Лабораторная работа №1	М3137	2022
Построение логических схем в среде моделирования	Нартов Дмитрий Николаевич	

Цель работы: моделирование логических схем на элементах с памятью.

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

Описание

Разработать схемы счетчика и РСЛОС с определенной конфигурацией и описать их принцип работы.

Вариант

Асинхронный суммирующий счетчик.

РСЛОС с конфигурацией Галуа, (13, 4, 3, 1).

Асинхронный суммирующий счетчик с модулем.

Для построения счетчика нужно собрать несколько подсхем. Первой будет RS-триггер, второй - JK-триггер.

RS-триггер.

Для того, чтобы собрать более сложные схемы и триггеры нужно сначала сделать RS-триггер.

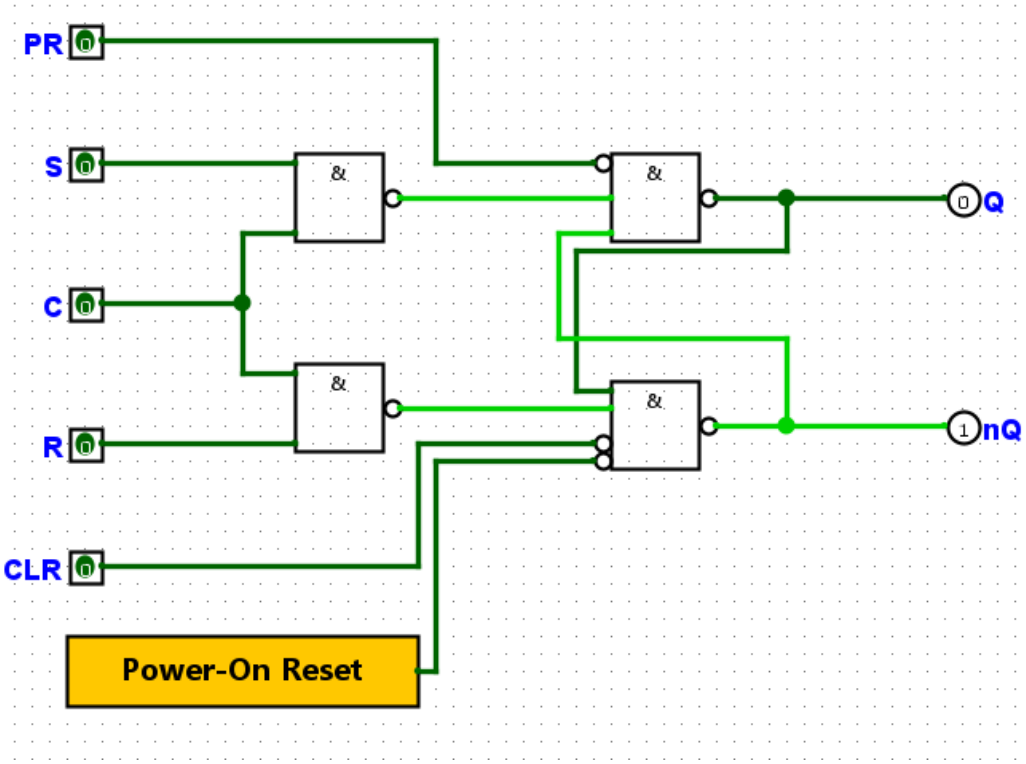


Рисунок №1 - RS-триггер.

Я выбрал схему построения триггера (рис. №1) на элементах NAND. Также было добавлено несколько входов. Первый вход - это C, который отвечает за синхронизацию (RS-триггер, конечно, может работать и без неё, но для построения более сложных триггеров, синхронизация нам необходима). Второй - PR (preset). При подаче на него сигнала, он ставит наш триггер в положение, когда выход Q = 1, а nQ = 0. При подаче сигнала

на CLR, триггер становится в положение: Q = 0, nQ = 1. Отсутствие сигнала на обоих входах никак не влияет на наш триггер. Одновременная подача сигнала на входы нецелесообразна. Также в схему добавлен POR (power-on reset), который выполняет такую же функцию, как и CLR. В начале симуляции он сбрасывает наш триггер.

R	S	Q	nQ
0	0	$Q_{сохр.}$	$nQ_{сохр.}$
0	1	1	0
1	0	0	1
1	1	не определенно	не определенно

Таблица №1 - Таблица истинности RS-триггера по входам S и R.

CLR	PR	Q	nQ
0	0	$Q_{сохр.}$	$nQ_{сохр.}$
0	1	1	0
1	0	0	1
1	1	не определенно	не определенно

Таблица №2 - Таблица истинности RS-триггера по входам CLR и PR.

Можно заметить, что таблицы идентичны, но на самом деле мой RS-триггер синхронизирован, а входы CLR и PR позволяют менять внутреннее состояние триггера независимо от синхронизации.

JK-триггер.

Следующей подсхемой будет JK-триггер, который работает несколько сложнее.

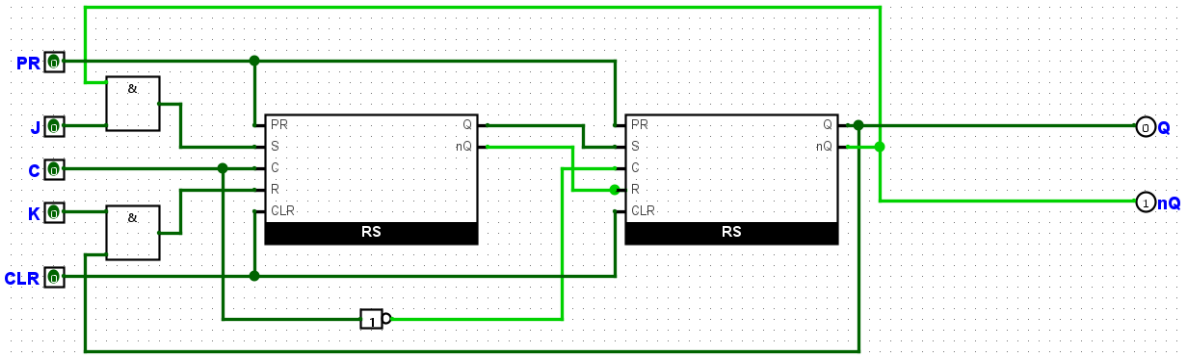


Рисунок №2 - JK-триггер.

Обычный JK-триггер, но в который добавили два новых входа: PR и CLR. Они выполняют такие же функции, как и в предыдущем триггере. Реализуем функционал этих входов просто подводя их в соответствующие входы на RS-триггерах, как показано на рис. № 2.

Если остановиться на устройстве и принципе работы JK-триггера - это RS-триггер, но который на J = 1 и K = 1 инвертирует значение, которое содержиться внутри и выводит его.

J	K	Q	nQ
0	0	$Q_{сохр.}$	$nQ_{сохр.}$
0	1	0	1
1	0	1	0
1	1	nQ	Q

Таблица №3 - Таблица Истинности JK-триггера.

Таблица Истинности для входов PR и CLR такая же, как и табл. №2.

Во внутреннем устройстве есть некоторая хитрость для того, чтобы реализовать инверсию. Второй RS-триггер как раз и позволяет выполнить эту операцию корректно.Он начинает работать только спустя полтакта, тем самым не давая бесконечно инвертироваться нашей схеме.

Асинхронный суммирующий счетчик с модулем.

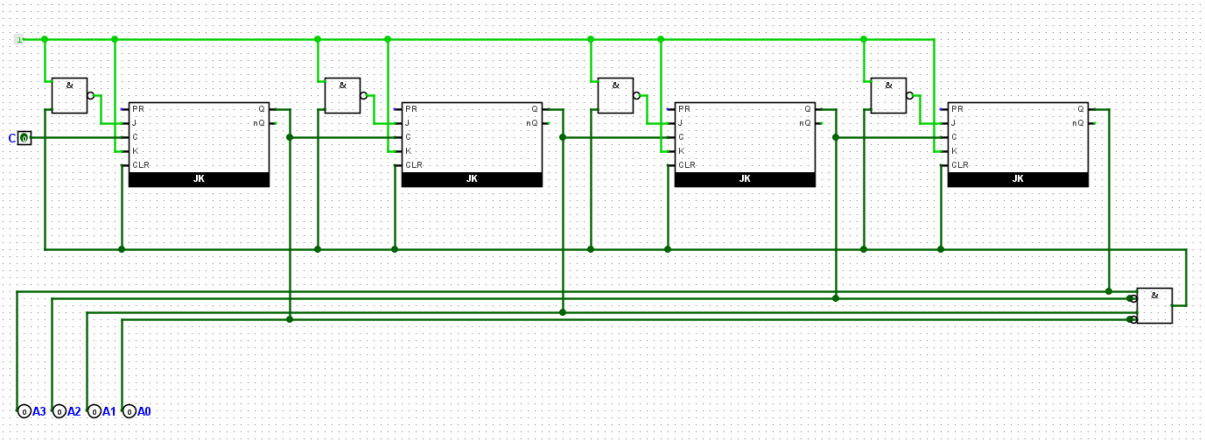


рисунок №3 - асинхронный суммирующий счетчик по модулю 10.

При построении счетчика я пользовался следующим материалом: https://www.youtube.com/watch?v=l20xHDJPHBM&t=653s&ab_channel=Nes

oAcademy. Конечно, пришлось добавить некоторые детали. Для реализации счетчика нам нужны JK-триггеры. Их понадобиться ровно 4 штуки, так как мой модуль равен 10, а для того, чтобы закодировать однозначно 10 значений нам нужно 4 бита, а в схеме за один бит отвечает один триггер, следователь минимальное количество триггеров позволяющее создать счетчик равно 4.

Для реализации схемы были выбраны JK-триггеры, так как они позволяют реализовать инверсию, на которой и основан принцип счетчика. Формально у нас храниться 4-битное число, к которому на каждом шаге прибавляется 1. Прибавление в двоичных числах - это найти первый равный 0 справа бит и сделать побитовую инверсию на суффиксе, который оканчивается этим элементом. Значит нам нужен триггер способный менять свое состояние на противоположное с подачей на него синхронизации. Ровно такой триггер мы можем реализовать используя JK-триггер. На вход синхронизации первого триггера подается синхронизация, на входы синхронизации всех остальных - выход Q предыдущего триггера. Именно эта деталь делает данный счетчик асинхронным.

Для того, чтобы нас счетчик работал по какому-то модулю, нужно Добавить один элемент AND, который будет выдавать 1, только когда мы получаем значение равное модулю. Можно это реализовать, ведь модуль представим в двоичной системе счисления, а двоичное число - это упорядоченный набор битов, следовательно можно сделать такую функцию, которая возвращает единицу только на таком наборе входных данных. А такая функция - это AND. Сигнал с этого AND подведем ко входу CLR у всех триггеров. Другими словами в момент, когда наш счетчик достигает модуля срабатывает AND, который сбрасывает все триггеры и мы оказываемся в состоянии, когда счетчик равен 0. В моем

случае нужно подвести к AND первый и третий бит (на схеме у меня также подведены нулевой и второй биты с инверсией, что необязательно, но делает схему нагляднее).

Также важно подключить входы J и K у всех триггеров. Так как нам всегда нужна инверсия, значит на J и K можно подавать только 1. Но здесь нужно быть аккуратным, так как в момент сброса триггеров некоторые RS-триггеры, которые находятся внутри JK-триггеров, могут встать в положение, когда Q = 1 и nQ = 1. Мы такого не хотим, поэтому перед входом J поставим элемент NAND, в который будут входить сигнал с тождественной 1 и сигнал с AND, который отвечает за сброс. Таким образом мы не получим такого состояния, когда RS-триггер сломается.

Для вывода двоичного числа, просто сделаем вывод с выхода Q для каждого JK-триггера, где триггер, который стоит первым в схеме отвечает за младший бит, а последний за старший.

Временная диаграмма.

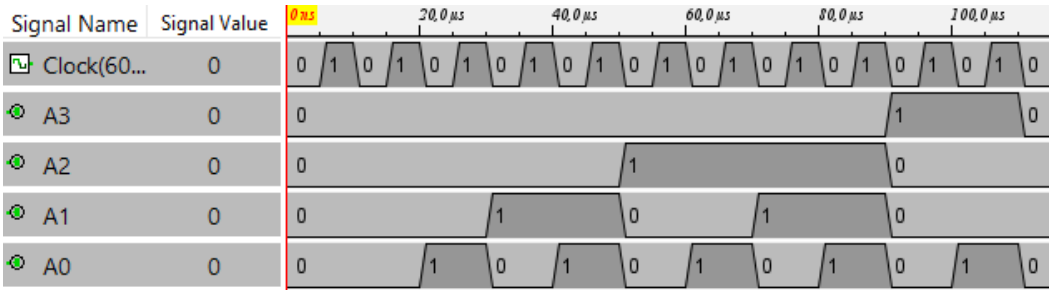


Рисунок №4 - временная диаграмма счетчика.

По схеме видно, что у нас происходит синхронизация по уровню. Задержка в начале на 2 секунды происходит из-за элемента power-on reset. По диаграмме отчетливо видно принцип с инверсией суффикса, который был описан выше. На первых 2 тактах мы имеем 0, затем на 3 такте

подается сигнал на первый триггер и он переходит в положение Q = 1. Затем синхронизация попадает во второй триггер на 4 такте, а первый триггер выключается. Дальше схема работает также(постепенно единица будет доходить до новых битов). На 12 такте происходит сброс значения.

Регистр сдвига с линейной обратной связью.

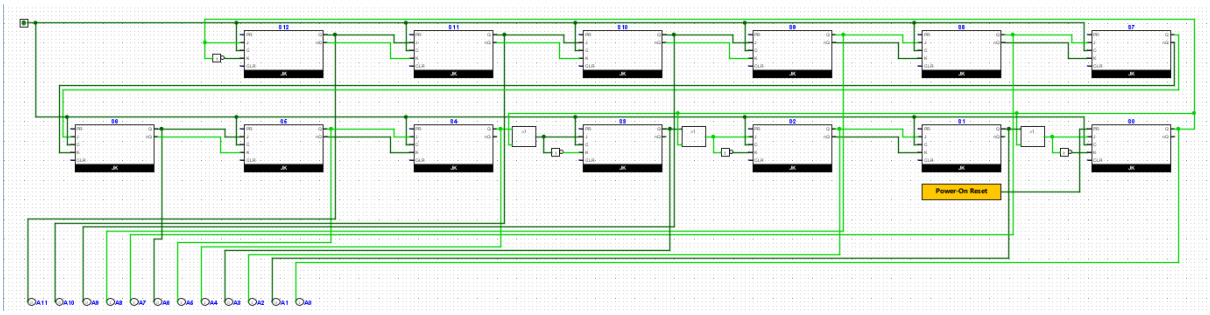


Рисунок №5 - РСЛОС с конфигурацией Галуа.

Для этой схемы тоже нужен RS и JK триггеры. Они точно такие же как и в схеме счетчика. При выполнении данной схемы я руководствовался следующим материалом: https://bit.ly/3CRk28g. Данную схему можно реализовать используя JK-триггеры, сделав из них D-триггеры. Конфигурация дает нам информацию после каких триггеров ставить XOR.

D-триггер - это триггер, который подает на выход, то что и пришло на единственный вход, но с учетом синхронизации. В данном случае удобно его реализовать на JK-триггерах, так как нужно запоздание, чтобы на первом шаге 1 не проникла во все триггеры.

Реализация триггера крайне проста, так как JK-триггер - это RS-триггер, но в котором при подаче двух единиц на входы реализуется инверсия. Так как у нас случай с двумя единицами никогда не используется, строим D-триггер на JK-триггере, на вход J подаем сигнал, а на K инверсию этого же сигнала. Получается следующее:

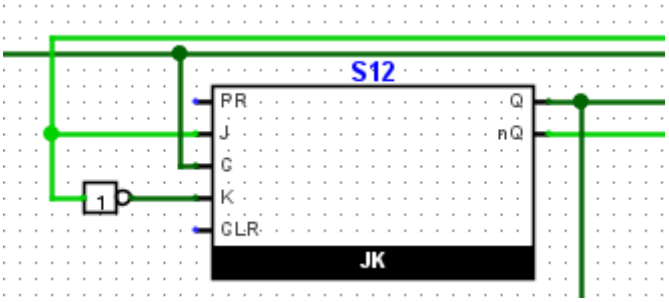


Рисунок №6 - D-триггер на JK-триггере.

Схема счетчика состоит, в моем случае, из 13 D-триггеров, где каждый следующий подключен к предыдущему, но между некоторыми стоит XOR, которые выдает значение основываясь на сигнале из предыдущего триггера и сигнала из 0 триггера.

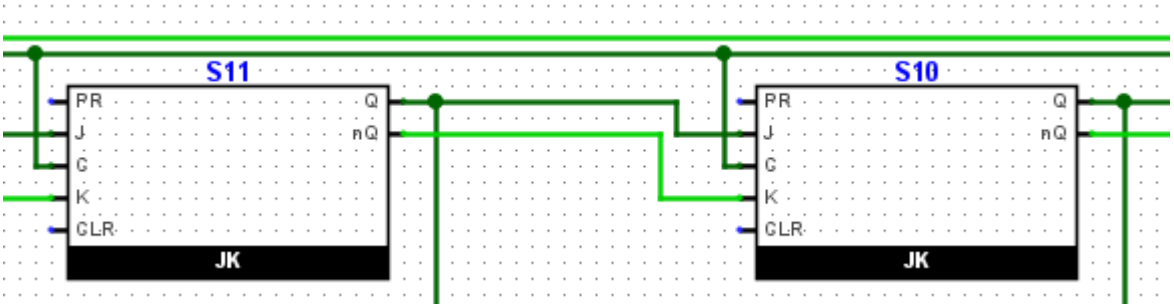
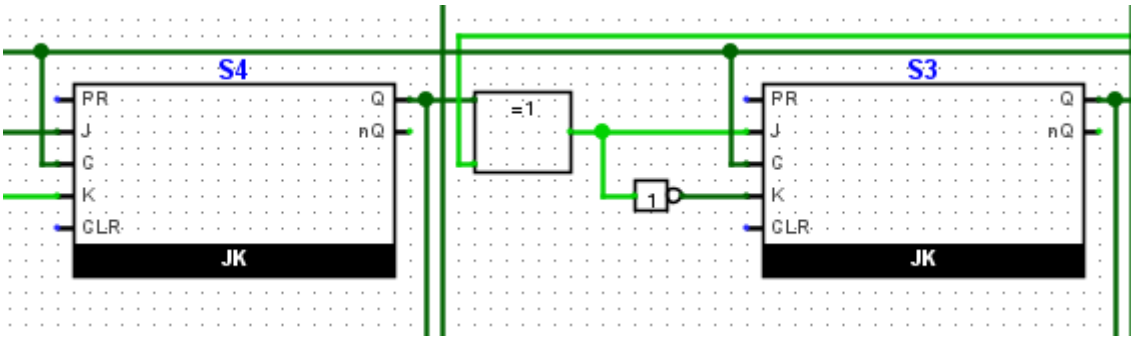


Рисунок №7 - Обычное соединение триггеров

В данной ситуации выход Q подключается во вход J, а nQ во вход K.



Рисунок№8 - Соединение триггеров с XOR.

Здесь нужно вставить еще и XOR, и подать на входы как показано на рисунке. Первый триггер принимает на вход сигнал с последнего (последний - это триггер S0, первый - S12).

Принцип работы состоит в переносе сигнала с последнего на первый и сдвига всех остальных значений на 1 бит вперед (за каждый бит - отвечает триггер). XORы стоящие между некоторыми значениями помогают нам получать новые значения для битов. Данная схема позволяет нам генерировать псевдослучайные числа.

Конечно, перед началом работы счетчик нужно поставить в состояние, когда младший бит равен 1, для этого воспользуемся элементом power-on reset, который подключим к триггеру S0 во вход PR. После этого триггер будет в состоянии Q = 1, nQ = 0.

Выводим биты следующим образом: S0 триггер отвечает за 0 бит, S1 триггер за 1 бит и т. д. Следовательно просто выводим значения с выхода Q каждого триггера.