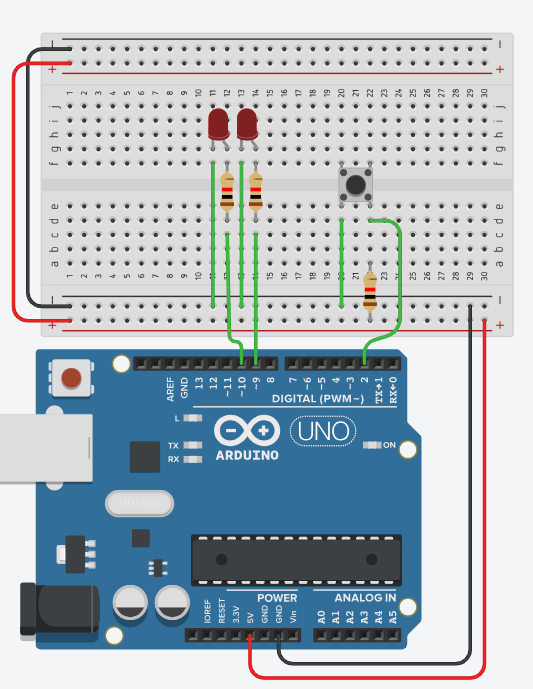
**Описание работы:**

* RS-триггер имеет два информационных входа: **R (Reset)** и **S (Set)**, и один выход (Q).
* **Вход R** (сброс) при подаче на него **логической единицы (1)** устанавливает выход Q в **логический ноль (0)**, независимо от состояния на другом входе (S).
* **Вход S** (установка) при подаче на него **логической единицы (1)** устанавливает выход Q в **логическую единицу (1)**, независимо от состояния на другом входе (R).
* **Если на оба входа одновременно подать логические единицы (1)**, то состояние выхода (Q) **не определено**. В зависимости от конкретной реализации триггера, его состояние может сохранить прежнее значение, перейти в ноль или единицу.



Ссылка на Gemini: https://docs.google.com/document/d/1hOmDPfWuH5LFAM3QJDlkUiQDLXAEXfx9L\_yD6SzXvm0/edit?usp=sharing

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одной тактовой кнопки и двух светодиодов. Написать программный код для работы схемы. При нажатии кнопки, первый светодиод включен, второй светодиод выключен, при следующем нажатии кнопки первый светодиод выключен, второй светодиод включен.**



const int buttonPin = 2; // номер пина, к которому подключена кнопка

const int ledPin1 = 9; // номер пина, к которому подключен первый светодиод

const int ledPin2 = 10; // номер пина, к которому подключен второй светодиод

int buttonState = 0; // текущее состояние кнопки

int lastButtonState = 0; // предыдущее состояние кнопки

int ledState = 0; // текущее состояние светодиодов (0 - первый включен, второй выключен; 1 - первый выключен, второй включен)

void setup() {

pinMode(buttonPin, INPUT); // устанавливаем пин кнопки как вход

pinMode(ledPin1, OUTPUT); // устанавливаем пин первого светодиода как выход

pinMode(ledPin2, OUTPUT); // устанавливаем пин второго светодиода как выход

// начинаем с первого светодиода включенного, второго выключенного

digitalWrite(ledPin1, HIGH);

digitalWrite(ledPin2, LOW);

}

void loop() {

// считываем состояние кнопки

buttonState = digitalRead(buttonPin);

// проверяем, изменилось ли состояние кнопки

if (buttonState != lastButtonState) {

// если кнопка была нажата

if (buttonState == HIGH) {

// переключаем состояние светодиодов

ledState = !ledState;

if (ledState == 0) {

digitalWrite(ledPin1, HIGH);

digitalWrite(ledPin2, LOW);

} else {

digitalWrite(ledPin1, LOW);

digitalWrite(ledPin2, HIGH);

}

}

// ждём некоторое время для устранения дребезга кнопки

delay(50);

}

// сохраняем текущее состояние кнопки для сравнения в следующем цикле

lastButtonState = buttonState;

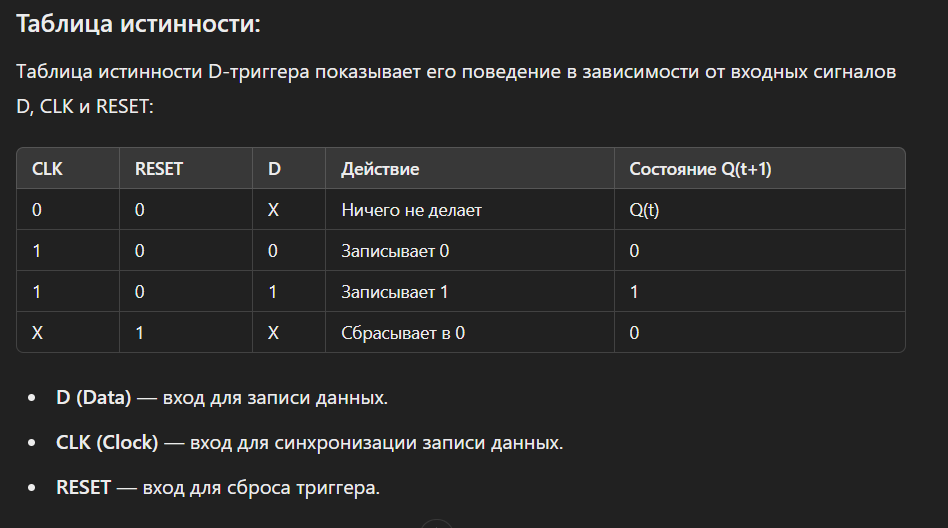
}

Ссылка на проект: https://www.tinkercad.com/things/i4LqMTcQpLk-bilet-8?sharecode=v0ypgruzjiuh1gVuh5vTFzJS31fgviBCUH4rReBxr7Q

**Билет 9**

2.D - триггер. Графическое обозначение, таблица истинности.

**D-триггер** является одним из базовых цифровых элементов, который используется для хранения одного бита информации. Он имеет один вход (Data или D) для записи информации и два управляющих входа: Clock (Синхронизация) и Reset (Сброс).



### Принцип работы:

* При установленном сигнале на CLK (обычно на фронте или спаде импульса), D-триггер сохраняет значение сигнала D на выходе Q.
* Если CLK = 0 или RESET = 1, триггер не меняет состояние, оставаясь в текущем состоянии (сохраняется последнее значение D).
* Если RESET = 1, то триггер сбрасывается в 0, независимо от значения D и CLK.

**Пример:**

* **D = 0, C = 0:** Q сохраняет свое текущее значение.
* **D = 1, C = 0:** Q сохраняет свое текущее значение.
* **D = 0, C = 1 (перепад):** Q = 0 (запись 0).
* **D = 1, C = 1 (перепад):** Q = 1 (запись 1).
* **D = 1, C = 0:** Q = 1 (сохраняет значение, записанное по перепаду).

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, двух тактовых кнопок и двух светодиодов. Написать программный код «RS - триггер» для работы схемы.**

Ссылка на проект: https://www.tinkercad.com/things/9IUvYZ90PgD-daring-inari?sharecode=4Q4eTETEmzQC\_rqVK8U\_goNcdDLnUI4uI5JJAb-Zc\_U

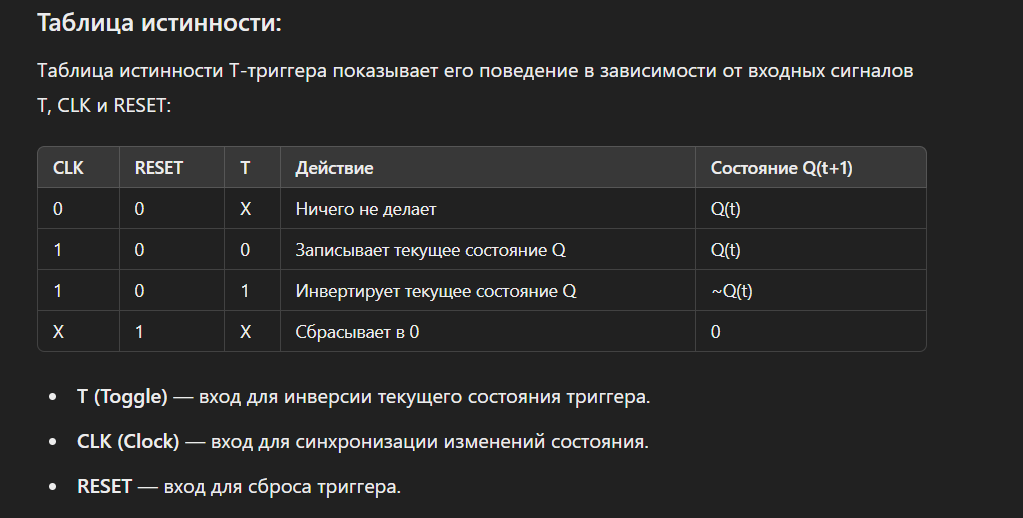
**Билет 10**

**2.T - триггер. Графическое обозначение, таблица истинности.**

**T-триггер** является одним из типов цифровых триггеров, который может работать в режимах захвата, хранения и инверсии состояния. Он имеет один вход (T) для управления и два управляющих входа: Clock (Синхронизация) и Reset (Сброс).

**Описание работы:**

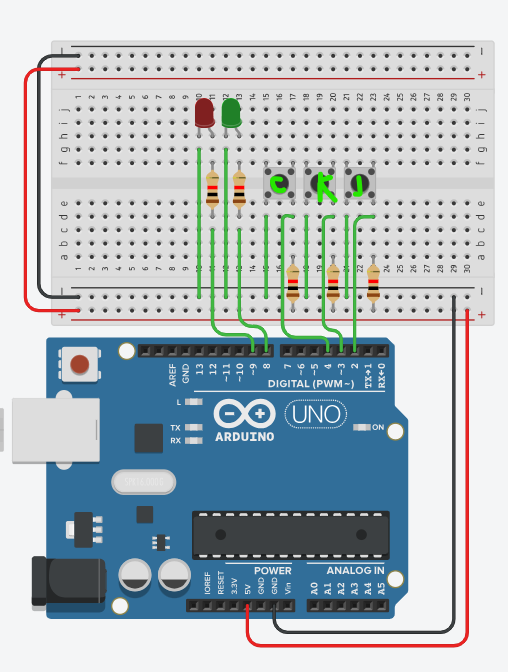
* T-триггер имеет один информационный вход (T) и один выход (Q).
* **По каждому перепаду (смене уровня) синхросигнала (C)** на **входе T** значение на **выходе Q** **инвертируется** (меняется на противоположное).
* **Если на входе T нет перепадов**, то **выход Q сохраняет свое текущее значение**.



Ссылка на Gemini : https://docs.google.com/document/d/1idr5FuvVXW4wID3zxuGuHsRa3gnRr7s\_cY71XV9DWaY/edit?usp=sharing

Cсылка на Copilot: https://sl.bing.net/gyxh3vj4P9M

**3.онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, трех тактовых кнопок и двух светодиодов. Написать программный код «JK - триггер» для работы схемы**



// Определение пинов

const int pinJ = 2; // Пин для сигнала J

const int pinK = 3; // Пин для сигнала K

const int pinClock = 4; // Пин для сигнала Clock

const int pinQ = 8; // Пин для выхода Q

const int pinNotQ = 9; // Пин для инверсного выхода Q

// Переменные для хранения состояния выходов Q и !Q

bool Q = false;

bool notQ = true;

// Переменные для хранения предыдущего состояния Clock

bool lastClockState = LOW;

void setup() {

// Настройка пинов

pinMode(pinJ, INPUT\_PULLUP); // Вход для сигнала J с подтягивающим резистором

pinMode(pinK, INPUT\_PULLUP); // Вход для сигнала K с подтягивающим резистором

pinMode(pinClock, INPUT\_PULLUP); // Вход для сигнала Clock с подтягивающим резистором

pinMode(pinQ, OUTPUT); // Выход для Q

pinMode(pinNotQ, OUTPUT); // Выход для инверсного Q

// Инициализация начального состояния выходов

digitalWrite(pinQ, Q);

digitalWrite(pinNotQ, notQ);

}

void loop() {

// Чтение состояния входов

bool jState = digitalRead(pinJ) == LOW; // Сигнал активен, когда кнопка нажата

bool kState = digitalRead(pinK) == LOW; // Сигнал активен, когда кнопка нажата

bool clockState = digitalRead(pinClock) == LOW; // Сигнал активен, когда кнопка нажата

// Обнаружение фронта синхронизации

if (clockState && !lastClockState) {

// Если обнаружен фронт синхронизации (переход из LOW в HIGH)

if (jState && !kState) {

// Если J=1 и K=0, устанавливаем Q в 1

Q = true;

notQ = false;

} else if (!jState && kState) {

// Если J=0 и K=1, устанавливаем Q в 0

Q = false;

notQ = true;

} else if (jState && kState) {

// Если J=1 и K=1, инвертируем Q

Q = !Q;

notQ = !notQ;

}

// Если J=0 и K=0, сохраняем текущее состояние Q

}

// Обновление предыдущего состояния Clock

lastClockState = clockState;

// Обновление выходов

digitalWrite(pinQ, Q);

digitalWrite(pinNotQ, notQ);

}

Ссылка на проект: https://www.tinkercad.com/things/eWyAqdLCVY5-jk?sharecode=Pvioikc1TjXhtehSxjCIxB3EyhsijNo6Z97VgJ3EBqE

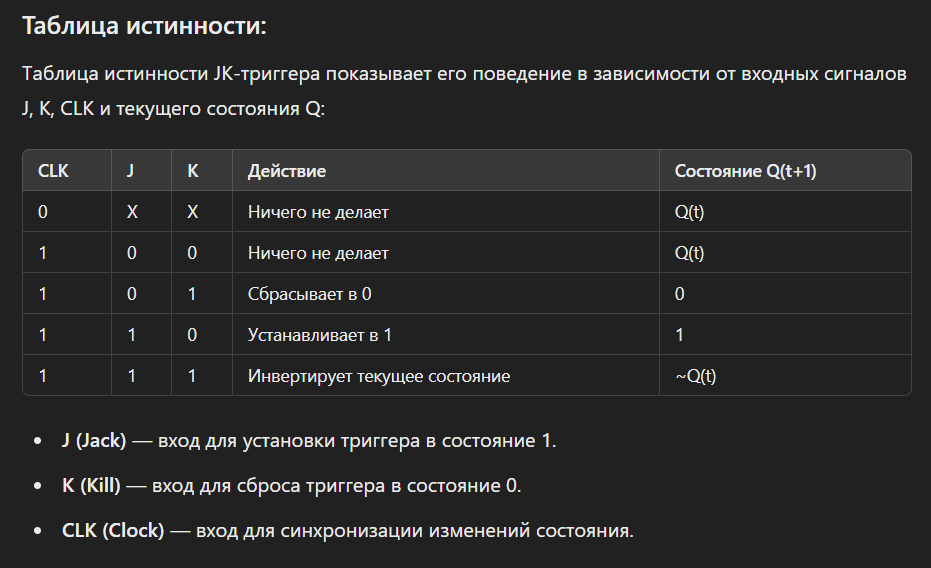
**Билет 11**

**2.JK - триггер. Графическое обозначение, таблица истинности.**

**JK-триггер** является ещё одним типом цифрового триггера, который имеет два управляющих входа (J и K) и один вход для синхронизации (Clock). Он способен выполнять операции установки (Set), сброса (Reset), и инверсии текущего состояния.

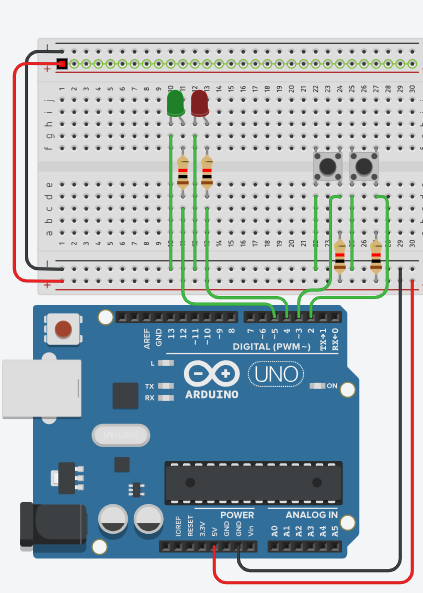
**Описание работы:**

* JK-триггер имеет два информационных входа (J и K) и один выход (Q).
* **Вход J** (установка) при подаче на него **логической единицы (1)** **устанавливает** выход Q в **единицу (1)** **по перепаду синхросигнала (C)**.
* **Вход K** (сброс) при подаче на него **логической единицы (1)** **сбрасывает** выход Q в **ноль (0)** **по перепаду синхросигнала (C)**.
* **Одновременная подача логических единиц (1) на оба входа (J и K)** приводит к **инвертированию** выходного значения Q **по перепаду синхросигнала (C)**.
* **Если на входах J и K нет перепадов**, то **выход Q сохраняет свое текущее значение**.



Ссылка на gemini: https://docs.google.com/document/d/1TqPDnCgX6C-Dhl9uPNX40fDftcPh15KuCGkTtNtiu6Q/edit

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, двух тактовых кнопок и двух светодиодов. Написать программный код «D - триггер» для работы схемы**



// Определяем пины для кнопок и светодиодов

const int buttonDPin = 2; // Пин для кнопки D

const int buttonClockPin = 3; // Пин для кнопки Clock

const int ledQPin = 4; // Пин для светодиода Q

const int ledNotQPin = 5; // Пин для светодиода not Q

// Переменные для хранения состояния кнопок

int buttonDState = LOW;

int buttonClockState = LOW;

int lastButtonClockState = LOW;

// Переменные для хранения состояния триггера

int Q = LOW;

int notQ = HIGH;

void setup() {

// Устанавливаем пины кнопок в режим ввода

pinMode(buttonDPin, INPUT);

pinMode(buttonClockPin, INPUT);

// Устанавливаем пины светодиодов в режим вывода

pinMode(ledQPin, OUTPUT);

pinMode(ledNotQPin, OUTPUT);

}

void loop() {

// Чтение состояния кнопок

buttonDState = digitalRead(buttonDPin);

buttonClockState = digitalRead(buttonClockPin);

// Обработка нажатия кнопки Clock

if (buttonClockState == HIGH && lastButtonClockState == LOW) {

// Если нажата кнопка Clock, обновляем состояние триггера

Q = buttonDState;

notQ = !buttonDState;

// Обновляем состояние светодиодов

digitalWrite(ledQPin, Q);

digitalWrite(ledNotQPin, notQ);

}

// Сохранение последнего состояния кнопки Clock

lastButtonClockState = buttonClockState;

}

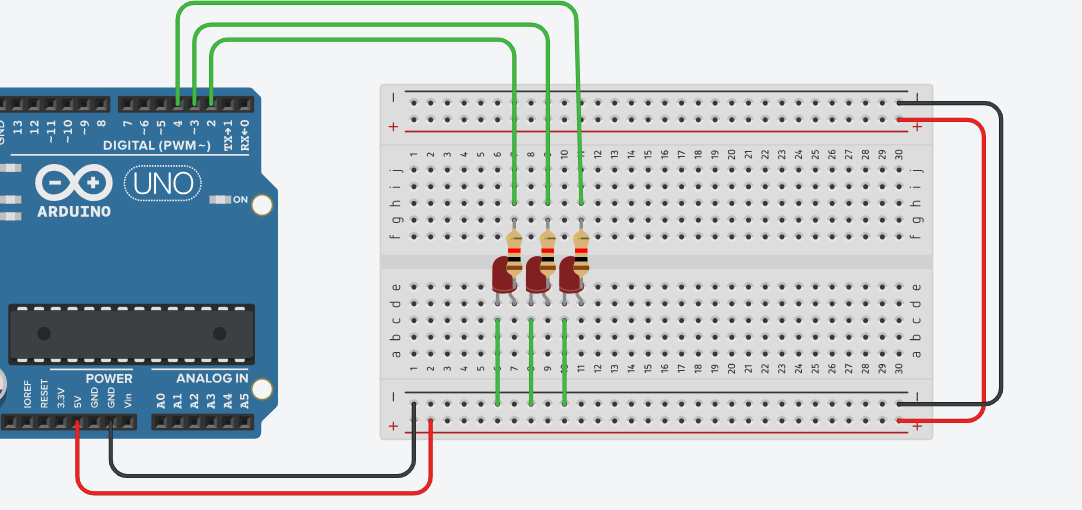
Ссылка на проект : https://www.tinkercad.com/things/hR7neYLy21N-copy-of-d-?sharecode=fITNiBvqM7bSZXRj31VhTLRD3EH4KeBbyzBaG9tmpKo

**Билет 12**

**2.Параллельный 4-х разрядный регистр. Графическое обозначение.**

Параллельный 4-разрядный регистр – это устройство, которое может одновременно записывать или считывать данные по всем разрядам. Такой регистр состоит из 4-х триггеров, каждый из которых хранит один бит данных.

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, трех светодиодов. Написать программный код «Последовательное включение одного светодиода «Бегущий огонь»» для работы схемы**



// Определяем пины, к которым подключены светодиоды

const int ledPins[] = {2, 3, 4}; // Пины для 3 светодиодов

const int numLeds = 3; // Количество светодиодов

const int delayTime = 500; // Время задержки в миллисекундах

void setup() {

// Устанавливаем пины в режим вывода

for (int i = 0; i < numLeds; i++) {

pinMode(ledPins[i], OUTPUT);

}

}

void loop() {

// Последовательно включаем один светодиод

for (int i = 0; i < numLeds; i++) {

// Выключаем все светодиоды

for (int j = 0; j < numLeds; j++) {

digitalWrite(ledPins[j], LOW);

}

// Включаем текущий светодиод

digitalWrite(ledPins[i], HIGH);

// Ждем перед следующим шагом

delay(delayTime);

}

}

Ссылка на проект: https://www.tinkercad.com/things/hIQB0QsFnZz-bilet-12?sharecode=XVPte3xgamQw0EotOJhcJfW8KiMe7VULK5J4Sg0vdRA

**Билет 13**

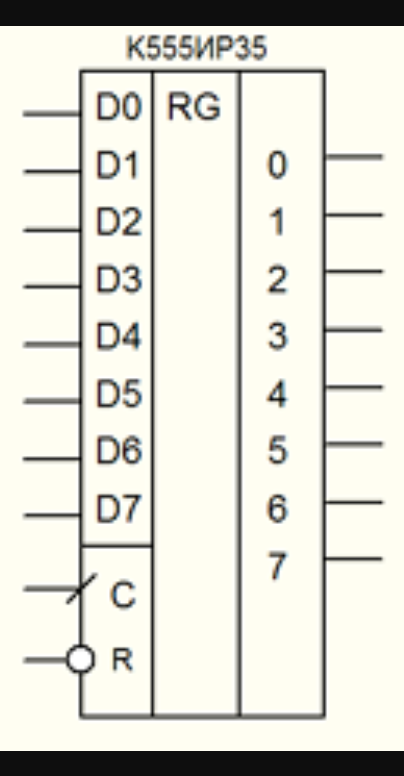
**2.Параллельный 8-ми разрядный регистр. Графическое обозначение.**

Параллельный 8-разрядный регистр используется для одновременной записи и считывания 8 бит данных. Как и 4-разрядный регистр, он состоит из триггеров (в данном случае 8), каждый из которых хранит один бит.

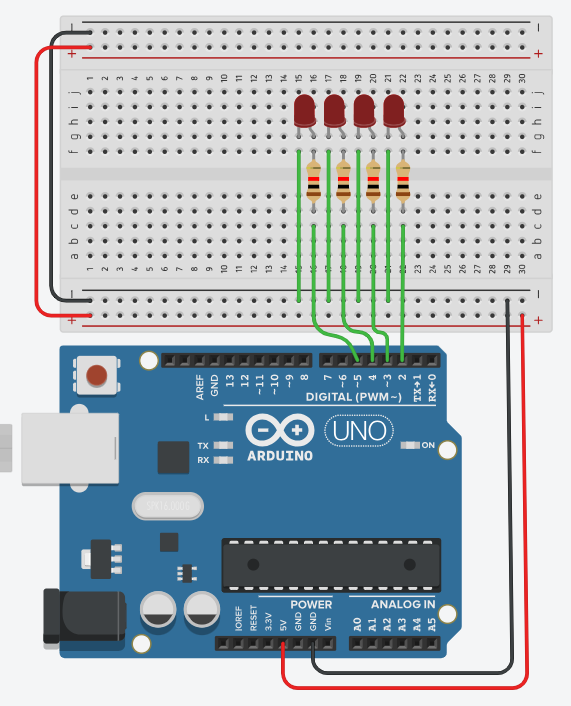
### Описание работы

1. **Запись данных**: При поступлении тактового сигнала (CLK) данные с входов D0-D7 записываются в регистр.
2. **Хранение данных**: Записанные данные хранятся в регистре до следующего тактового импульса или до сброса.
3. **Вывод данных**: Состояние регистров можно считать с выходов Q0-Q7.
4. **Сброс данных**: При поступлении сигнала сброса (RST) все разряды регистра устанавливаются в нулевое состояние.

Параллельные 8-разрядные регистры широко применяются в цифровой электронике, например, в микропроцессорах для хранения временных данных, в интерфейсах ввода-вывода для передачи байтов информации и в других системах, где требуется быстрая и синхронная передача и хранение данных.



**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, четырех светодиодов. Написать программный код «Последовательное «движение» двух включенных светодиодов» для работы схемы**



// Определяем пины, к которым подключены светодиоды

const int ledPins[] = {2, 3, 4, 5}; // Пины для 4 светодиодов

const int numLeds = 4; // Количество светодиодов

const int delayTime = 500; // Время задержки в миллисекундах

void setup() {

// Устанавливаем пины в режим вывода

for (int i = 0; i < numLeds; i++) {

pinMode(ledPins[i], OUTPUT);

}

}

void loop() {

// Последовательно включаем два светодиода

for (int i = 0; i < numLeds; i++) {

// Выключаем все светодиоды

for (int j = 0; j < numLeds; j++) {

digitalWrite(ledPins[j], LOW);

}

// Включаем текущий светодиод

digitalWrite(ledPins[i], HIGH);

// Включаем следующий светодиод, если он существует

if (i + 1 < numLeds) {

digitalWrite(ledPins[i + 1], HIGH);

}

// Ждем перед следующим шагом

delay(delayTime);

}

}

Ссылка на проект: https://www.tinkercad.com/things/aAgQVyXY14r-bilet-13?sharecode=khFQm0GW8xWD\_qI0stIgb7mjarpEmxDc1TkdQxO-BFk

**Билет 14**

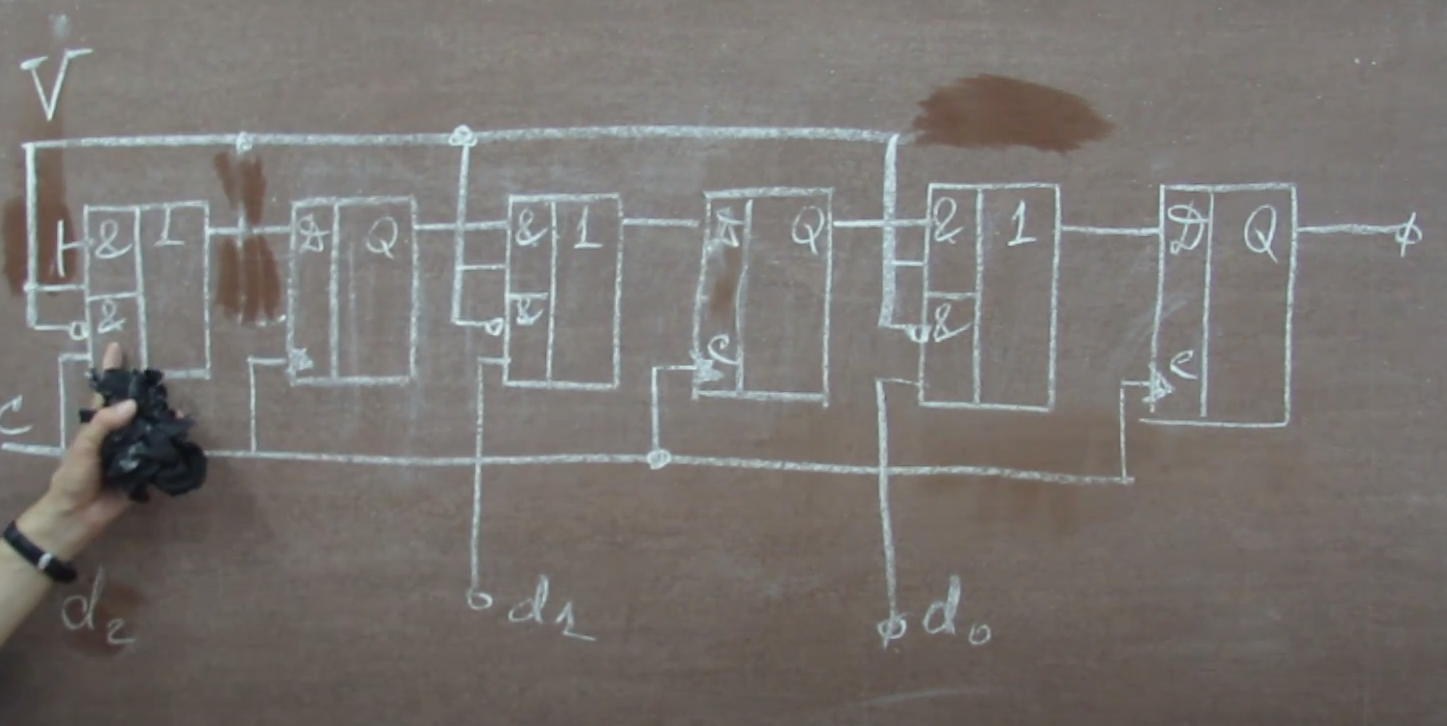
**2.Параллельно - последовательный регистр. Графическое обозначение.**

Параллельно-последовательный регистр (иногда также называемый сдвиговый регистр) представляет собой устройство цифровой электроники, которое может передавать данные из всех своих ячеек параллельно или последовательно.

### Принцип работы:

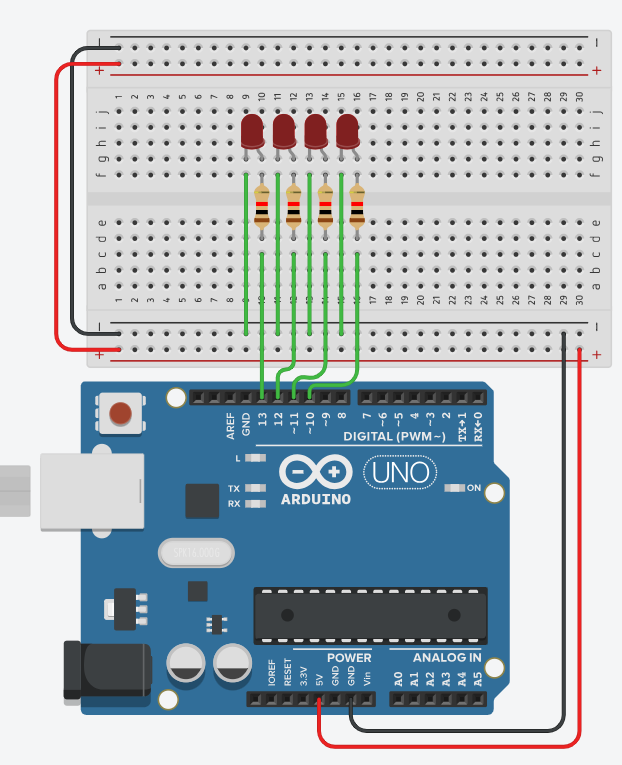
* В параллельном режиме все ячейки (D0-D7) могут одновременно принимать или передавать данные, что позволяет быстро загружать или выгружать данные в регистр.
* В последовательном режиме данные передаются последовательно из одной ячейки в другую, что позволяет сдвигать данные на одну позицию (вправо или влево) за каждый такт синхронизации (обычно сигнал CLK).

Ссылка на сайт : https://www.electronicsblog.ru/cifrovaya-sxemotexnika/parallelnye-i-posledovatelnye-registry.html



Параллельно-последовательные регистры широко используются для сдвига данных в цифровых схемах, таких как сдвиговые регистры (shift registers), которые могут использоваться для последовательного чтения или записи данных во временном хранилище.

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, четырех светодиодов. Написать программный код «Накапливающееся включение светодиодов: 1, 2, 3, 4 включенных светодиодов» для работы схемы.**



// Определяем пины, к которым подключены светодиоды

const int ledPins[] = {13, 12, 11, 10};

const int numLEDs = 4;

void setup() {

// Устанавливаем пины светодиодов как выходы

for (int i = 0; i < numLEDs; i++) {

pinMode(ledPins[i], OUTPUT);

}

}

void loop() {

// Поочередно включаем светодиоды

for (int i = 0; i < numLEDs; i++) {

digitalWrite(ledPins[i], HIGH);

delay(200); // Ждем 200 миллисекунд перед включением следующего светодиода

}

// Поочередно выключаем светодиоды

for (int i = 0; i < numLEDs; i++) {

digitalWrite(ledPins[i], LOW);

delay(200); // Ждем 200 миллисекунд перед выключением следующего светодиода

}

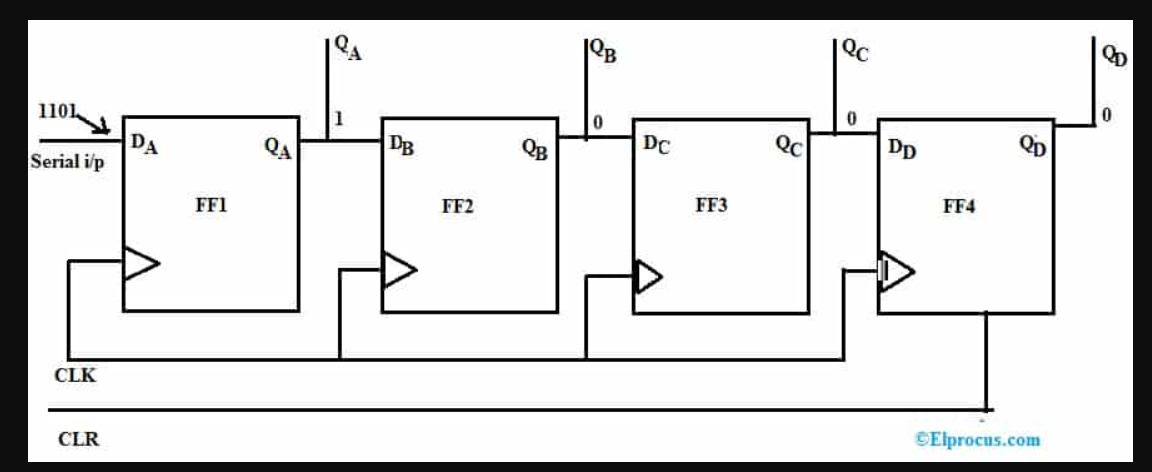
}

Ссылка на проект: https://www.tinkercad.com/things/dsfqtd8bVWM-bilet-14?sharecode=5h7ScOlqFeW--Pc2iOVG97FV2NSC6e4LD2\_G-nIHPBQ

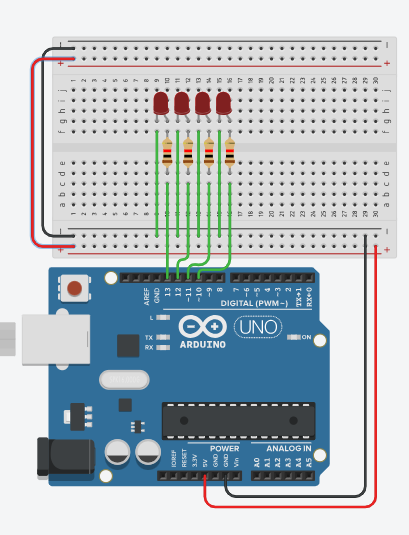
**Билет 15**

**2.Последовательно - параллельный регистр. Графическое обозначение.**

**Последовательно-параллельный регистр (SIPO, Serial-In, Parallel-Out)** принимает данные последовательно (по одному биту за такт) и выводит их параллельно (все биты одновременно). Это устройство часто используется для преобразования последовательного потока данных в параллельный формат.



**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, четырех светодиодов. Написать программный код «Выключение светодиодов: 4, 3, 2, 1 включенных светодиодов» для работы схемы.**



// Определяем пины, к которым подключены светодиоды

const int ledPins[] = {13, 12, 11, 10};

const int numLEDs = 4;

void setup() {

// Устанавливаем пины светодиодов как выходы

for (int i = 0; i < numLEDs; i++) {

pinMode(ledPins[i], OUTPUT);

}

// Инициализация: включаем все светодиоды

for (int i = 0; i < numLEDs; i++) {

digitalWrite(ledPins[i], HIGH);

}

}

void loop() {

// Поочередно выключаем светодиоды, начиная с четвертого

for (int i = numLEDs - 1; i >= 0; i--) {

digitalWrite(ledPins[i], LOW);

delay(200); // Ждем 200 миллисекунд перед выключением следующего светодиода

}

// Поочередно включаем светодиоды, начиная с первого

for (int i = 0; i < numLEDs; i++) {

digitalWrite(ledPins[i], HIGH);

delay(200); // Ждем 200 миллисекунд перед включением следующего светодиода

}

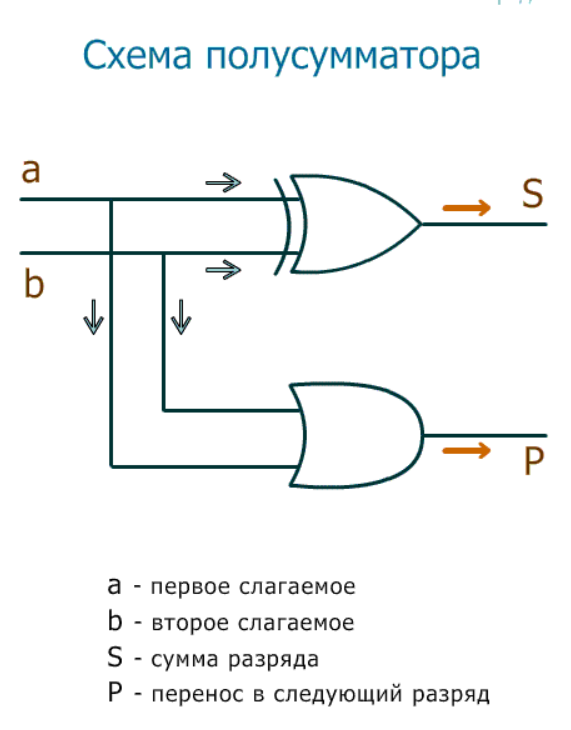
}

Ссылка на проект: https://www.tinkercad.com/things/3ziIP4AaDbx-bilet-15?sharecode=iss9NMl3leBWPlPpnuZe1FKHDeyf9iKXBojtc7qNetM

**Билет № 16**

**2.Полусумматор. Графическое обозначение, таблица истинности**

Полусумматор — это логический элемент, который выполняет операции сложения двух однобитовых чисел без учета переноса (carry). Он имеет два входа: A и B, и два выхода: сумма (Sum) и перенос (Carry).





### Принцип работы:

* **Sum (Сумма)** вычисляется как результат логической операции XOR между входами A и B. Он показывает бит результата сложения.
* **Carry (Перенос)** вычисляется как результат логической операции AND между входами A и B. Этот выход показывает, есть ли перенос при сложении двух бит.

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одного трех цветного (RGB) светодиода. Написать программный код «Плавное переключение цветов» для работы схемы.**



// Определяем пины, к которым подключены RGB светодиод

const int redPin = 9;

const int greenPin = 10;

const int bluePin = 11;

void setup() {

// Устанавливаем пины RGB светодиода как выходы

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

}

void loop() {

// Плавное изменение цвета от красного к зеленому

for (int i = 0; i < 256; i++) {

analogWrite(redPin, 255 - i);

analogWrite(greenPin, i);

analogWrite(bluePin, 0);

delay(10);

}

// Плавное изменение цвета от зеленого к синему

for (int i = 0; i < 256; i++) {

analogWrite(redPin, 0);

analogWrite(greenPin, 255 - i);

analogWrite(bluePin, i);

delay(10);

}

// Плавное изменение цвета от синего к красному

for (int i = 0; i < 256; i++) {

analogWrite(redPin, i);

analogWrite(greenPin, 0);

analogWrite(bluePin, 255 - i);

delay(10);

}

}

Ссылка на проект: https://www.tinkercad.com/things/74uIuiHHdkC-bilet-16?sharecode=qxJ2YrxYohhxU1O1a-Uc7\_-sPK4iPNJOv7KGj86pajM

**Билет № 17**

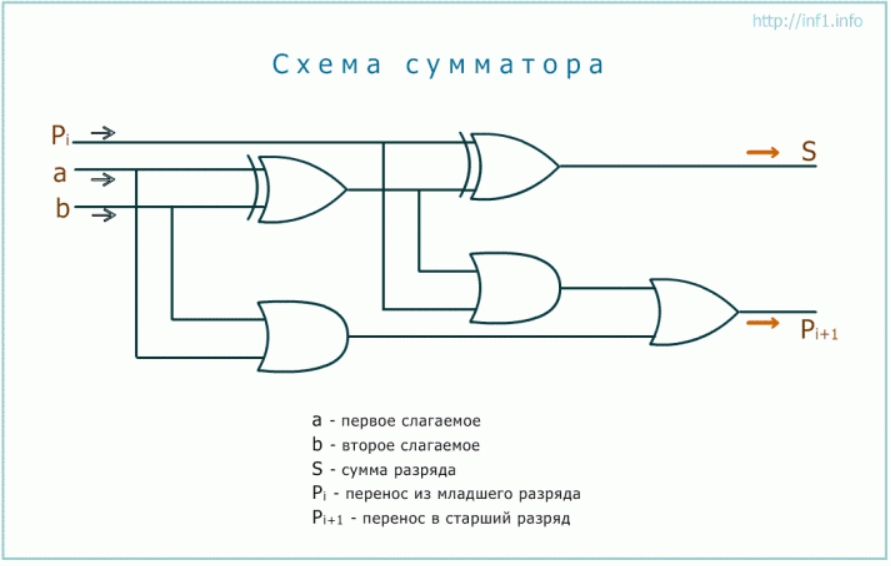
**2.Сумматор. Графическое обозначение, таблица истинности.**

### **Сумматор**

Сумматор — это логическое устройство, которое выполняет операцию сложения двух двоичных чисел с учетом возможного переноса (carry). Обычно сумматор может складывать два однобитовых числа, а также учитывать перенос от предыдущего бита.

### Принцип работы:

* **Sum (Сумма)** вычисляется как результат логической операции XOR (исключающее ИЛИ) между A, B и Cin. Он показывает бит результата сложения текущих разрядов.
* **Cout (Перенос)** вычисляется как результат логической операции OR (ИЛИ) между двумя или тремя входами (A AND B, A AND Cin, B AND Cin). Этот выход показывает, есть ли перенос на следующий разряд.



**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одного трех цветного (RGB) светодиода. Написать программный код «Переключение цветов: красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый» для работы схемы.**

// Определяем пины, к которым подключены RGB светодиод

const int redPin = 9;

const int greenPin = 10;

const int bluePin = 11;

// Определяем массив цветов в формате RGB

int colors[][3] = {

{255, 0, 0}, // Красный

{255, 165, 0}, // Оранжевый

{255, 255, 0}, // Желтый

{0, 255, 0}, // Зеленый

{0, 255, 255}, // Голубой

{0, 0, 255}, // Синий

{128, 0, 128} // Фиолетовый

};

const int numColors = 7;

void setup() {

// Устанавливаем пины RGB светодиода как выходы

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

}

void loop() {

// Переключаемся между цветами

for (int i = 0; i < numColors; i++) {

analogWrite(redPin, colors[i][0]);

analogWrite(greenPin, colors[i][1]);

analogWrite(bluePin, colors[i][2]);

delay(1000); // Ждем 1 секунду перед переключением на следующий цвет

}

}

Ссылка на проект: https://www.tinkercad.com/things/12oozksPf6z-bilet-17?sharecode=TIFsDQI5tBVpDKjTyZT82LPfjUa1hjRvmDWXTzM0TeY

**Билет № 18**

**2.Арифметико логическое устройство, сколько и каких выполняет операций?**

Арифметико-логическое устройство (АЛУ) представляет собой ключевой элемент центрального процессора (ЦП) компьютера, который выполняет арифметические (сложение, вычитание, умножение, деление) и логические операции (И, ИЛИ, НЕ, XOR) над двоичными данными.

### Количество АЛУ в ЦП:

1. **Одно АЛУ в одном ядре ЦП**: В одиночном ядре ЦП обычно присутствует одно арифметико-логическое устройство, которое выполняет все основные арифметические и логические операции.
2. **Несколько АЛУ в многопроцессорных системах**: В многопроцессорных или многоядерных системах может быть несколько АЛУ, каждое из которых может выполнять операции параллельно. Это позволяет увеличить общую производительность системы, распределяя нагрузку между несколькими ядрами или процессорами.

**Сколько АЛУ в микропроцессоре:**

* В **современных микропроцессорах** АЛУ не является **отдельным блоком**, а **интегрировано в ядро процессора**.
* Количество АЛУ в ядре процессора может **варьироваться** в зависимости от его **архитектуры**.
  + **Простые процессоры** могут иметь **одно** АЛУ.
  + **Многоядерные процессоры** могут иметь **по одному АЛУ на каждое ядро**.
  + **Специализированные процессоры**, такие как **графические процессоры (GPU)**, могут иметь **множество АЛУ**, оптимизированных для выполнения **специфических задач**.

### Операции, выполняемые АЛУ:

1. **Арифметические операции**:
   * **Сложение**: Суммирует два числа.
   * **Вычитание**: Вычитает одно число из другого.
   * **Умножение**: Перемножает два числа.
   * **Деление**: Делит одно число на другое, обычно выполняется с использованием деления в степенях двойки (сдвиги и вычитания).
2. **Логические операции**:
   * **И (AND)**: Выполняет логическое И для каждого бита двух чисел.
   * **ИЛИ (OR)**: Выполняет логическое ИЛИ для каждого бита двух чисел.
   * **Исключающее ИЛИ (XOR)**: Выполняет логическое XOR для каждого бита двух чисел.
   * **НЕ (NOT)**: Инвертирует все биты числа.

**Работа АЛУ:**

АЛУ получает операнды и управляющие сигналы от управляющего устройства процессора (контроллера), выполняет требуемую операцию в соответствии с командами, закодированными в машинных инструкциях, и выдает результат операции в регистры или на шины данных процессора.

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, трех резисторов переменного сопротивления, одного трех цветного (RGB) светодиода. Написать программный код «При вращении ручек резисторов переменного сопротивления, изменяются яркости свечения светодиодов» для работы схемы.**



// Определяем пины, к которым подключены RGB светодиод

const int redPin = 9;

const int greenPin = 10;

const int bluePin = 11;

// Определяем пины, к которым подключены резисторы переменного сопротивления

const int redPotPin = A0;

const int greenPotPin = A1;

const int bluePotPin = A2;

void setup() {

// Устанавливаем пины RGB светодиода как выходы

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

// Устанавливаем пины резисторов переменного сопротивления как входы

pinMode(redPotPin, INPUT);

pinMode(greenPotPin, INPUT);

pinMode(bluePotPin, INPUT);

}

void loop() {

// Считываем значения с резисторов переменного сопротивления

int redValue = analogRead(redPotPin);

int greenValue = analogRead(greenPotPin);

int blueValue = analogRead(bluePotPin);

// Преобразуем значения с резисторов переменного сопротивления в диапазон 0-255

redValue = map(redValue, 0, 1023, 0, 255);

greenValue = map(greenValue, 0, 1023, 0, 255);

blueValue = map(blueValue, 0, 1023, 0, 255);

// Устанавливаем яркость RGB светодиода

analogWrite(redPin, redValue);

analogWrite(greenPin, greenValue);

analogWrite(bluePin, blueValue);

// Задержка для стабилизации считывания

delay(10);

}

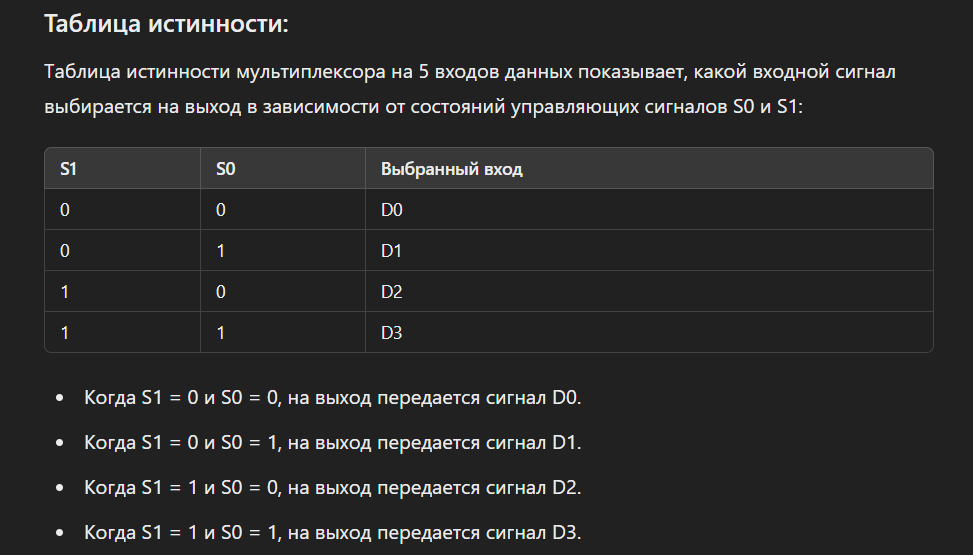
Ссылка на проект : https://www.tinkercad.com/things/8NhhCFCEau1-bilet-18?sharecode=udQfqJatjhAwpA4uEVmLiM6NeDXTmvHtr7-RqWRgRRM

**Билет № 19**

**2.Мультиплексор на 5 входов данных. Графическое обозначение, таблица истинности.**

Мультиплексор (MUX) — это устройство цифровой электроники, которое выбирает один из нескольких входных сигналов и передает его на выход в зависимости от управляющих сигналов. Мультиплексор на 5 входов данных имеет 5 входов данных (D0-D4) и несколько управляющих входов для выбора нужного входа.

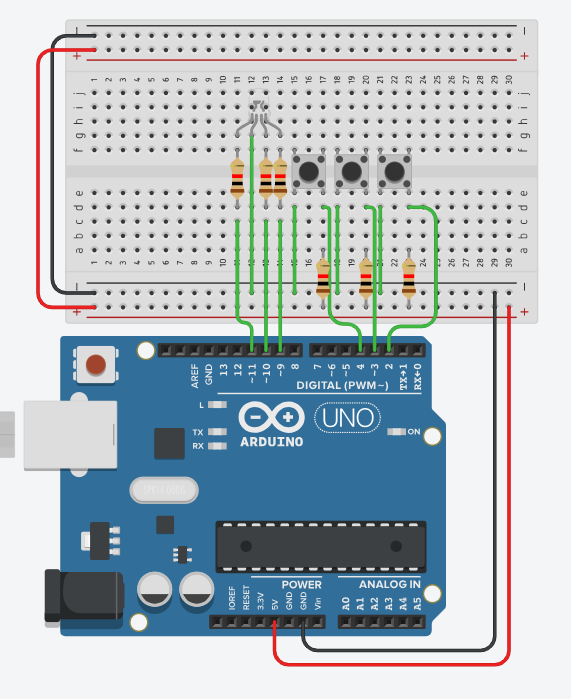




### Применение:

Мультиплексоры широко используются в цифровых системах для выбора и коммутации данных с нескольких источников на один выход, что делает их важным компонентом в построении мультиплексированных схем и переключателей.

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, трех тактовых кнопок, одного трех цветного (RGB) светодиода. Написать программный код «При нажатии на одну из трех кнопок, светодиод включается одним из трех цветов» для работы схемы**



// Определение пинов

const int buttonRed = 2; // Пин для кнопки красного цвета

const int buttonGreen = 3; // Пин для кнопки зеленого цвета

const int buttonBlue = 4; // Пин для кнопки синего цвета

const int pinRed = 9; // Пин для управления красным цветом RGB-светодиода

const int pinGreen = 10; // Пин для управления зеленым цветом RGB-светодиода

const int pinBlue = 11; // Пин для управления синим цветом RGB-светодиода

void setup() {

// Настройка пинов кнопок как входов

pinMode(buttonRed, INPUT\_PULLUP);

pinMode(buttonGreen, INPUT\_PULLUP);

pinMode(buttonBlue, INPUT\_PULLUP);

// Настройка пинов RGB-светодиода как выходов

pinMode(pinRed, OUTPUT);

pinMode(pinGreen, OUTPUT);

pinMode(pinBlue, OUTPUT);

// Инициализация RGB-светодиода в выключенное состояние

digitalWrite(pinRed, LOW);

digitalWrite(pinGreen, LOW);

digitalWrite(pinBlue, LOW);

}

void loop() {

// Чтение состояния кнопок

bool redButtonState = digitalRead(buttonRed) == LOW;

bool greenButtonState = digitalRead(buttonGreen) == LOW;

bool blueButtonState = digitalRead(buttonBlue) == LOW;

// Обновление состояния RGB-светодиода в зависимости от нажатия кнопок

if (redButtonState) {

digitalWrite(pinRed, HIGH);

digitalWrite(pinGreen, LOW);

digitalWrite(pinBlue, LOW);

} else if (greenButtonState) {

digitalWrite(pinRed, LOW);

digitalWrite(pinGreen, HIGH);

digitalWrite(pinBlue, LOW);

} else if (blueButtonState) {

digitalWrite(pinRed, LOW);

digitalWrite(pinGreen, LOW);

digitalWrite(pinBlue, HIGH);

} else {

// Если ни одна кнопка не нажата, выключаем все цвета

digitalWrite(pinRed, LOW);

digitalWrite(pinGreen, LOW);

digitalWrite(pinBlue, LOW);

}

}

Ссылка на проект: https://www.tinkercad.com/things/0SmJXX53fBM-bilet-19?sharecode=mRytGrxLuo8nN2KToV9WBkBdzPVE-E\_jlUeejIFbS9A

**Билет № 20**

**2.Последовательный периферийный интерфейс – SPI.**

SPI (Serial Peripheral Interface) — это последовательный периферийный интерфейс, используемый для обмена данными между микроконтроллерами (или другими цифровыми устройствами) и периферийными устройствами. Он представляет собой один из наиболее распространенных и гибких способов коммуникации между различными устройствами в цифровых системах.

### Основные характеристики SPI:

1. **Проводка и сигналы**:
   * SPI использует четыре основных линии:
     + **SCLK (Serial Clock)** — серийный тактовый сигнал, который определяет скорость передачи данных между устройствами.
     + **MOSI (Master Out Slave In)** — линия передачи данных от мастера (обычно микроконтроллера) к слейву (периферийному устройству).
     + **MISO (Master In Slave Out)** — линия передачи данных от слейва к мастеру.
     + **SS/CS (Slave Select/Chip Select)** — линия, которая активизирует конкретное периферийное устройство для обмена данными (может быть несколько линий SS для выбора нескольких устройств).
2. **Режимы передачи данных**:
   * SPI поддерживает различные режимы передачи данных, определяющие фазы тактового сигнала (SCLK) и положение данных:
     + **CPOL (Clock Polarity)** и **CPHA (Clock Phase)** определяют фазы SCLK, при которых данные передаются (например, данные могут читаться на фронте или на спаде SCLK).
3. **Одновременность передачи и приема**:
   * SPI поддерживает полудуплексную передачу данных, что позволяет микроконтроллеру и периферийному устройству одновременно передавать и принимать данные.
4. **Преимущества SPI**:
   * Высокая скорость передачи данных.
   * Простота реализации и использования.
   * Гибкость конфигурации (различные режимы и конфигурации линий).
   * Поддержка множества устройств на одной шине (при наличии достаточного числа линий SS/CS).

### Пример применения:

SPI широко используется для связи с различными периферийными устройствами, такими как дисплеи, датчики, память EEPROM, адаптеры Wi-Fi и многие другие. Он обеспечивает надежную и быструю передачу данных, что делает его предпочтительным интерфейсом для многих приложений в электронике и встроенных системах.

### Дополнительные сведения:

* <https://en.wikipedia.org/wiki/Serial_Peripheral_Interface>
* <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/introduction>
* <https://www.analog.com/en/resources/app-notes/an-1248.html>

**3.В онлайн-программе Tinkercad — собрать схему с использованием Arduino – UNO, одного трех цветного (RGB) светодиода. Написать программный код «Плавное переключение цветов» для работы схемы.**

**16 билет**

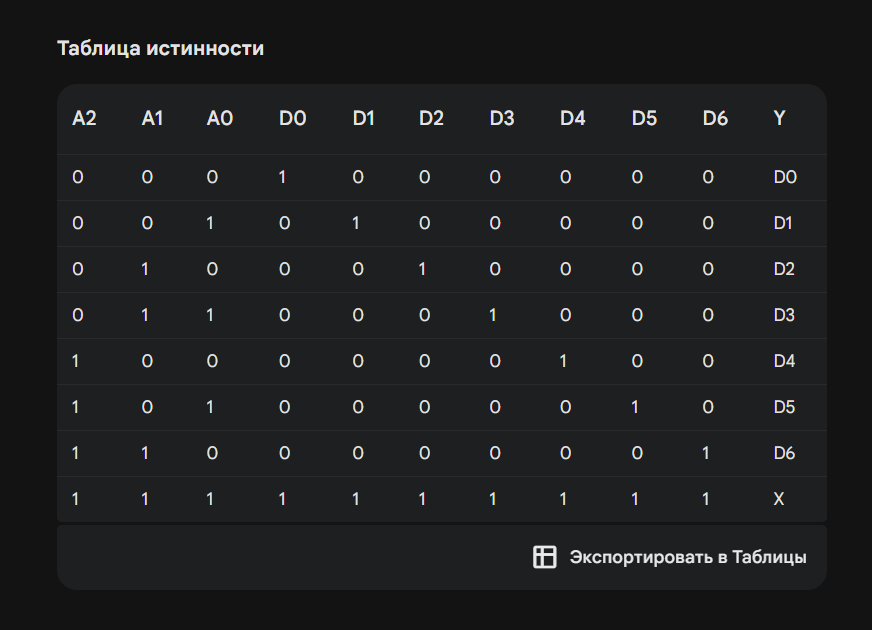
**Билет № 21**

**2.Мультиплексор на 7 входов данных. Графическое обозначение, таблица истинности.**

## **Мультиплексор на 7 входов данных**

**Мультиплексор на 7 входов данных** (7 к 1) - это цифровая схема, которая **выбирает один из семи входных сигналов** и **передает его на свой выход**. Выбор входного сигнала осуществляется **с помощью двоичного кода**, подаваемого на **адресные входы**.





**В таблице:**

* **A2, A1, A0** - двоичный код, подаваемый на адресные входы.
* **D0 - D6** - значения на информационных входах.
* **Y** - значение на выходе.
* **X** - значение на выходе не определено (зависит от реализации мультиплексора).

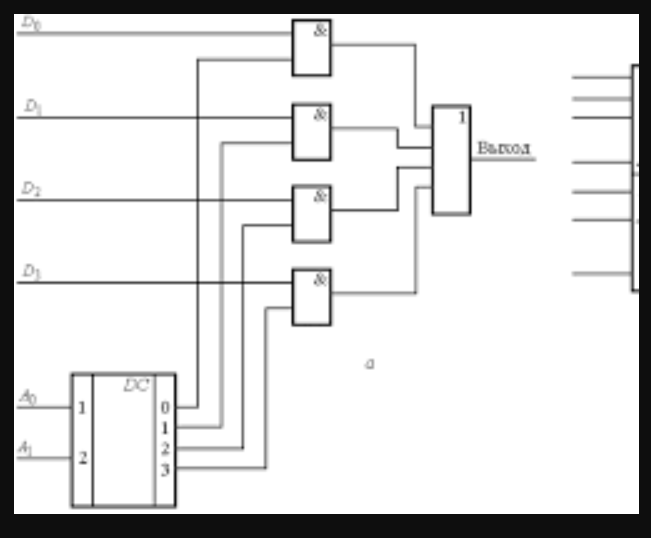
**Как работает мультиплексор:**

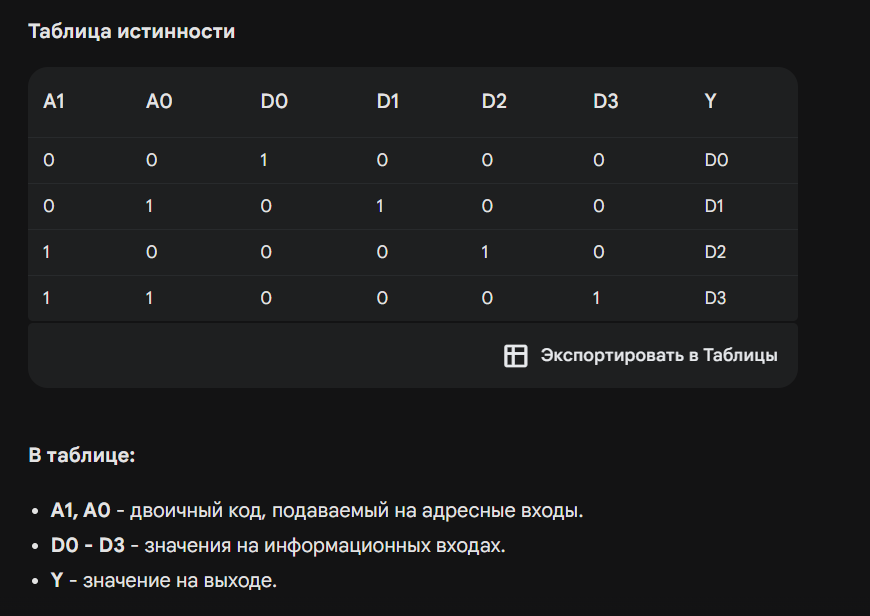
1. **Двоичный код** на **адресных входах** (A2, A1, A0) **определяет**, какой **информационный вход** (D0 - D6) будет **выбран**.
2. **Значение** выбранного **информационного входа** **передается на выход** (Y).
3. **Значения** на **невыбранных информационных входах** **игнорируются**.

**Билет № 22**

**2.Мультиплексор на 4 входа данных. Графическое обозначение, таблица истинности.**

**Мультиплексор на 4 входа данных** (4 к 1) - это цифровая схема, которая **выбирает один из четырех входных сигналов** и **передает его на свой выход**. Выбор входного сигнала осуществляется **с помощью двоичного кода**, подаваемого на **адресные входы**.





**Как работает мультиплексор:**

1. **Двоичный код** на **адресных входах** (A1, A0) **определяет**, какой **информационный вход** (D0 - D3) будет **выбран**.
2. **Значение** выбранного **информационного входа** **передается на выход** (Y).
3. **Значения** на **невыбранных информационных входах** **игнорируются**.

**Билет № 23**

**2.Универсальный синхронный последовательный приемопередатчик.**

Универсальный асинхронный последовательный приемопередатчик (УАПП), известный также как UART (Universal Asynchronous Receiver/Transmitter), является ключевым элементом для обеспечения асинхронной последовательной связи между микроконтроллерами (или другими устройствами) и внешними устройствами.

### Основные характеристики UART:

1. **Асинхронная передача данных**:
   * UART поддерживает передачу данных без общего тактового сигнала (например, без использования внешней синхронизации), что делает его удобным для использования в различных приложениях.
2. **Поддержка полудуплексной передачи**:
   * В режиме полудуплексной передачи UART позволяет одновременно передавать и принимать данные, но не одновременно.
3. **Скорость передачи данных (бод)**:
   * UART поддерживает настройку скорости передачи данных (частоту бод), что позволяет адаптировать передачу данных в зависимости от требований конкретного приложения.
4. **Формат кадра данных**:
   * Включает количество бит данных (обычно от 5 до 9), количество стоповых бит и возможность использования бита контроля четности для обеспечения целостности данных.
5. **Протоколы и режимы работы**:
   * Поддерживает различные протоколы и режимы работы, включая стандартные и расширенные режимы, которые могут варьироваться в зависимости от конкретной реализации.

### **Принцип работы UART:**

UART состоит из двух основных функциональных блоков — приемника (Receiver) и передатчика (Transmitter):

* **Приемник (Receiver)**: Принимает последовательные данные, полученные от внешнего устройства, и конвертирует их в параллельный формат для дальнейшей обработки микроконтроллером.
* **Передатчик (Transmitter)**: Преобразует параллельные данные, полученные от микроконтроллера, в последовательный формат для передачи на внешнее устройство.

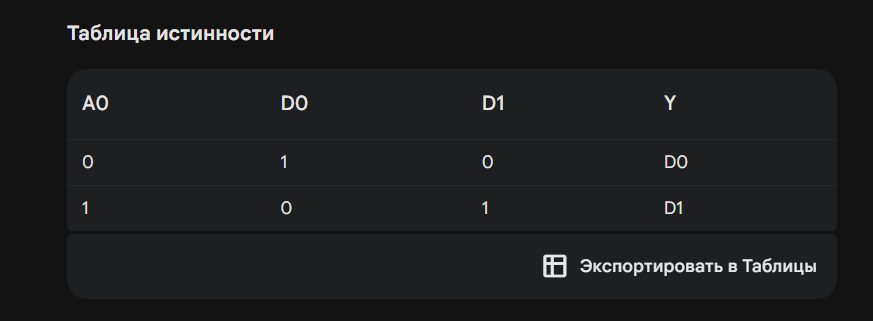
### **Применение UART:**

UART широко используется во встроенных системах для коммуникации с различными внешними устройствами, такими как модемы, GPS-приемники, Bluetooth-модули, датчики, RFID-считыватели и другие устройства. Он обеспечивает надежную и простую в реализации асинхронную связь между микроконтроллером и внешними устройствами, что делает его неотъемлемой частью в электронике и встроенных системах.

**Билет № 24**

**2.Мультиплексор на 2 входа данных. Графическое обозначение, таблица истинности.**

**Мультиплексор на 2 входа данных** (2 к 1) - это цифровая схема, которая **выбирает один из двух входных сигналов** и **передает его на свой выход**. Выбор входного сигнала осуществляется **с помощью двоичного кода**, подаваемого на **вход выбора**.



**Как работает мультиплексор:**

* **Значение на входе выбора (A0)** **определяет**, какой **информационный вход** (D0 или D1) будет **выбран**.
* **Значение** выбранного **информационного входа** **передается на выход** (Y).
* **Значение** на **невыбранном информационном входе** **игнорируется**.

**Билет № 25**

**Микропроцессорная система (МПС)** - это **компьютерная система**, построенная на основе **микропроцессора**. МПС включает в себя:

* **Центральный процессор (ЦП)** - микропроцессор, который является "мозгом" системы и выполняет все вычисления.
* **Оперативная память (ОЗУ)** - память, в которой хранятся программы и данные, необходимые для работы ЦП.
* **Постоянная память (ПЗУ)** - память, в которой хранятся программы и данные, которые не должны меняться.
* **Устройства ввода-вывода (УВВ)** - устройства, которые позволяют пользователю взаимодействовать с системой.
* **Системная шина** - набор проводников, по которым передаются данные между различными компонентами системы.
* **Системные часы** - источник тактовых импульсов, которые синхронизируют работу всех компонентов системы.

### Описание типовой МПС

**ЦП** является основным компонентом МПС. Он выполняет все вычисления и логические операции, необходимые для работы системы. ЦП состоит из следующих основных блоков:

* **Арифметико-логическое устройство (АЛУ)** - выполняет арифметические и логические операции.
* **Устройство управления (УУ)** - управляет работой АЛУ и других блоков ЦП.
* **Регистры** - хранят временные данные и команды.

**ОЗУ** используется для хранения программ и данных, которые ЦП использует в данный момент. ОЗУ - это **быстрая память**, к которой можно быстро получить доступ.

**ПЗУ** используется для хранения программ и данных, которые не должны меняться. ПЗУ - это **медленная память**, чем ОЗУ, но к ней можно получить доступ только для чтения.

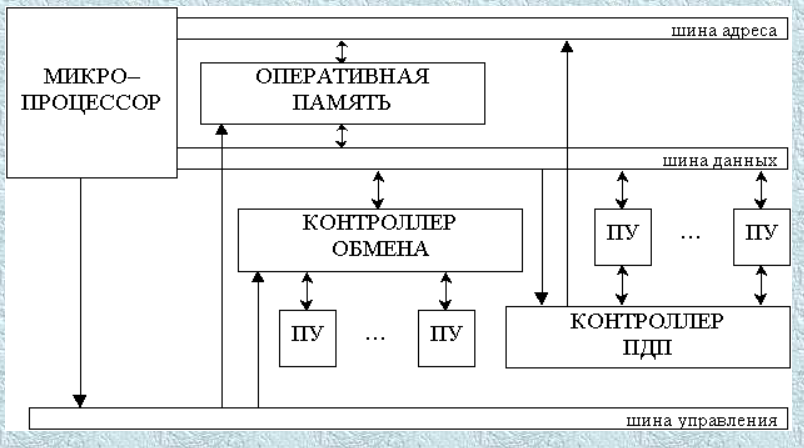
**УВВ** позволяют пользователю взаимодействовать с системой. К УВВ относятся:

* **Клавиатура** - используется для ввода текстовых данных.
* **Монитор** - используется для вывода информации.
* **Мышь** - используется для управления курсором на экране.
* **Принтер** - используется для вывода информации на бумагу.

**Системная шина** - это набор проводников, по которым передаются данные между различными компонентами системы. Системная шина состоит из трех основных частей:

* **Шина адреса** - используется для передачи адресов памяти или устройств ввода-вывода.
* **Шина данных** - используется для передачи данных между ЦП, ОЗУ, ПЗУ и УВВ.
* **Шина управления** - используется для передачи управляющих сигналов между различными компонентами системы.

**Системные часы** - это источник тактовых импульсов, которые синхронизируют работу всех компонентов системы. Тактовые импульсы определяют частоту, с которой ЦП выполняет инструкции



Ссылка на сайт : https://conspect.narod.ru/c\_4.htm

**Билет № 26**

**2.Универсальный асинхронный последовательный приемопередатчик — УАПП – UART.**

## Универсальный асинхронный приемопередатчик (УАПП) - UART

**Универсальный асинхронный приемопередатчик (УАПП)**, также известный как **UART**, - это **физический узел** в вычислительных устройствах, предназначенный для **организации связи с другими цифровыми устройствами**.

**UART** обеспечивает **последовательную передачу данных** по **двум проводам**, используя **асинхронный протокол**.

**Асинхронный протокол** означает, что **данные передаются без использования тактового сигнала**, а **синхронизируются с помощью стартового и стопового битов**.

[learn.adacore.com/pdf\_books/learning-ada.pdf](https://learn.adacore.com/pdf_books/learning-ada.pdf)

## Универсальный асинхронный приемопередатчик (УАПП) - UART

**Универсальный асинхронный приемопередатчик (УАПП)**, также известный как **UART**, - это **физический узел** в вычислительных устройствах, предназначенный для **организации связи с другими цифровыми устройствами**.

**UART** обеспечивает **последовательную передачу данных** по **двум проводам**, используя **асинхронный протокол**.

**Асинхронный протокол** означает, что **данные передаются без использования тактового сигнала**, а **синхронизируются с помощью стартового и стопового битов**.

### Характеристики UART:

* **Двухпроводной интерфейс:** UART использует два провода: один для передачи данных (TX) и один для приема данных (RX).
* **Асинхронный протокол:** Данные передаются без использования тактового сигнала.
* **Настраиваемая скорость передачи данных:** Скорость передачи данных может быть настроена в широком диапазоне, обычно от 50 бод до 230 400 бод.
* **Контроль четности:** UART может добавлять бит четности к передаваемым данным для обнаружения ошибок.
* **Обнаружение ошибок:** UART может обнаруживать ошибки в принимаемых данных.
* **Простота реализации:** UART имеет простую аппаратную реализацию, что делает его экономичным и надежным.

### **Применение UART:**

* **Связь с микроконтроллерами:** UART часто используется для связи между микроконтроллерами и другими устройствами.
* **Отладка:** UART может использоваться для отладки программного обеспечения микроконтроллера.
* **Интерфейсы пользователя:** UART может использоваться для реализации интерфейсов пользователя, таких как текстовые консоли.
* **Сетевые коммуникации:** UART может использоваться для реализации простых сетевых протоколов.
* **Промышленные приложения:** UART используется в различных промышленных приложениях, таких как системы управления и автоматизированные системы.

**Билет № 27**

## Двухпроводной последовательный интерфейс TWI (Two-Wire Interface)

**TWI** (**Two-Wire Interface**) - это **двухпроводной последовательный интерфейс**, совместимый со стандартами **I2C** и **SMBus**.

Он широко используется для **подключения периферийных устройств** к микроконтроллерам.

**TWI** использует **мастер-подчиненную архитектуру**, где **один мастер** может управлять **до 127 подчиненных устройств**.

### Характеристики TWI:

* **Двухпроводной интерфейс:** TWI использует два провода: один для тактового сигнала (SCL) и один для линии данных (SDA).
* **Последовательная передача данных:** Данные передаются по одному биту за раз.
* **Мастер-подчиненная архитектура:** Один мастер может управлять до 127 подчиненных устройств.
* **Адресное пространство:** Каждому подчиненному устройству присваивается уникальный 7-битный адрес.
* **Обнаружение коллизий:** TWI использует арбитраж шины для обнаружения и разрешения коллизий.
* **Поддержка многомастерного режима:** Некоторые реализации TWI поддерживают многомастерный режим, который позволяет нескольким мастерам делить шину.

### Применение TWI:

* **Подключение датчиков:** TWI часто используется для подключения датчиков, таких как датчики температуры, акселерометры и датчики давления.
* **Подключение преобразователей аналого-цифрового и цифро-аналогового сигналов (АЦП/ЦАП):** TWI используется для подключения АЦП/ЦАП, которые преобразуют аналоговые сигналы в цифровые и наоборот.
* **Подключение устройств памяти:** TWI используется для подключения устройств памяти, таких как EEPROM и флэш-память.
* **Подключение дисплеев:** TWI используется для подключения дисплеев, таких как LCD-дисплеи и OLED-дисплеи.
* **И многое другое:** TWI может использоваться для подключения широкого спектра других периферийных устройств.

**Билет № 28**

Демультиплексор (англ. Demultiplexer, сокр. Demux) — это логическое устройство, которое принимает один входной сигнал данных и направляет его на один из множества выходных каналов в зависимости от значения управляющих сигналов.

### Графическое обозначение демультиплексора:

Демультиплексор на 4 выхода данных имеет один входной сигнал данных (D), два управляющих входа для выбора выходного канала (S0 и S1), и четыре выхода данных (Y0, Y1, Y2, Y3).



### Объяснение таблицы истинности:

* **S1 и S0** — управляющие входы, которые выбирают один из четырех выходных каналов в зависимости от их комбинации.
* **D** — входной сигнал данных, который будет направлен на выбранный выходной канал.
* **Y0, Y1, Y2, Y3** — выходные каналы, к которым будет направлен входной сигнал D в зависимости от управляющих сигналов S1 и S0.

Например, если S1 = 0, S0 = 1, то входной сигнал D будет направлен на выход Y1, а все остальные выходы Y0, Y2 и Y3 будут равны нулю.

**Билет № 29**

**2.Последовательная шина - I2C.**

**I2C (Inter-Integrated Circuit) — это двухпроводный последовательный интерфейс, разработанный фирмой Philips (ныне NXP Semiconductors), который предоставляет способ для взаимодействия между различными интегральными схемами в электронных устройствах. I2C использует две линии для передачи данных — SDA (Serial Data Line) и SCL (Serial Clock Line).**

### Основные характеристики и принцип работы I2C:

1. **Двухпроводная шина**:
   * **SDA (Serial Data Line)**: Передает данные между устройствами.
   * **SCL (Serial Clock Line)**: Предоставляет тактовый сигнал, по которому синхронизируется передача данных.
2. **Мастер-слейв архитектура**:
   * Один устройство действует как мастер (Master), который инициирует передачу данных и генерирует тактовый сигнал SCL.
   * Остальные устройства работают как слейвы (Slave), которые отвечают на запросы мастера и передают данные по запросу.
3. **Адресация устройств**:
   * Каждое устройство на шине I2C имеет уникальный 7-битный адрес, который используется для идентификации устройства во время передачи данных.
   * В некоторых случаях поддерживается 10-битная адресация для расширенной адресации устройств.
4. **Формат кадра данных**:
   * Данные передаются последовательно, в формате байта данных, с возможностью опционального использования бита подтверждения приема (ACK/NACK).
5. **Режимы работы**:
   * I2C поддерживает два режима работы: стандартный режим (Standard Mode, до 100 кбит/с) и быстрый режим (Fast Mode, до 400 кбит/с). Некоторые устройства поддерживают еще более высокие скорости передачи (Fast Mode Plus, High-Speed Mode).

### Применение I2C:

I2C широко используется для коммуникации между различными компонентами в электронных устройствах, включая датчики, EEPROM-память, LCD-дисплеи, акселерометры, гироскопы, управляющие устройства, аудио-кодеки и многое другое. Это делает его особенно полезным во встроенных системах, где требуется эффективное взаимодействие между различными устройствами при минимальном использовании проводов и пинов микроконтроллера.

**Билет № 30**

**2.Опишите транзисторно-транзисторная логику — ТТЛ - TTL**

**Транзисторно-транзисторная логика (ТТЛ, TTL) — это стандарт логических схем, основанный на использовании биполярных транзисторов (обычно биполярных транзисторов с изолированным затвором, BJT) для реализации логических функций. TTL был одним из первых и наиболее распространенных стандартов логики в электронике, использовавшимся с 1960-х по начало 2000-х годов.**

### Основные характеристики TTL:

1. **Базовая схема:**
   * Основной элемент TTL — это два транзистора, соединенных между источником питания и землей (напряжением питания Vcc и землей GND). Обычно используются транзисторы с открытым коллектором или открытым стоком.
2. **Логические уровни:**
   * TTL работает с двумя логическими уровнями:
     + **Высокий уровень (High)**: Обычно от 2.4V до Vcc (обычно 5V), что соответствует логической "1".
     + **Низкий уровень (Low)**: Обычно от 0V до 0.8V, что соответствует логической "0".
3. **Стандартные TTL-схемы:**
   * TTL включает различные типы логических элементов, такие как инверторы, И-ИЛИ-НЕ элементы, триггеры и счетчики, которые сочетаются для создания сложных логических схем.
4. **Скорость работы:**
   * Изначально TTL обеспечивал высокую скорость работы по сравнению с другими стандартами логики того времени. Скорость TTL может достигать нескольких мегагерц в зависимости от конкретной реализации.
5. **Потребление энергии:**
   * TTL характеризуется относительно высоким потреблением энергии по сравнению с более современными стандартами, такими как CMOS (Complementary Metal-Oxide Semiconductor).

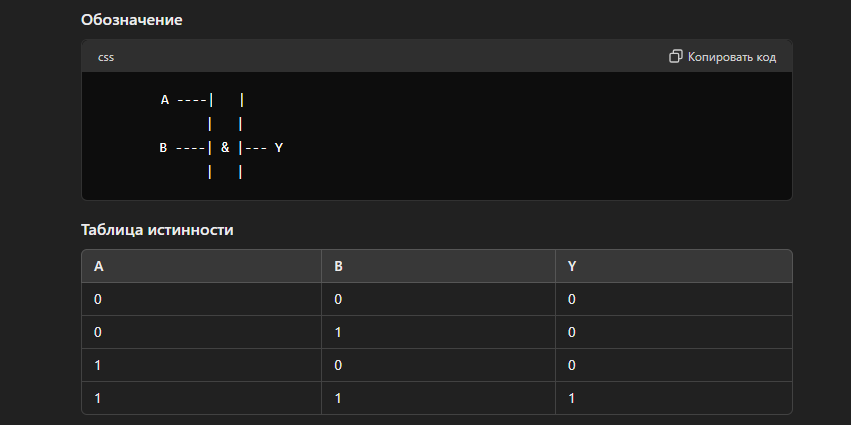
### Применение TTL:

* TTL был широко использован в цифровой электронике для построения вычислительных устройств, компьютеров, управляющих систем и других электронных устройств.
* С развитием технологий, TTL был постепенно замещен более современными стандартами, такими как CMOS, благодаря их более низкому потреблению энергии и лучшей совместимости с микроконтроллерами и интегральными схемами.

**Заметка для себя**

### 1. Логический элемент И (AND)

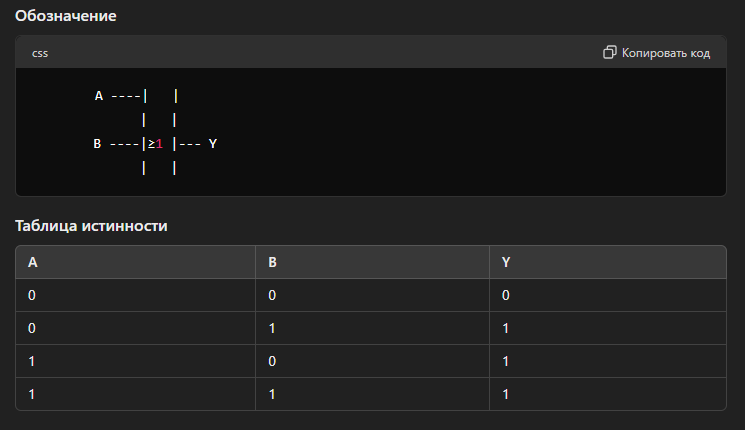
**Логический элемент И** выполняет операцию логического умножения. Выход элемента И равен 1 только тогда, когда все его входы равны 1.



Свет включается (Y), если оба выключателя (A и B) включены (оба равны 1).

### 2. Логический элемент ИЛИ (OR)

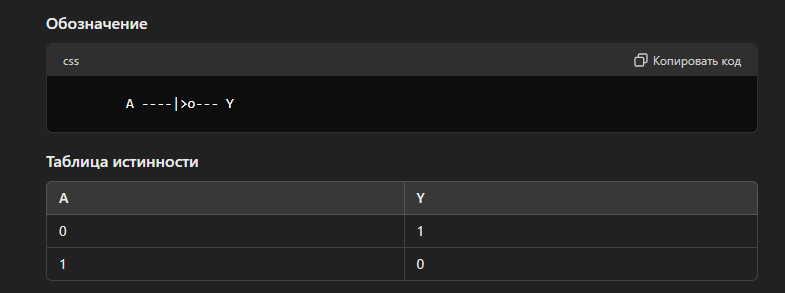
**Логический элемент ИЛИ** выполняет операцию логического сложения. Выход элемента ИЛИ равен 1, если хотя бы один из его входов равен 1.



Свет включается (Y), если хотя бы один из выключателей (A или B) включен.

### 3. Логический элемент НЕ (NOT)

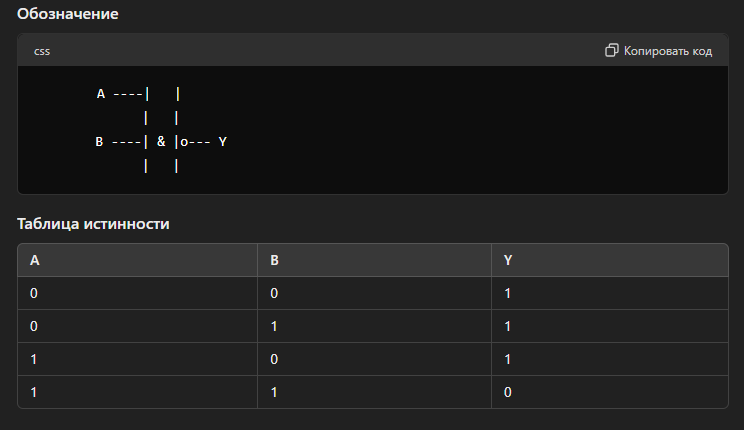
**Логический элемент НЕ** инвертирует входной сигнал. Выход элемента НЕ равен 1, если вход равен 0, и наоборот.



Лампочка горит (Y), когда выключатель (A) выключен.

### 4. Логический элемент И-НЕ (NAND)

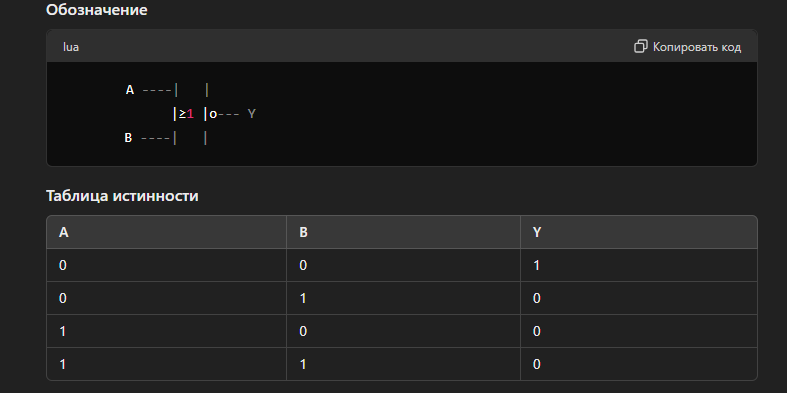
**Логический элемент И-НЕ** выполняет операцию логического умножения с последующим инвертированием. Выход элемента И-НЕ равен 0 только тогда, когда все его входы равны 1.



Сигнал тревоги (Y) срабатывает, когда хотя бы одно из условий безопасности (A или B) не выполнено.

### 5. Логический элемент ИЛИ-НЕ (NOR)

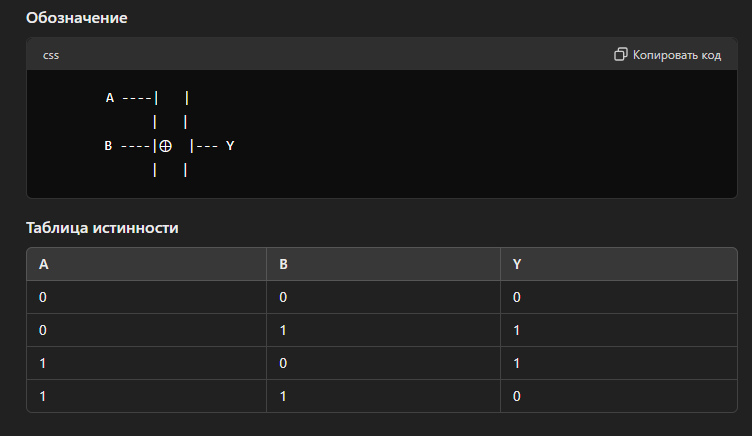
**Логический элемент ИЛИ-НЕ** выполняет операцию логического сложения с последующим инвертированием. Выход элемента ИЛИ-НЕ равен 1, если все его входы равны 0.



Сигнал тревоги (Y) срабатывает, если оба датчика (A и B) отключены.

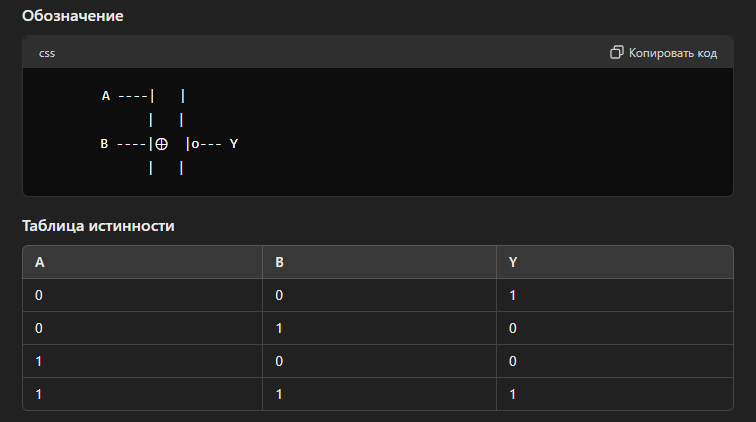
### 6. Логический элемент Исключающее ИЛИ (XOR)

**Логический элемент Исключающее ИЛИ** выполняет операцию сложения по модулю 2. Выход элемента XOR равен 1, если его входы различны.

  
Лампочка (Y) горит, если выключатели (A и B) находятся в разных состояниях.

### 7. Логический элемент Исключающее ИЛИ-НЕ (XNOR)

**Логический элемент Исключающее ИЛИ-НЕ** выполняет операцию исключающего ИЛИ с последующим инвертированием. Выход элемента XNOR равен 1, если его входы одинаковы.



#### Пример использования

Лампочка (Y) горит, если оба выключателя (A и B) находятся в одном состоянии.

Простыми словами

**AND**: Все должны быть включены, чтобы получить выход 1.

**OR**: Достаточно одного включенного, чтобы получить выход 1.

**NOT**: Инвертирует вход.

**NAND**: Противоположность AND.

**NOR**: Противоположность OR.

**XOR**: Включен, если входы разные.

**XNOR**: Включен, если входы одинаковы.

Дешифратор — показывает цифру на входе и в соответствии с цифрой подает сигнал на выход как на цифре

Шифратор — переводит номер входа в бинарную цифру на выходе

Мультиплексор — имеет адресную шину и какое- то количество входов (2 в n степени где n – это количество адресных входов) и один выход. Число из адресной шины определяет номер входа которая поступит на выход

Демультиплексор — входную информацию выводит в выход соответтсвующий числу в адресной шине