Глава 2. SPL, GPIO и немного хардвара

Ежелев Г. И.

30 января 2026 г.

В предыдущей главе мы пользовались библиотекой CMSIS - Cortex Microcontroller Software Interface Standard. Подобные библиотеки есть для всех МК, в том числе и ардуино. Везде один и тот же принцип - настройка всего через регистры. Более того, любая программа для любого ПК в итоге сводиться к схожему набору действий. Однако для STM32 есть еще 2 библиотеки для работы с периферией: SPL - Standard Peripheral Library и HAL - Hardware Abstraction Layer. HAL самая высокоуровневая, SPL чуть ближе к CMSIS. SPL и HAL под капотом используют CMSIS.

SPL по факту сопоставляет каждому значению каждого параметра свой набор битов в регистре, причем SPL будет подстраиваться под выбранный МК (у разных МК могут по разному работать регистры с одинаковым названием и назначением), что ускорит переход с одной STM на другую.

Устройство SPL

Для каждого устройства внутри STM32 в библиотеке SPL есть своя структура (structure). Загуглите, что такое structure в C++.

Она содержит значение каждого параметра устройства (Весь набор значений одного параметра часто тоже является структурой).

Когда мы настраиваем какое-то устройство, сначала создаем экземпляр структуры, далее заполняем поля структуры и далее передаем заполненную структуру в функцию инициализации структуры.

Разберем на примере GPIO


```
typedef struct
{
  uint32_t GPIO_Pin;              /*!< Specifies the GPIO pins to be configured.
                                       This parameter can be any value of @ref GPIO_pins_define */

  GPIOMode_TypeDef GPIO_Mode;     /*!< Specifies the operating mode for the selected pins.
                                       This parameter can be a value of @ref GPIOMode_TypeDef */

  GPIOSpeed_TypeDef GPIO_Speed;   /*!< Specifies the speed for the selected pins.
                                       This parameter can be a value of @ref GPIOSpeed_TypeDef */

  GPIOOType_TypeDef GPIO_OType;   /*!< Specifies the operating output type for the selected pins.
                                       This parameter can be a value of @ref GPIOOType_TypeDef */

  GPIOPuPd_TypeDef GPIO_PuPd;     /*!< Specifies the operating Pull-up/Pull down for the selected pins.
                                       This parameter can be a value of @ref GPIOPuPd_TypeDef */
}GPIO_InitTypeDef;
```

Эту структуру можно найти в файле stm32f4xx_gpio.h в разделе spl4.

Слева параметры, справа их описание. После слова @ref (reference) идет текст, который есть перед структурой, содержащей все возможные значения этого параметра, при поиске текста после @ref через ctrl+f в этом файле вы найдете все допустимые значения параметра. Просто исследуя этот файл можно полностью понять принцип работы любого устройства и его настроить.

Пример настройки GPIO

```
1   GPIO_InitTypeDef _initParams;
2
3   _initParams.GPIO_Pin = GPIO_Pin_1;
4
5   _initParams.GPIO_Mode = GPIO_Mode_OUT;
6   _initParams.GPIO_Speed = GPIO_Speed_100MHz;
7   _initParams.GPIO_OType = GPIO_OType_PP;
8   _initParams.GPIO_PuPd = GPIO_PuPd_DOWN;
9
10  GPIO_Init(GPIOA, &_initParams);
```

Попробуйте самостоятельно изучить за что отвечает каждый параметр. Можно гуглить.
Отдельно отметим строчку 1 - создание структуры, строчку 10 - инициализация пина GPIO по заполненной структуре и строчку 7 выбор режима подтяжки.
Пин может быть подтянут к земле(вниз) или к питанию (вверх), за это отвечает параметр PuPd. В обоих случаях подтяжка происходит либо через резистор - PP, либо напрямую OD (Open Drain). Второй случай не безопасен - возможно короткое замыкание, такое подключение используется редко, если вам нужна просто подтяжка всегда выбирайте PP.

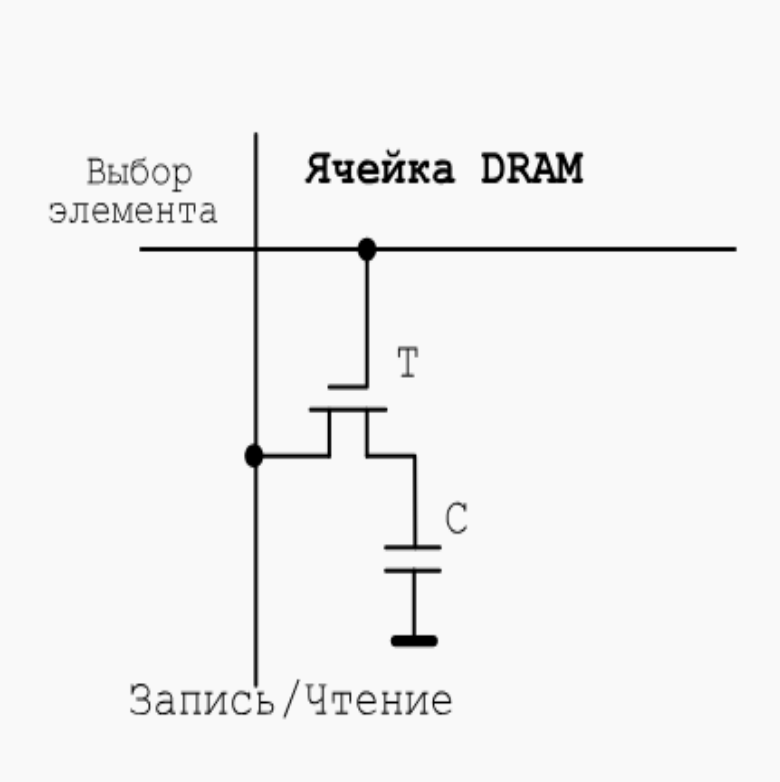
Read & Write

GPIO_SetBits(GPIOA, GPIO_Pin_1); или
GPIO_SetBits(GPIOA, _initParams.GPIO_Pin); - выставление единицы
GPIO_ResetBits(GPIOA, GPIO_Pin_1); - выставление 0

GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) - чтение пина PA1
GPIO_ReadInputData(GPIOA) - чтение всего порта GPIOA, эквивалентно чтению всего регистра IDR

Остальные функции так же можно найти в файле stm32f4xx_gpio.h раздел spl4.
Очевидно SPL удобнее, чем CMSIS, поэтому далее мы будем пользоваться преимущественно SPL (за исключением главы 3).

Память. DRAM



Ячейка DRAM - Dynamic Random Access Memory состоит из емкости и транзистора.

Конструкция довольно простая => память дешевая

Главный минус - постепенная утечка заряда с емкости.

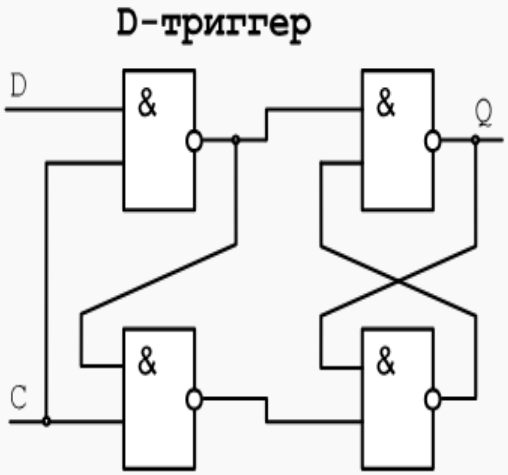
Решает эту проблему устройство регенерации памяти, которое раз в $\approx$ 64мс читает значение с каждой ячейки и перезаписывает его, для обновления заряда. Так устроена ОЗУ.

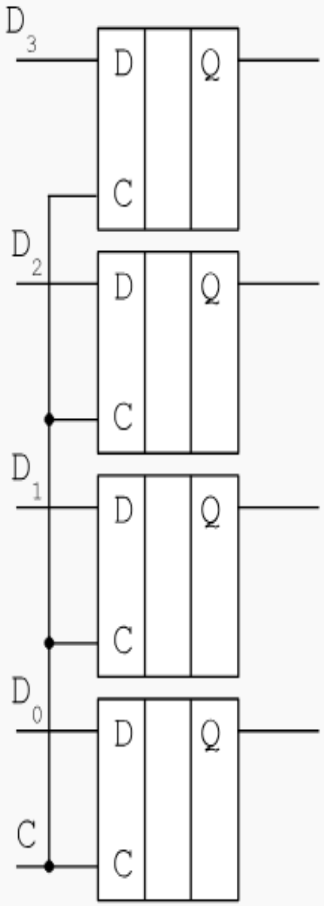
Защелки. D-триггеры. Регистры

В самом ядре так же находятся свои маленькие ячейки памяти для хранения промежуточных результатов и управления устройствами - регистры. Они состоят из защелок (триггеров), обычно это D-триггеры.
D-триггер имеет 2 входа - Data и Clock. Data выставляет значение, Clock определяет, когда значение с входа D будет записано в триггер. Выход отвечает за хранимое значение.
Регистр - объединение D-триггеров с общим выводом Clock. Это самый быстрый (по скорости доступа) тип памяти в любой ЭВМ.

Функциональные элементы ЭВМ

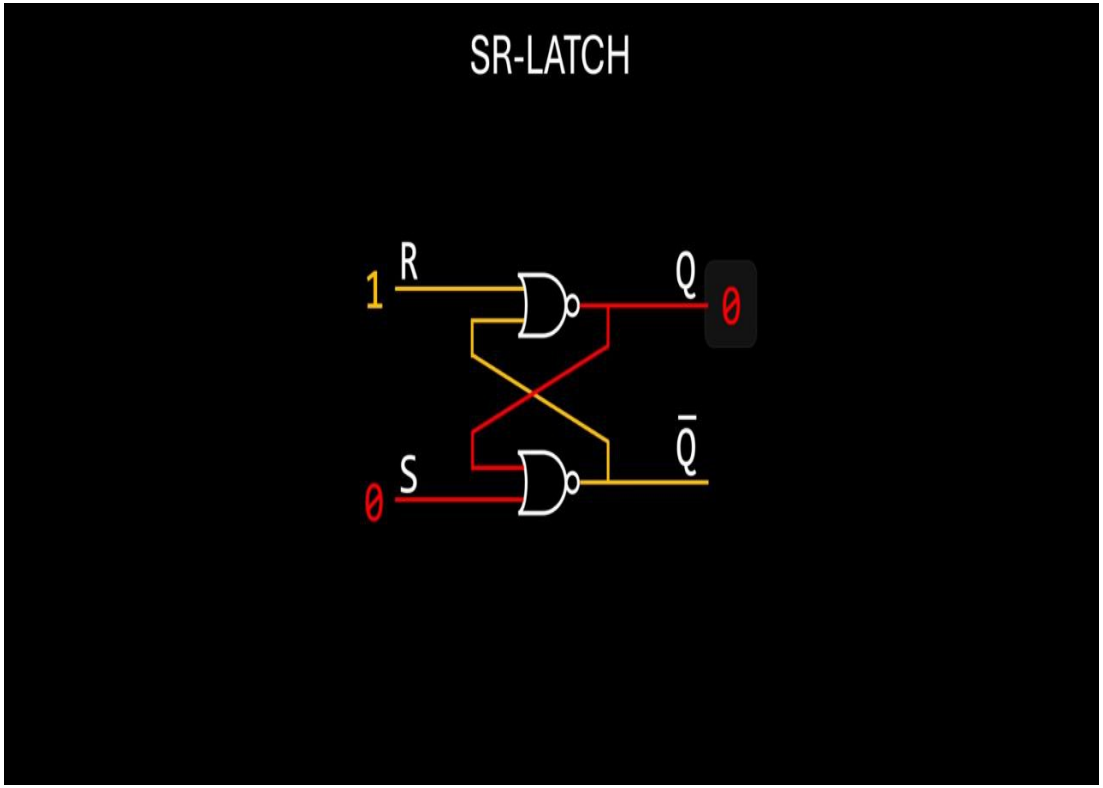
- Элементы хранения (триггеры, регистры)





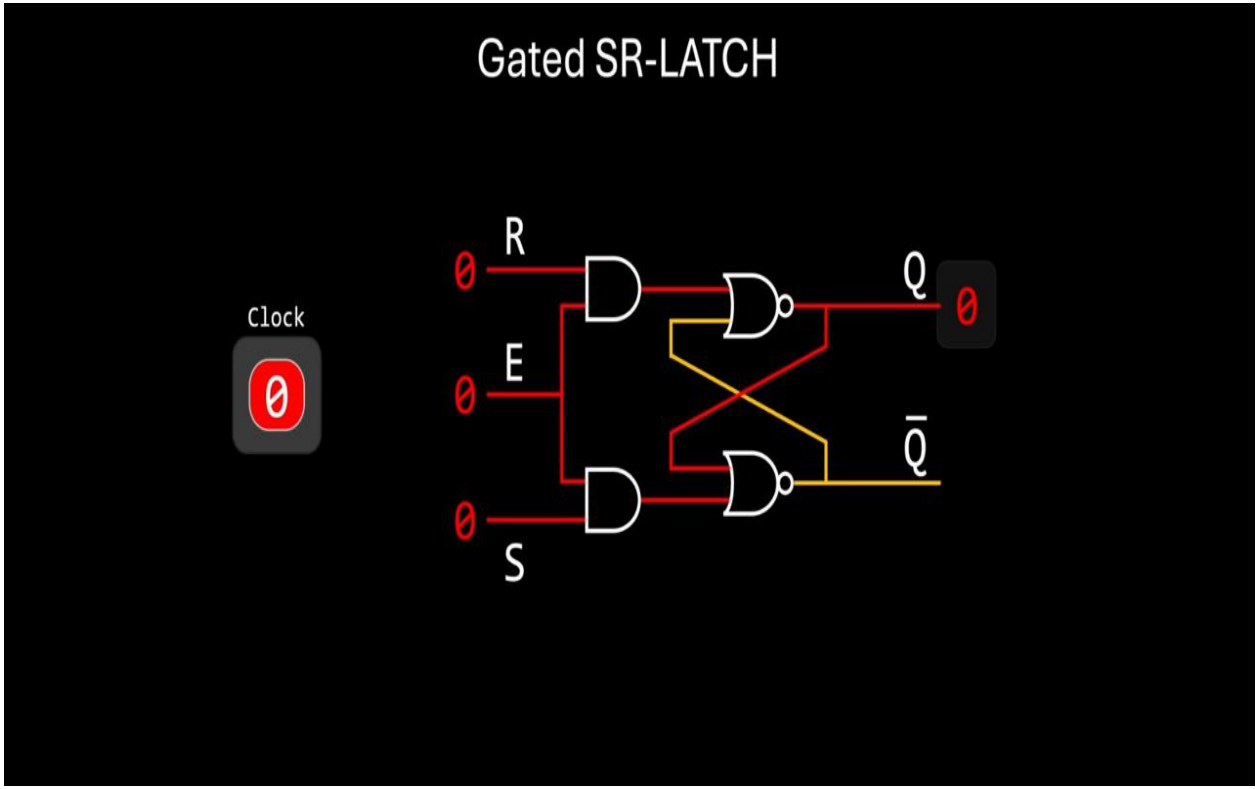
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0	0	0	0	0	1	0	1	1

Защелки. SR-триггеры.



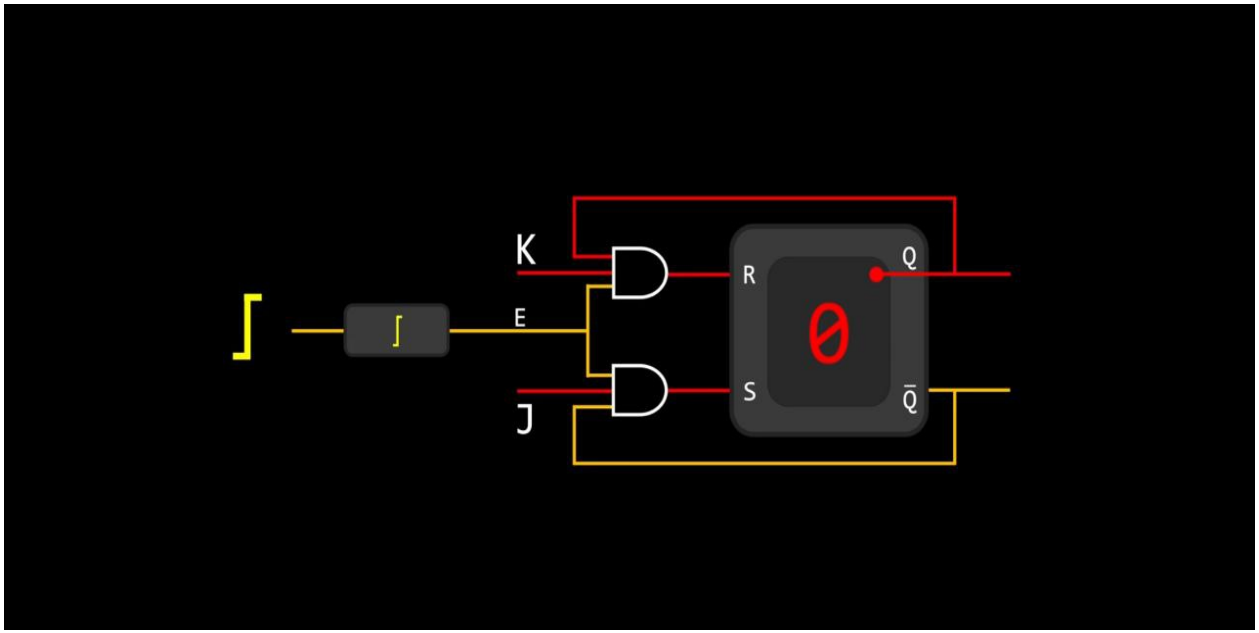
Такая защелка называется SR-триггер - Set-Reset триггер. То есть отдельный канал для записи единицы и для записи 0. попробуйте самостоятельно проследить, как происходит переключение сигналов. Что будет, если подать на обе шины 0? 1?

Защелки. SR-триггеры.



Дополним его сигналом Enable и подключим его к тактовому генератору (Clock). Таким образом мы исключили доступ к триггеру вне периода импульса тактового генератора. Обозначим полученную схему отдельным элементом.

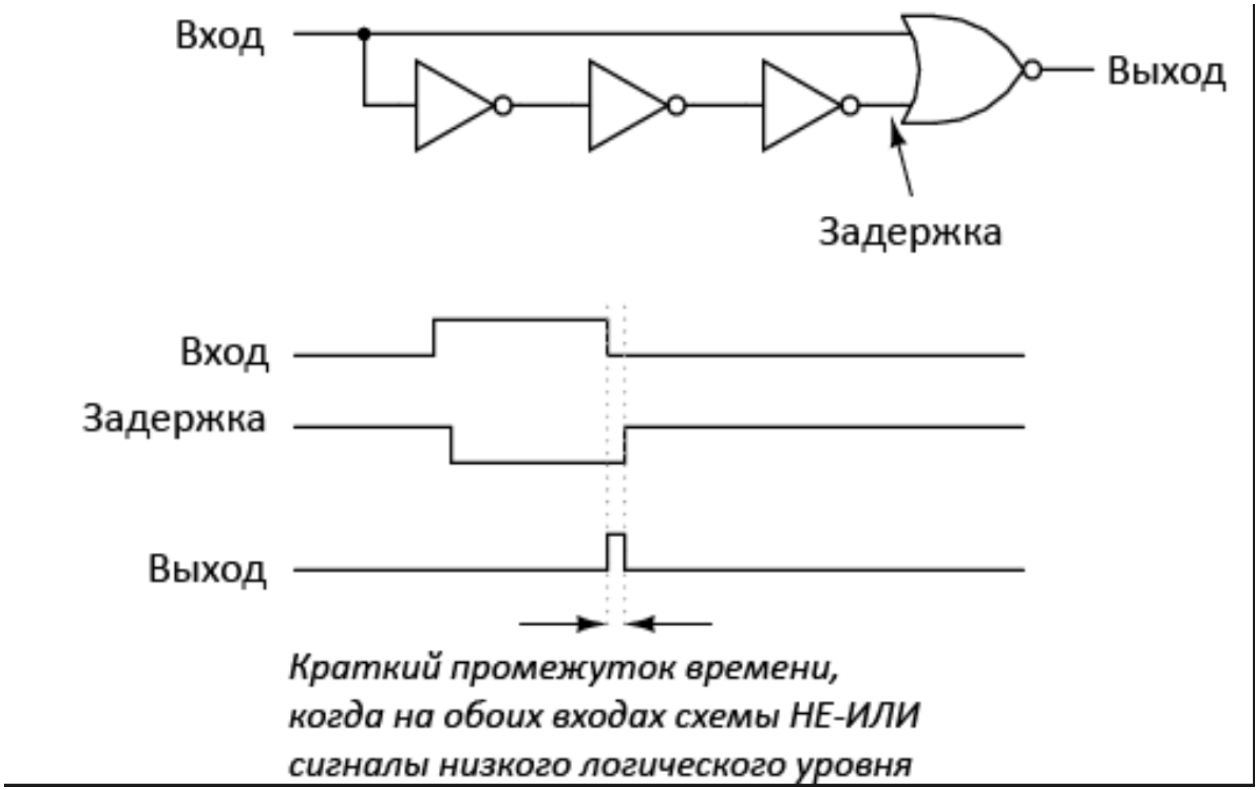
Защелки. Flip-flop. Делитель частоты



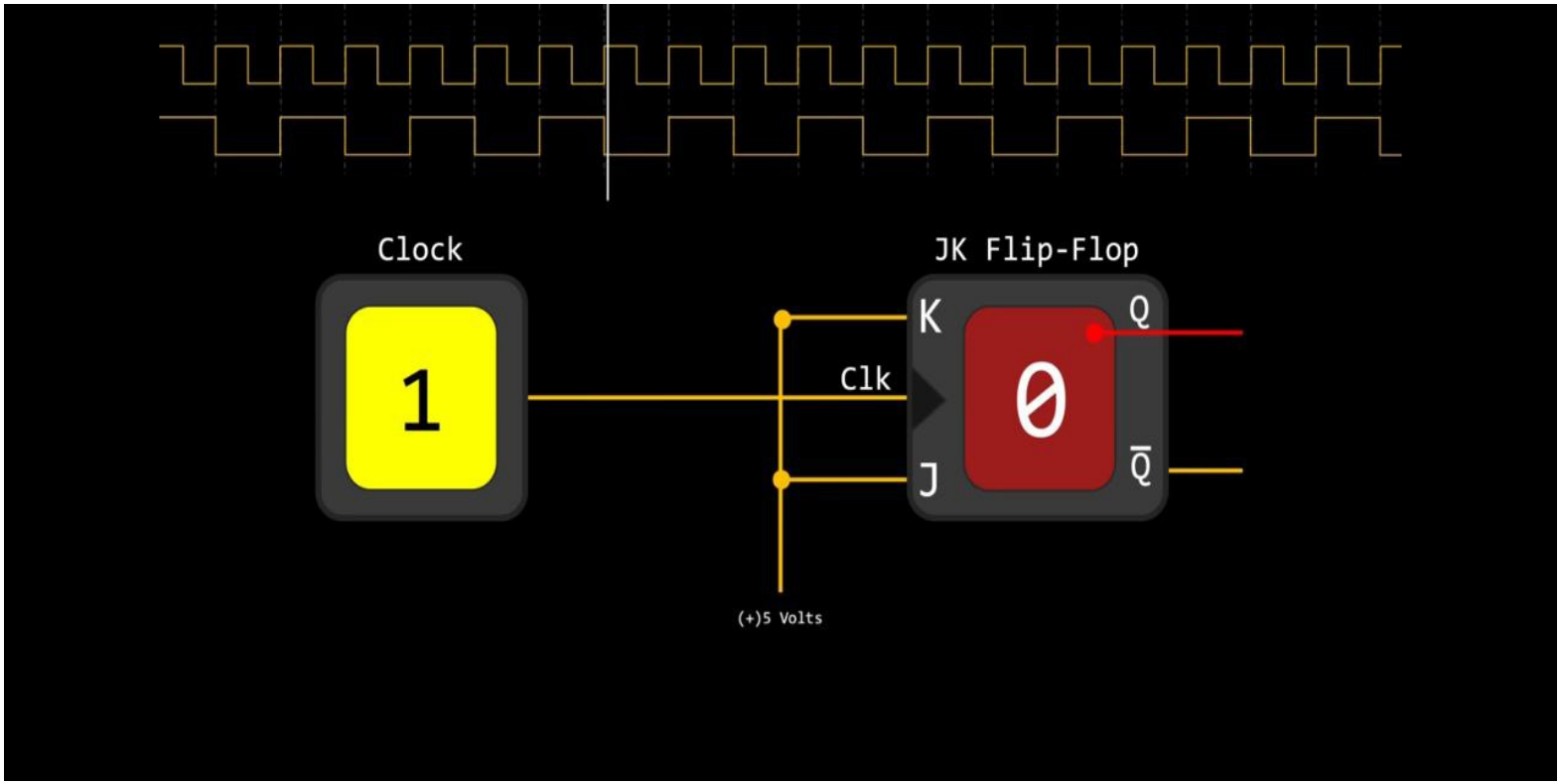
Подключим через логическое И соответствующие выходы триггера к входам. Enable подключим к тактовому генератору через детектор переднего фронта. Обозначим полученную схему - Flip-Flop - как один элемент. Входы обозначим буквами J и K. Отметим, что теперь мы можем только переключать значение, выставить одновременно 1 на оба входа не получиться. Если J и K оба единицы, то значение будет переключаться ровно 1 раз при восходящем фронте тактового генератора

Edge detector. Делитель частоты

А как работает детектор переднего фронта?



Делитель частоты



Соеденим выходы Set и Reset и подключим их всегда на логическую 1. Теперь при каждом восходящем фронте тактового генератора значение схемы будет меняться, а при спаде сигнала значение меняться не будет.

Получили деление частоты ровно в 2 раза. Эта схема часто используется в устройстве RCC

В этой презентации использованы фрагменты видео с очень интересного канала Core Dumped https://youtube.com/@coredumpped?si=aH26LLnzdDBBSxPN

Также использованы слайды курса на кафедре вычислительной техники ИТМО, их можно найти в папке docs в репозитории

Материалы курса можно найти по ссылке:
https://github.com/haaroner/ezh239