Лабораторная работа №1	Группа M3138	2022
Название работы Построение логических схем в среде моделирования	ФИО Селезнев Дмитрий Александрович	

Цель работы: моделирование логических схем на элементах с памятью.

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

Счетчик

Описание

Счетчик - устройство для подсчёта числа входных импульсов.

Счетчики делятся на:

- Суммирующие
- Вычитающие

Суммирующие счетчики изменяют свои значения от 0 и до m-1, где m – модуль счета. Вычитающие наоборот от m-1 до 0.

Также счетчики бывают:

- Асинхронные
- Синхронные

Асинхронный счетчик – счетчик, в котором каждый последующий разрядный триггер синхронизируется выходными импульсами предыдущего, то есть триггеры меняют свои состояния последовательно, в отличие от синхронного, где сигнал подается на все триггеры одновременно, и состояния меняются параллельно.

Вариант

В моем варианте требовалось построить асинхронный суммирующий счетчик по модулю 15. То есть требуется 4 разрядных триггера.

Для начала рассмотрим несколько вспомогательных схем.

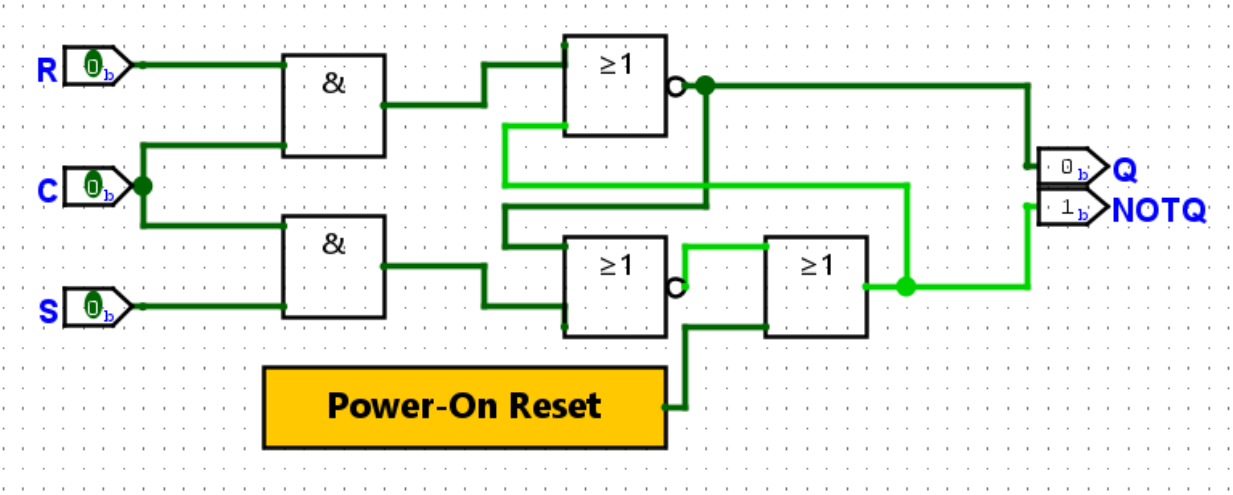

Рисунок №1 – RSCTrigger

Вот таблица истинности схемы на рисунке выше:

R	S	Q
0	0	Сохр
0	1	1
1	0	0
1	1	

Таблица №1 – Таблица истинности RS-триггера

На схеме изображен синхронный RS-триггер, где Q – выход, а NOTQ – инвертированный выход.

Во-первых, стоит отметить, что какие-либо изменения на входах возможны лишь при синхронизации (то есть C) равной 1, иначе S, R не проходят дальше элементов «и».

Во-вторых, RS-триггер – это простейшая ячейка памяти, которая при двух нолях, сохраняет значение на входах, так как не меняет значение на выходе элементов «или-не», полученное ранее. Когда входы имеют вид: R=0, S=1, то нижний элемент «или-не» возвращает 0, тогда верхний вернет 1 и на выходе Q будет 1. Зеркальная ситуация в случае R=1, S=0, на выход NOTQ вернется 1, следовательно Q=0.

Красная ячейка обозначает неопределенное/запрещенное состояние (то есть при данной ситуации на выходе будет что угодно). Power-On Reset (POR) нужен, чтобы установить значение по умолчанию в триггер.

Теперь рассмотрим схему JK-триггера.

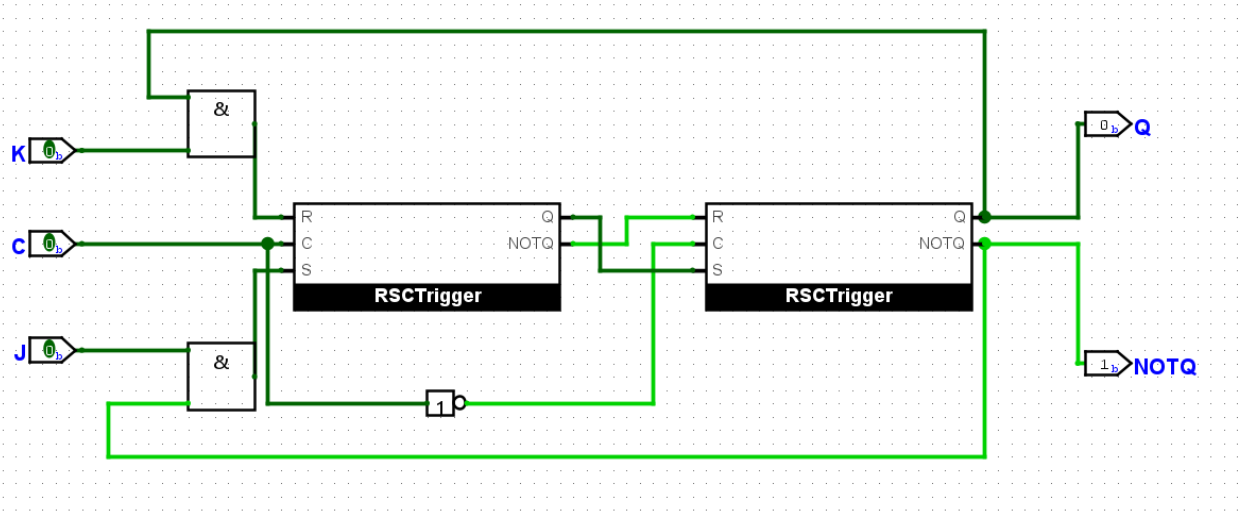

Рисунок №2 – JKTrigger

Его таблица истинности:

J	K	Q
0	0	Сохр.
0	1	0
1	0	1
1	1	Инв.

Таблица №2 – таблица истинности JK-триггера

Он очень похож на синхронный RS-триггер, только при поступлении двух единичек инвертирует значение, записанное в ячейке памяти.

Как он работает: при поступлении разных значений на входах J и K и синхронизации (вход C), он записывает значение в первую ячейку памяти. При отключении синхронизации инвертированный сигнал синхронизации подается на вторую ячейку и значение из первой переписывается туда, из-за того что значение во вторую ячейку записывается (а следовательно и подается на выход) с отставанием в пол такта получается JK-триггер работающий по второму фронту.

Далее рассмотрим блок Mult, который реализует стандартный мультиплексор 2 в 1.

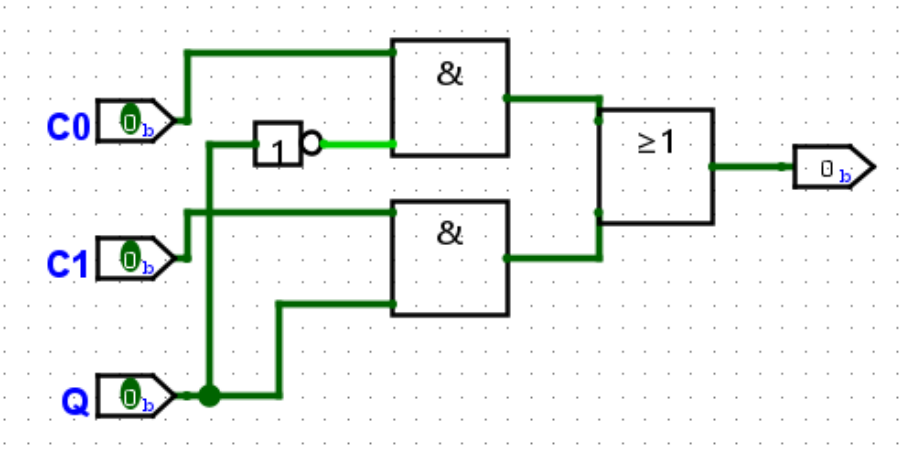

Рисунок №3 - Mult

Как это работает: при Q=0, нижнее «и» обратится в 0, а из верхнего выйдет значение C0, при Q=1 наоборот, верхнее обратится в 0, а нижний вернет C1, то есть пропустят дальше только значение на нужном входе. Далее значения на выходах передаются в «или», из которого на выход пойдет единичка, если на соответствующем значению Q входе была подана единичка и 0 в остальных случаях. Это и требуется от мультиплексора.

Сейчас рассмотрим триггер, на основе которого я собираюсь сделать счетчик.

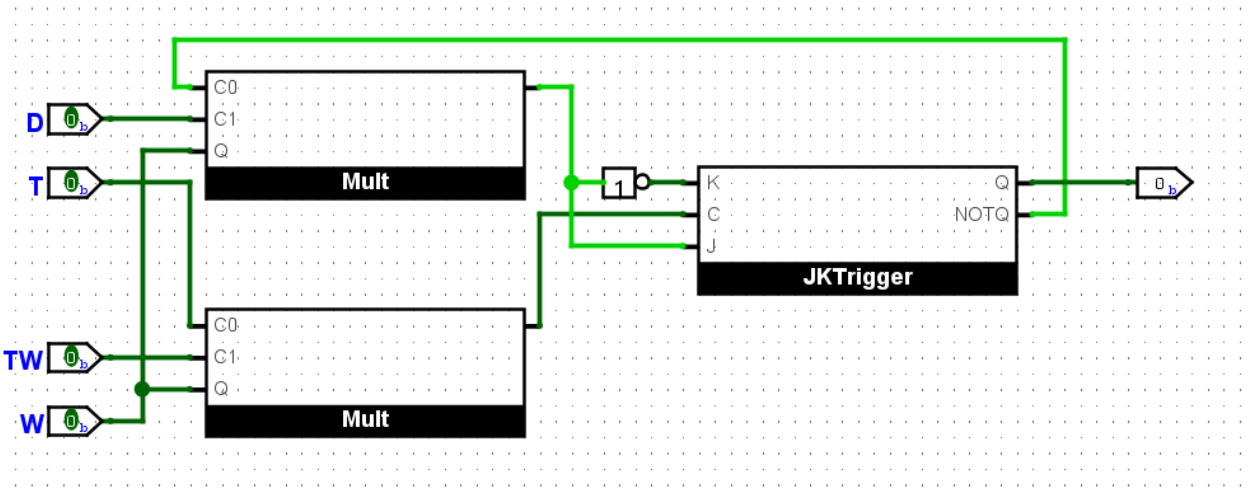

Рисунок №4 - MyTrigger

Для начала нужно сказать, что он умеет. При поступлении сигнала меняет своё значение на противоположное, то есть имитирует прибавление единички, а в специальном случае (о нем будет позднее) записывает в себя константное значение, поданное на вход D.

Теперь рассмотрим, как он работает. Когда на входе W=0, то триггер меняет своё значение на противоположное, записывая значение с NOTQ, так

как из нижнего блока Mult выходит сигнал синхронизации (вход T), а с верхнего – NOTQ. Если W=1, то с нижнего блока Mult выходит сигнал из TW, который отвечает за возможность записи (по сути, другой сигнал синхронизации), а с верхнего D, которое и запишется в ячейку памяти. Также стоит отметить, что все изменения происходят по второму фронту, так как JK-триггер работает по второму фронту. Следовательно, изменения состояний триггеров подключенных последовательно будут изменяться асинхронно, а не синхронно, что нужно в счетчике.

Перед тем как рассмотреть основную схему счетчика стоит рассмотреть ещё несколько вспомогательных схем.

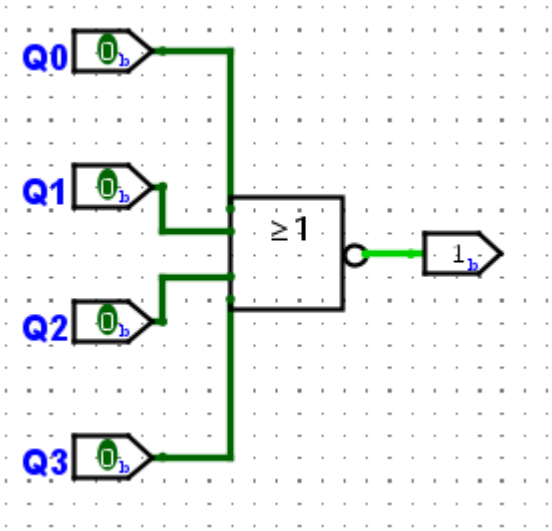

Рисунок №5 - MyModule

Схема на рисунке 5 возвращает 1 только когда на входах все 0 за счет или-не для 4 входов.

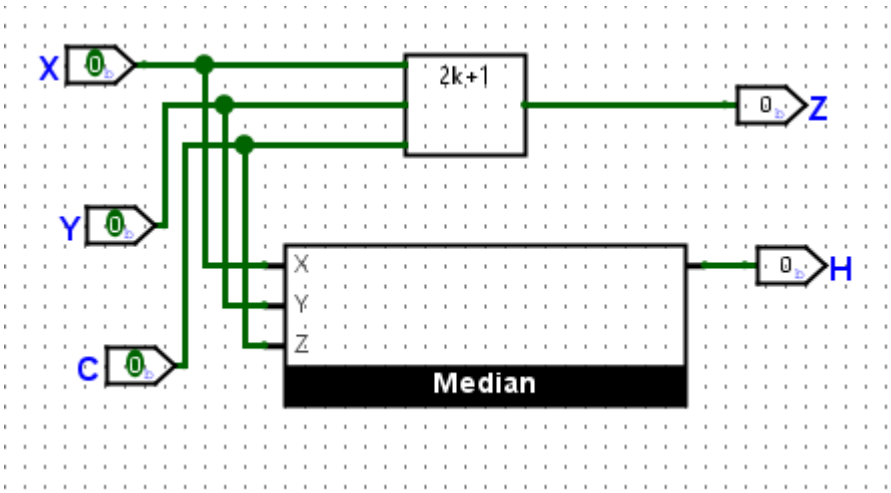

Рисунок №6 – FullAdder

На рисунке 6 изображен полный сумматор, то есть сумматор с возможностью получить перенос бита из предыдущего разряда. На выход Z подается значение полученное на этом бите, на выход H – перенос разряда. Блок «2k+1» это xor 3 чисел, то есть сумма трех по модулю 2, что нам и нужно. Блок Median возвращает 1, если на входах больше 1, иначе 0, что и требуется от переноса разряда. Его схема:

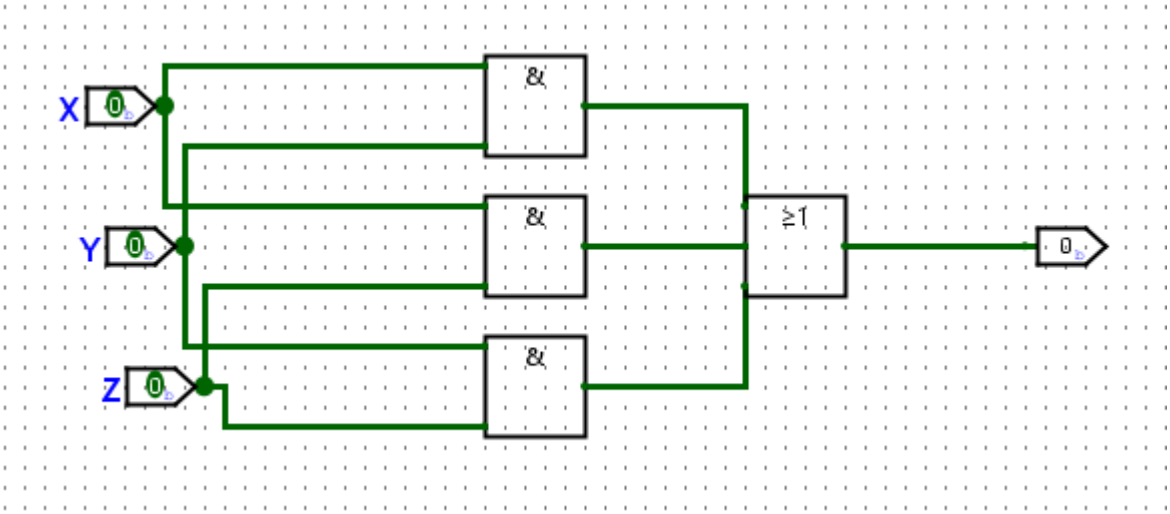

Рисунок №7- Median

По сути, он возвращает 1, если хотя бы на одном из элементов «и» получилось 1, то есть хотя бы на двух входах единички.

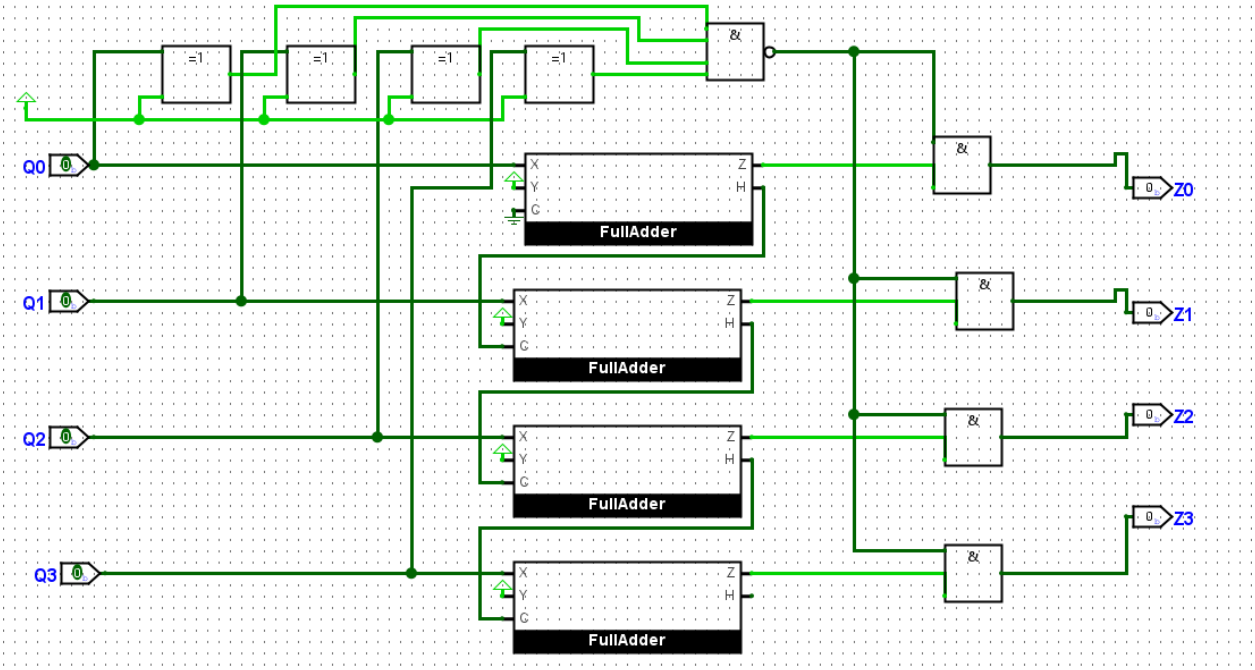

Рисунок №8 – Plus

На рисунке 6 изображен обычный сумматор для 2 чисел длинной 4 бита по модулю 16. Все элементы «или» и «и» отвечают за то, чтобы при четырех нолях на входах, то есть при 0_{10}, выход остался 0 (для чего это сделано будет сказано при рассмотрении основной схемы). В данном случае к значению на входе прибавляется 15, то есть вычитается 1 по модулю 16.

Теперь можно перейти к основной схеме.

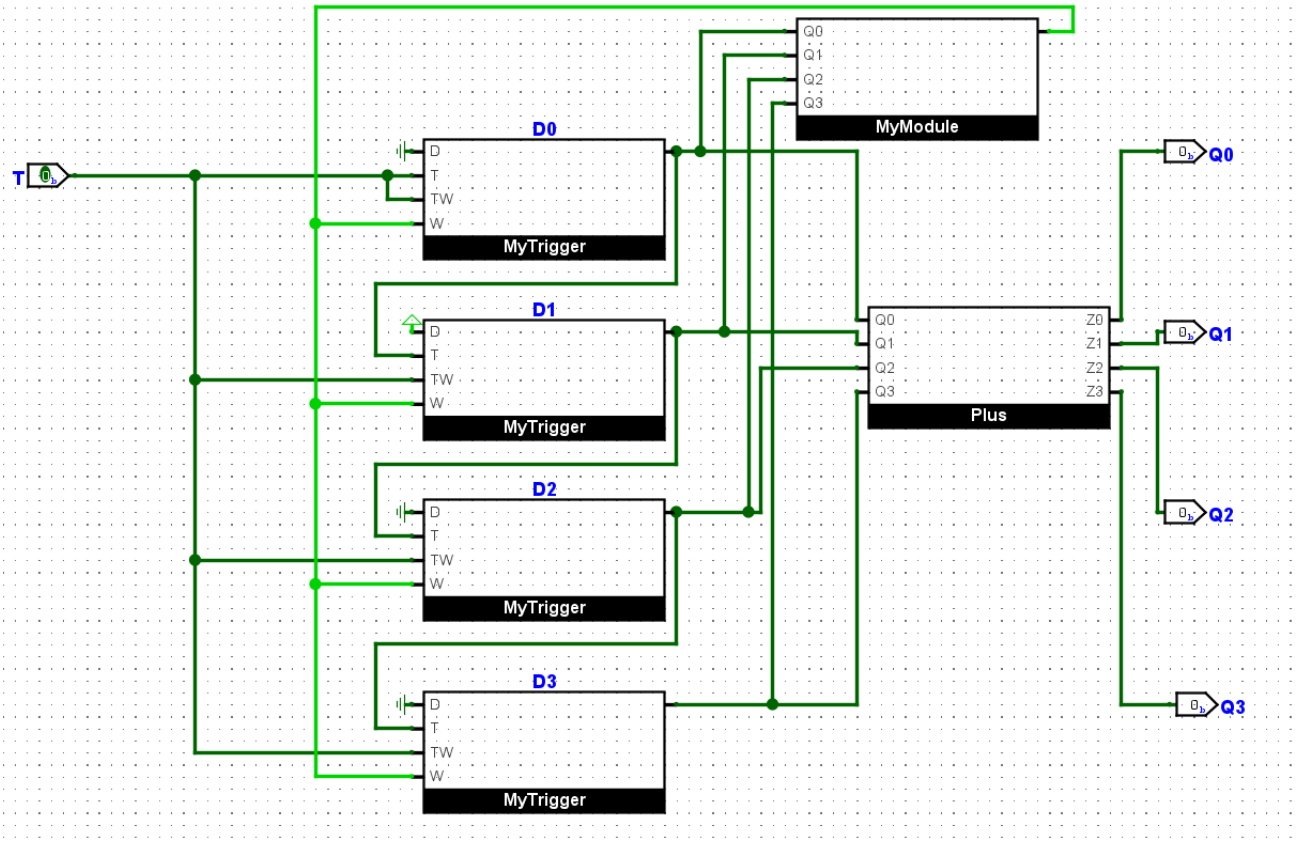

Рисунок №9 - main

Вход только 1, это T, то есть такт. Эта схема лежит в файле main, при запуске программы нужно немного подождать, чтобы POR отработал. Далее можно изменять значение на входе, и на выходах будут появляться корректные значения.

Рассмотрим схему поподробнее. На схеме видно, что такт подается только в первый триггер (из схемы MyTrigger ясно, что вход TW идеологически отвечает не за такт, а за возможность записи в триггер значения передающегося на D), а дальше значение выхода на первом триггере передается в следующий, и так далее, то есть счетчик действительно асинхронный. То состояние, в котором он показан на схеме, это его начальное состояние, видно, что на выходе 0, а MyModule возвращает 1 (это и есть специальный случай, о котором я упоминал ранее). Тогда при поступлении сигнала, триггеры сработают на запись значения из D, то есть

запишут $0010_2=2_{10}$, MyModule станет возвращать 0. При поступлении следующих сигналов триггеры будут увеличивать хранимое значение за счет инверсии, пока не станут равны 16_{10}, то есть 0000_2, так как всего 4 бита, тогда MyModule вернет 1, и снова запишется 2, то есть значения перебираются от 2 до 15, потом 0, пропускается 1, и так по кругу. Перед выходом ответа все значения из триггеров отправляются в блок Plus, в котором значение из триггеров уменьшаются на 1, для всех чисел, кроме 0, то есть на выходе будут значения от 0 до 14, то и требовалось для счетчика в моем варианте.

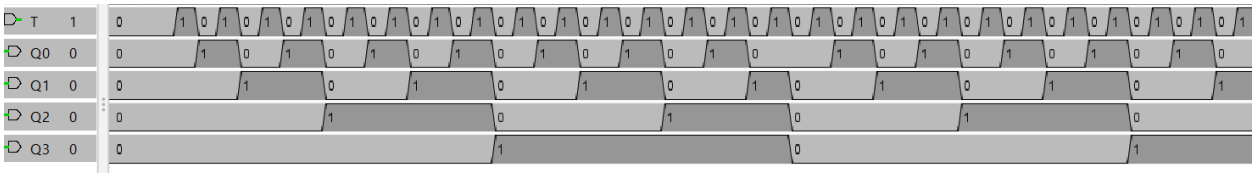

Рисунок №10 – временная диаграмма для счетчика

Регистр сдвига с линейной обратной связью

Описание

Регистр сдвига с линейной обратной связью – сдвиговый регистр, значение входного бита которого вычисляется в зависимости от некоторой линейной функции. Нужен для генерации псевдослучайной битовой последовательности.

Вариант

Мне нужно было реализовать сдвиговый регистр в конфигурации Галуа, с отводными битами (12, 6, 4, 1) следовательно, мне необходимо 13 ячеек памяти, чтобы хранить состояния регистра. Отводные биты – биты, задающие функцию. В регистре Галуа нужно каждый отводной бит перед сдвигом xor-ить с выходным битом, который после сдвига запишется в последнюю (входную) ячейку. Ячейки и биты нумеруются с 0 до 12 , номер ячейки совпадает с номером бита.

При сборке схемы мне также потребовались RS и JK триггеры, схемы которых можно найти выше. Однако мне также потребовался новый триггер. Вот его схема:

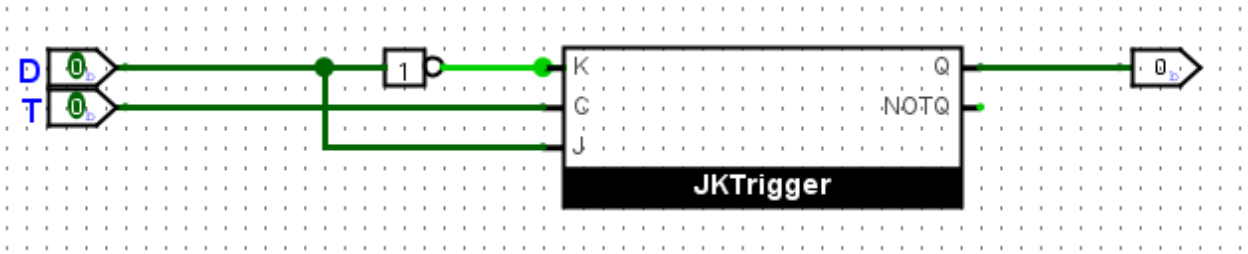

Рисунок №11 – MyTrigger

Так как в регистре все ячейки будут соединены последовательно, мне также как и в счетчике, нужна ячейка памяти, работающая по второму фронту. В этом триггере записывается значение с входа D, на каждом такте T.

Теперь рассмотрим основную схему.

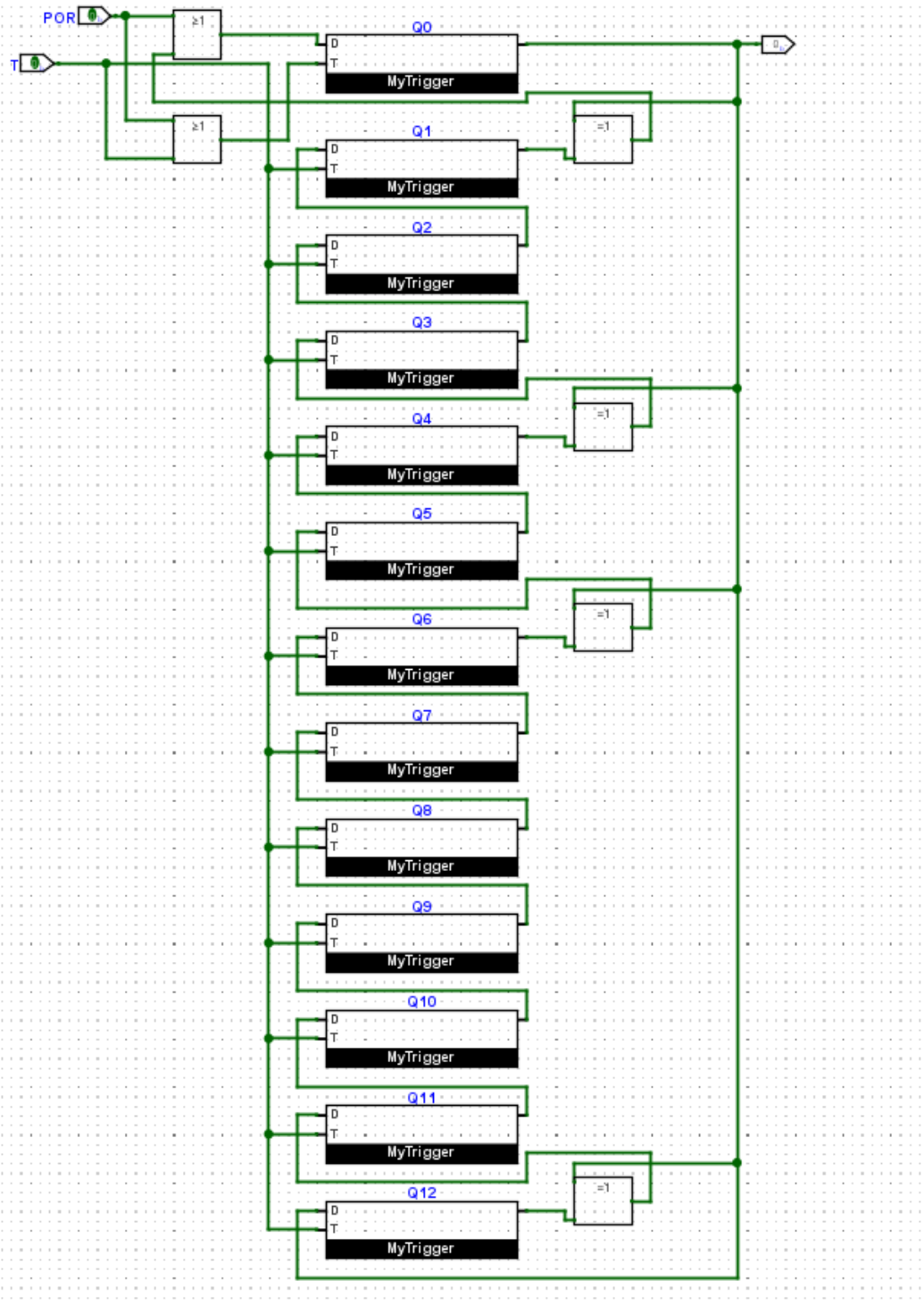

Рисунок №12 – main

При запуске нужно немного подождать, чтобы POR из RS-триггера отработал, затем нужно два раза нажать на POR из main, чтобы в регистр записалась 1, иначе на выходе всегда будет 0 и последовательность не сгенерируется. Q0 – выходной бит, Q12 – входной бит.

Теперь рассмотрим схему поподробнее. В регистре необходимо, чтобы выходной бит xor-ился с отводными битами перед передачей их в следующую ячейку, это и происходит на схеме. Следовательно, регистр работает корректно.