Лабораторная работа №1	группа 05	2022
Построение логических схем в среде моделирования	Москаленко Т.Д.	

Цель работы: моделирование логических схем на элементах с памятью.

Инструментарий и требования к работе: работа выполняется в среде моделирования Logisim evolution. Ссылка на скачивание

Счётчик

Вариант Асинхронный суммирующий счетчик. Модуль 17

Описание В работе представлен асинхронный суммирующий счётчик. Счётчик — это схема, которая подсчитывает количество входных импульсов. Суммирующий означает, что каждый импульс прибавляет к результату 1. Асинхронный значит, что вычисление происходит не одновременно, а последовательно от разряда к разряду.

Для работы нам потребовались RS, D и 'reset-T' триггеры, счётчик по модулю 32 и модуль, который проверяет то, что входное число равно 17.

RS_trigger

Ниже представлена таблица истинности для RS триггера

R	S	Q
0	0	сохраняем значение
0	1	1
1	0	0
1	1	запрещаем такой вход

Таблица 1: Таблица истинности RS триггера.

Ниже представлена реализация

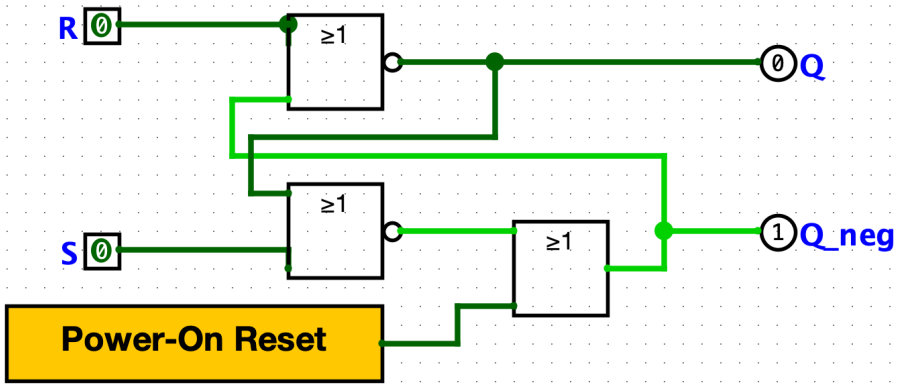

Рис. 1: Реализация RS триггера

D_trigger

Ниже представлена таблица истинности для D триггера

D	C	Q
0	0	сохраняем значение
0	1	0
1	0	сохраняем значение
1	1	1

Таблица 2: Таблица истинности D триггера.

Ниже представлена реализация

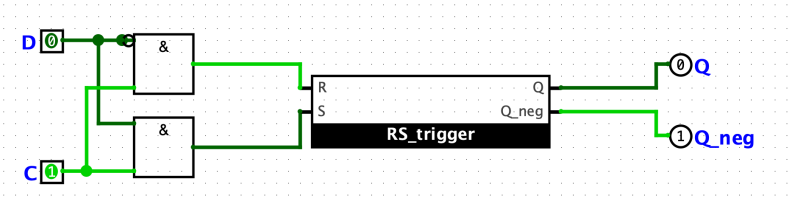

Рис. 2: Реализация D триггера

Reset-T_trigger

Теперь определим новый элемент: reset-T триггер. Это элемент с двумя входами T и R, который можно назвать "счётчик по модулю два с кнопкой сброс". Из второго названия понятно, что он должен делать. Данный прибор выдаёт на выходе количество раз, которое элемент T становился единичкой по модулю 2. Но если в любой момент подать сигнал на вход R, прибор обнулит выход (сбросит счётчик). Результат увеличивается, когда T меняется с 0 на 1.

Ниже представлена таблица истинности для reset-T триггера

T	R	Q
0	0	сохраняем значение
1	0	инвертируем значение
0	1	0
1	1	0

Таблица 3: Таблица истинности reset-T триггера.

Реализован этот элемент через 2 последовательных D триггера с противоположной синхронизацией. За один тик (изменение входа T $0 \to 1 \to 0$) первый D триггер запоминает противоположный выход второго, а второй запоминает выход первого (в итоге происходит та самая инверсия). Данная реализация без R-входа находится во вкладке $T\_trigger$.

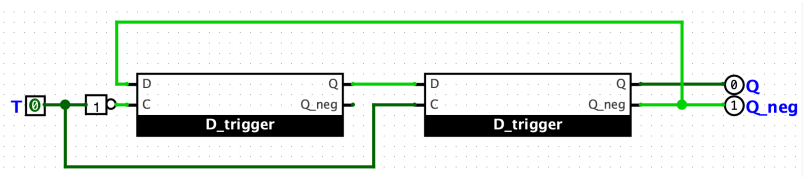

Рис. 3: Реализация T триггера

Ниже представлена реализация

Далее мы хотим, чтобы при $R = 0$ схема не изменилась, а при $R = 1$ оба D триггера в схеме стали давать на выходе 0. Сделаем это таким образом. Перед каждым входом поставим логическое ИЛИ, а перед каждым D входом логическое И. Провода, которые вели к этим входам подведём к первым входам соответствующих логических элементов. В качестве второго входа компонента ИЛИ будет R, а в качестве второго входа И будет $\overline{R}$. Тогда при $R = 0$ схема не изменилась так как $\overline{R} \wedge a = a$, $R \vee a = a$. А при $R = 1$ на входы к обоим D триггерам поступит значения $D = 0, C = 1$ и они оба на выходе дадут 0. Данная реализация представлена в схеме Reset_T_triger.

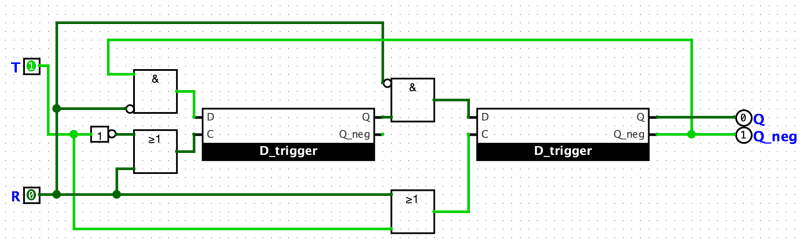

Рис. 4: Реализация reset-T триггера

is_17

Модуль, который проверяет то, что входное число равно 17 это логическое И, которое по 5 входным битам (записи числа в двоичной системе с 5ю знаками) выдаст 1 только если из входных битов получается число 17. Реализация в схеме is_17.

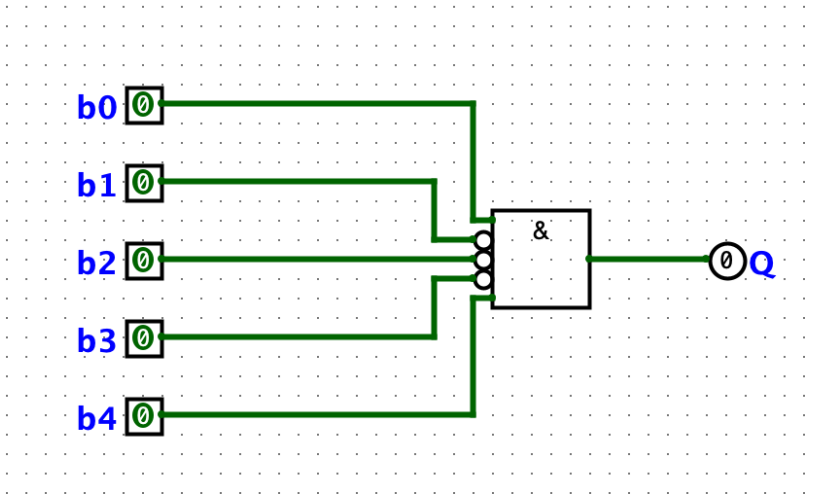

Рис. 5: Реализация is_17

counter_32

Это асинхронный суммирующий счётчик по модулю 32. Его можно понимать, как reset-T триггер, у которого модуль вычисления равен 32. То есть этот элемент имеет 2 входа i и R. На выходе он выдаёт 5 битов — количество раз, которое изменили вход i с 0 на 1 по модулю 32 в двоичной системе счисления. Но так же как и в reset-T триггере у него есть вход R, который обнуляет все внутренние reset-T триггеры.

Работает он таким образом. Мы последовательно подключаем несколько reset-T триггеров, вход T следующего соединён с отрицанием выхода предыдущего.

i	0	1	0	1
$\overline{Q}$	1	0	0	1

Таблица 4: Вывод самого левого reset_T триггера в при 4 первых значениях бита i

Таблица 4 указывает на то, что за 2 перехода с 0 на 1 у нас произойдёт всего 1 переход с 0 на 1 на отрицании выхода. Это значит, что у нас период смены 0 на 1 уменьшается в 2 раза, а следовательно этот выход можно использовать как вход в следующем reset-T триггере. тем самым, каждый раз, когда у нас будет Q становиться 0, $\overline{Q}$ будет становиться единицей и в новом reset-T триггере у нас будет симулироваться "переход через десяток". Вход R подсоединён ко всем R-входам reset-T триггеров.

Реализация в схеме counter_32.

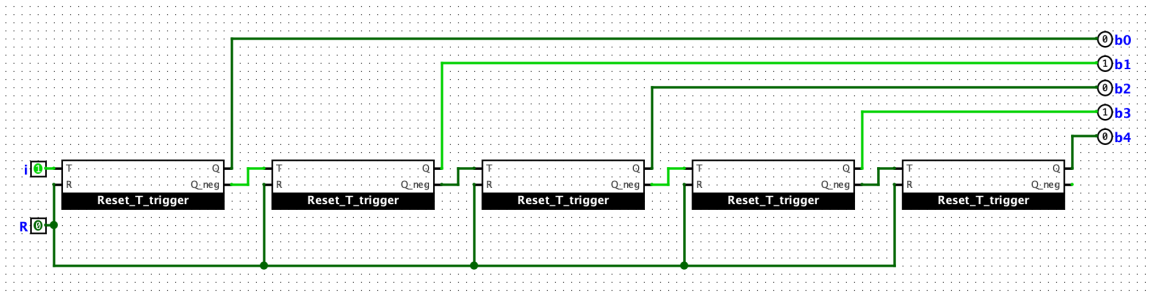

Рис. 6: Реализация counter_32

counter_17

Объединим 2 предыдущие схемы таким образом, как это сделано в схеме main. Тогда, как только counter_32 выдаст значение 17, модуль is_17 отправит сигнал сброса и сбросит счётчик до 0. Тем самым мы реализовали первую часть лабораторной.

Находится в схеме main.

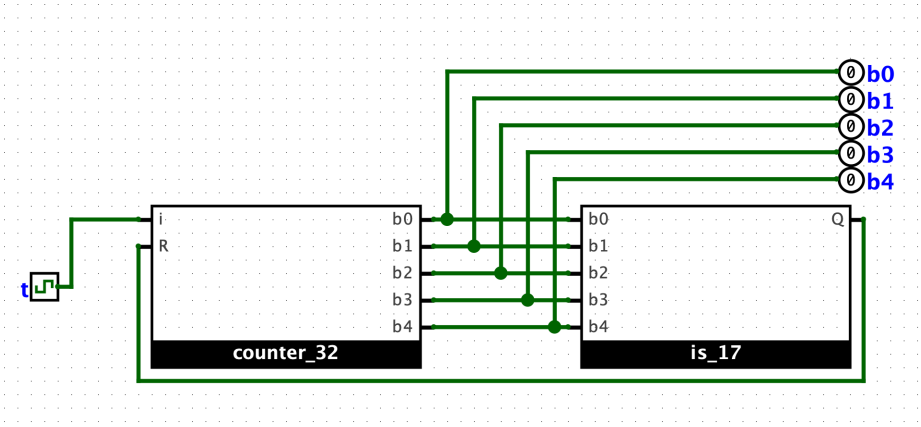

Рис. 7: Реализация counter_17

Временная диаграмма представлена рисунком ниже

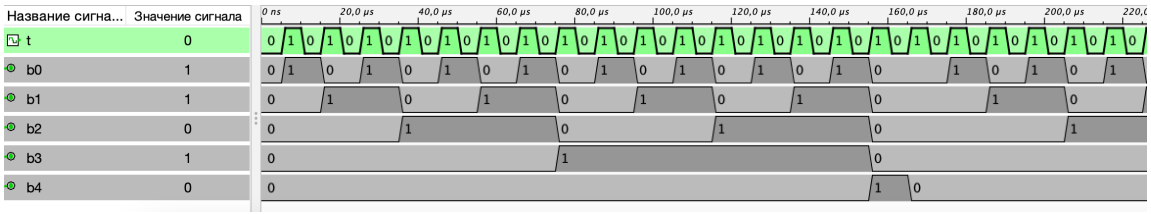

Рис. 8: Временная диаграмма

Регистр сдвига с линейной обратной связью.

Вариант Конфигурация Галуа. (12, 11, 8, 6).

Описание В работе представлен регистр сдвига с линейной обратной связью в конфигурации Галуа. Регистр сдвига — это устройство, которое осуществляет сдвиг цифрового слова, которое дано на вход, по цепочке триггеров. Линейная обратная связь означает, что некоторые внутренние триггеры линейно изменятся в зависимости от выхода регистра сдвига.

Для реализации нам потребовались RS и D триггером из предыдущей части а также асинхронный D триггер.

D_async

Это асинхронный D триггер. Его суть в том, что при синхронизации 1 он «работает на вход», а при изменении синхронизации с 1 на 0 запоминает значение на входе и выдаёт его на выходе "работает на выход". Он нужен для того, чтобы за один такт бит информации распространился ровно на один шаг вперёд. (используя обычные D триггеры при синхронизации всё резко станет либо 1, либо 0).

Реализация в схеме D_async

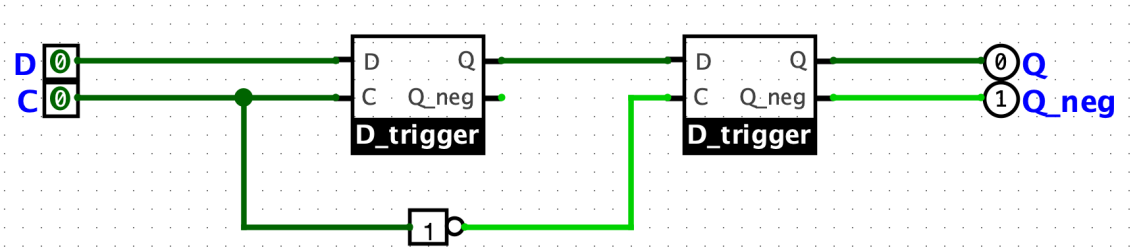

Рис. 9: Реализация D_async

LFSR

Регистр сдвига с линейной обратной связью. Он хранит последовательность из 12 битов и каждый тик он сдвигает эту последовательность на один вправо по циклу, а самый последний бит каждый тик ксорит с выделенными в конфигурации (11,8,6). Реализован с помощью асинхронных D триггеров, которые соединены последовательно. Выходным битом является самый последний, который выдаёт псевдослучайную последовательность. Чтобы настраивать входные данные добавлен контакт и оператор XOR.

Реализация в схеме main

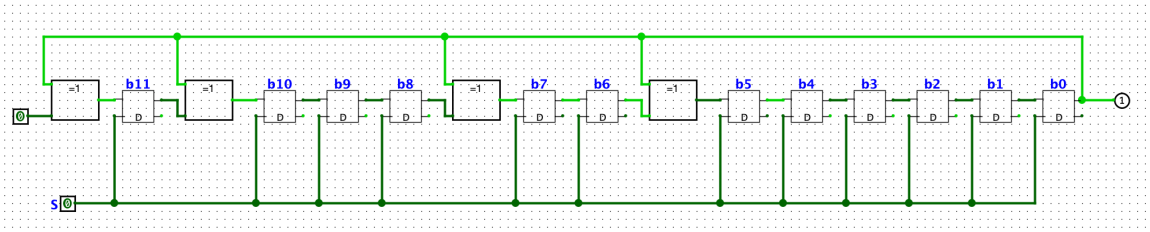

Рис. 10: Реализация LFSR