Лабораторная работа 1

Построение логических схем в среде моделирования

Выполнил: Шибанов Игорь M3136 2022 год

Цель работы: моделирование логических схем на элементах с памятью.

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

Описание: составить счётчик и регистр сдвига с линейной обратной связью

Вариант:

1) Асинхронный вычитающий счетчик по модулю 13.
2) Регистр сдвига с линейной обратной связью. Тип конфигурации – Фибоначчи. Конфигурация (7, 1, 0).

Асинхронный вычитающий счетчик по модулю 13

Принцип работы:

Асинхронный вычитающий счетчик – это такой счетчик, в котором выходное значение по запросу уменьшается на 1 (в моем случае по модулю 13).

В начале своей работы я построил синхронный RS-триггер (Рисунок 1). В проекте он имеет название «RSTrigger».

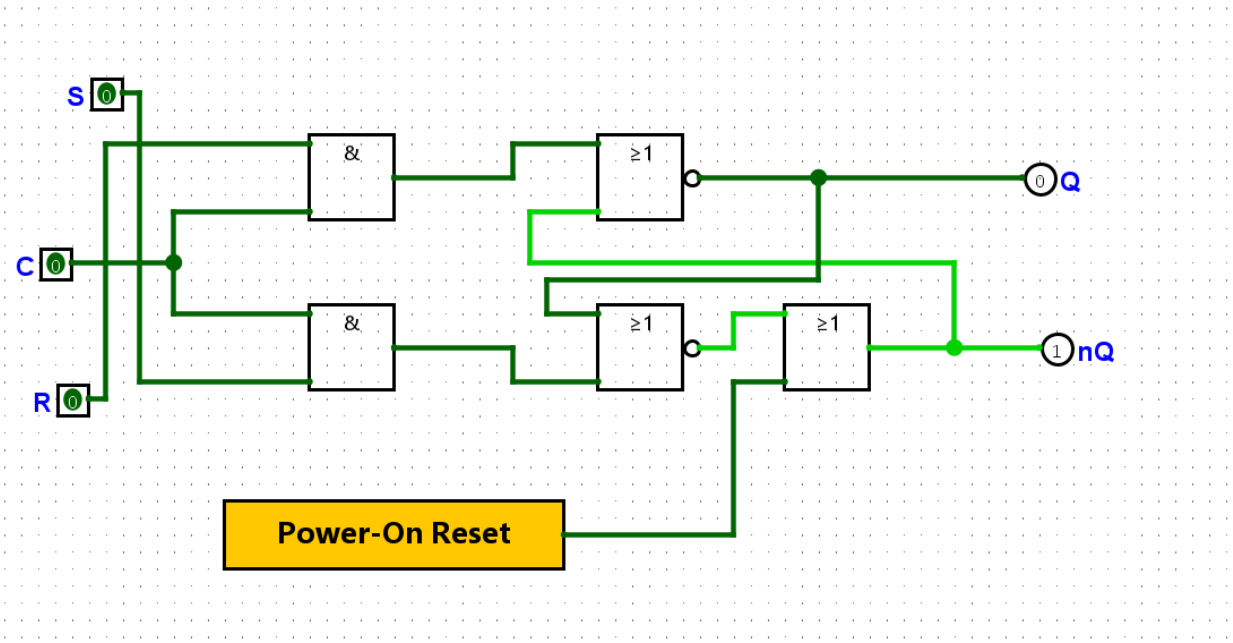


Рисунок 1 – RS триггер

Таблица истинности для RS триггера:

R	S	Q
0	0	сохранение
0	1	1
1	0	0
1	1	запрещено

Отличие синхронного RS триггера от обычного в том, что, когда на контакте синхронизации 0, триггер приостанавливает свою работу.

На основе RS триггера я собрал синхронный JK триггер (Рисунок 2). В проекте он имеет название «JK».

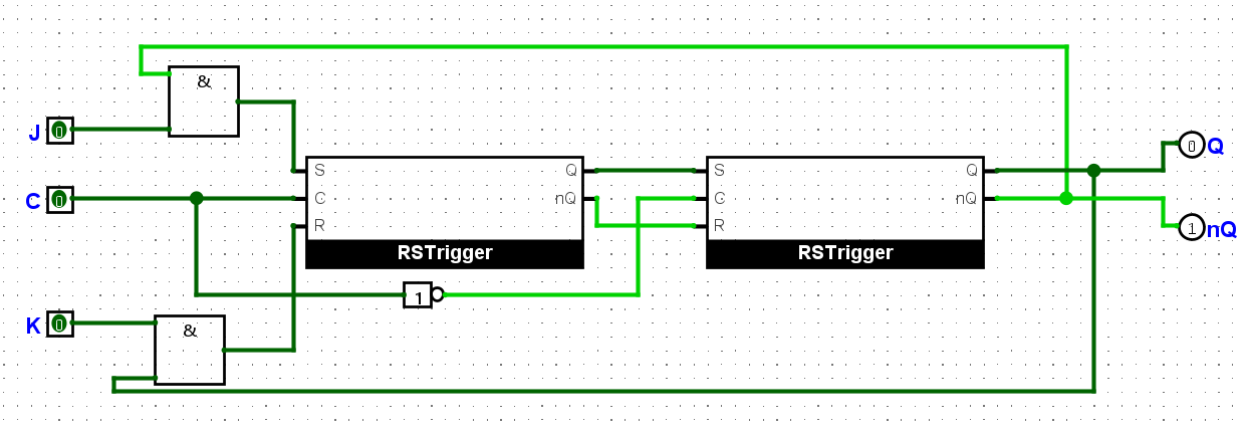


Рисунок 2 – JK триггер (C - контакт синхронизации)

JK триггер работает так же, как и RS триггер, однако он инвертирует предыдущее значение, если подать в него 1 1. Также стоит заметить, что в моей реализации синхронный JK триггер срабатывает только при переходе C из 1 в 0.

Далее на основе JK триггера я собрал еще один триггер (Рисунок 3). В проекте он имеет название «Ttrigger».

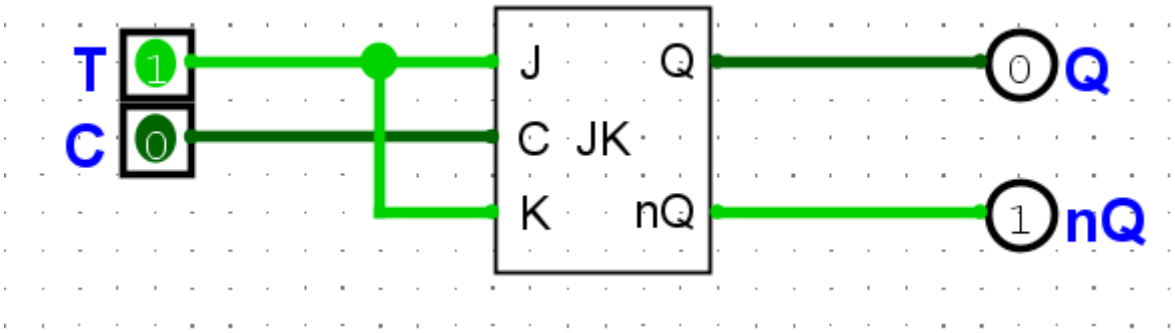


Рисунок 3

В нем я замкнул контакты JK, благодаря чему стало удобно считать по mod 2 (т.е к предыдущему значение мы прибавляем T и берем mod 2). *Примечание*: этот триггер также работает при переходе C из 1 в 0.

После этого я построил асинхронный вычитающий счетчик без модуля (Рисунок 4). *Примечание*: т.к T триггер работает при переходе C из 1 в 0, я инвертирую значение на входе. Когда подаешь первую 1 на вход, во все T триггеры подаются 0, и значения на выходе становятся 1111. После подачи второй 1 на вход, значения на входах первого T триггера изменятся на 0, и значения на выходе становятся 1110. После подачи третьей 1 на вход, значения на входах первого T триггера изменятся на 0, а далее и значения на входах второго T триггера изменятся на 0, после чего значения на выходе становятся 1101 и.т.д.

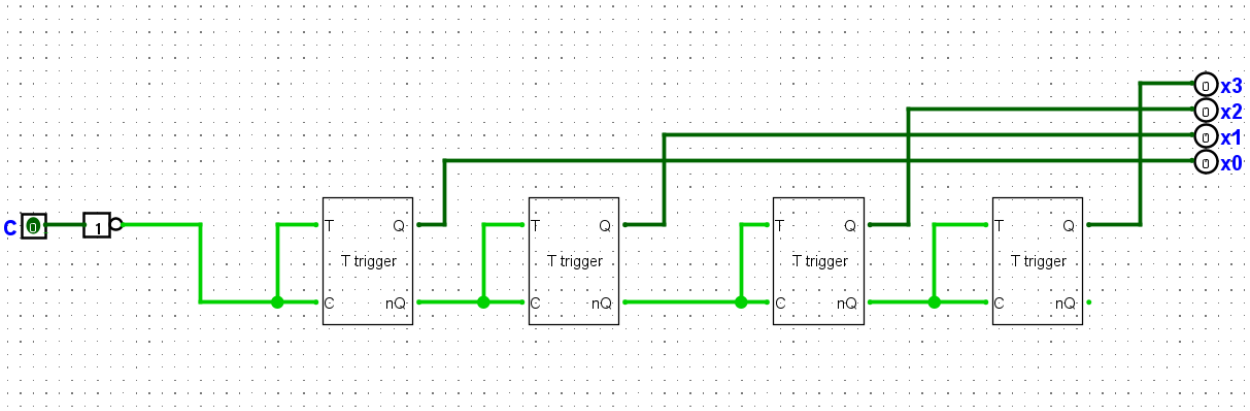
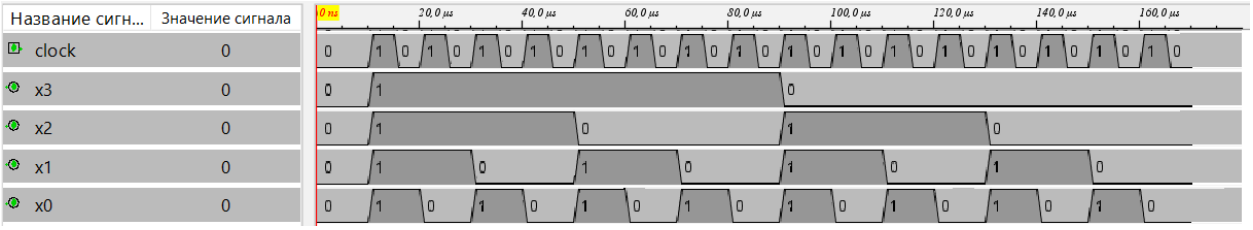


Рисунок 4 - Асинхронный вычитающий счетчик

Временная диаграмма асинхронного вычитающего счетчика:



Осталось добавить модуль к моему счетчику. Для этого нужно, чтобы после 0 счетчик переходил к 12 (1100). Чтобы обрабатывать ноль на выходах, я поставил ИЛИ от 4 элементов (Рисунок 5). Далее для удобства я буду называть этот элемент orAns.

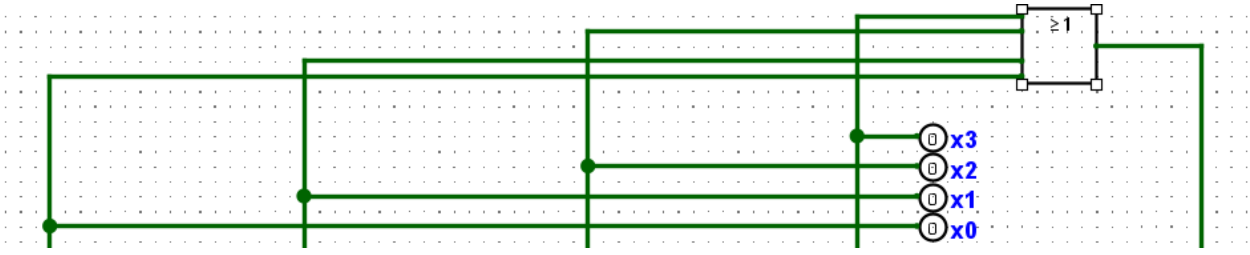
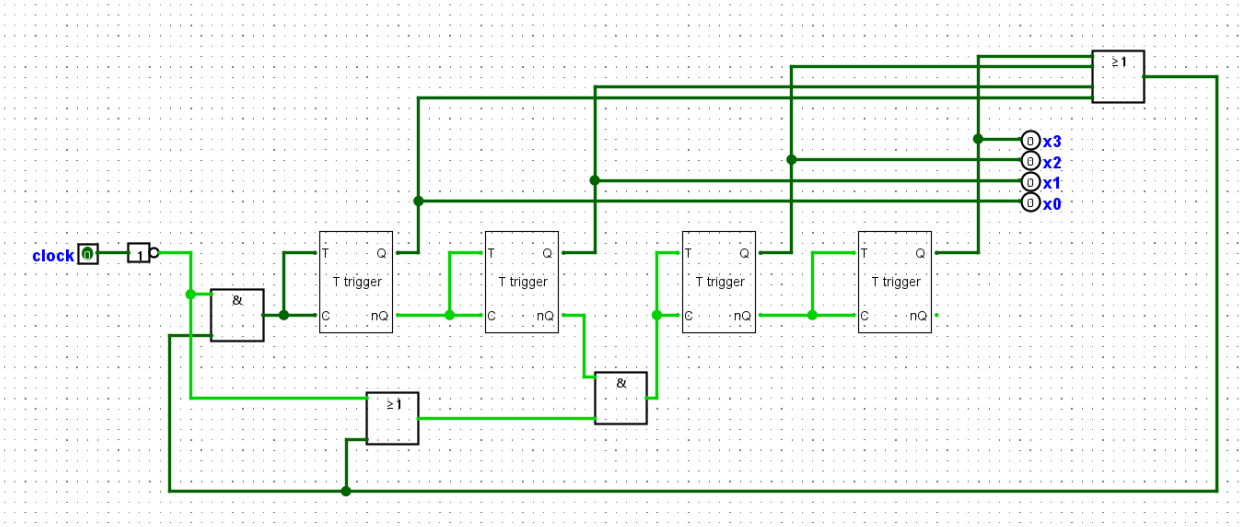


Рисунок 5 – Или от выходов

После того, как я научился обрабатывать 0 на выходах, я построил такую схему:



Теперь при всех нулях на выходах первые два T триггера не будут задействованы, а вторые будут. Заметим, что, чтобы задействовать 3 и 4 триггеры, нужно взять или от входа и orAns, иначе в них бы всегда подавался 0. Аналогично, чтобы в первые 2 триггера не подалась 1 нужно было взять и от входа и orAns. Однако при такой реализации мой счетчик переходил из 1 сразу к 12. Это получалось из-за того, что срабатывали 3 и 4 триггеры. Рисунки 7 – 11 детально показывают этот момент.

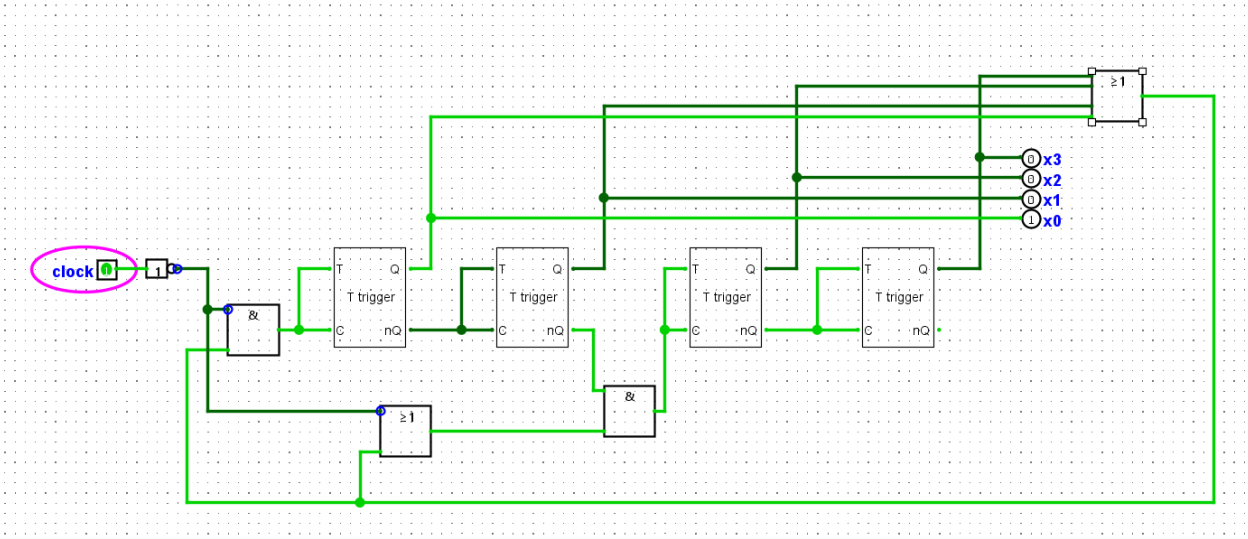


Рисунок 7

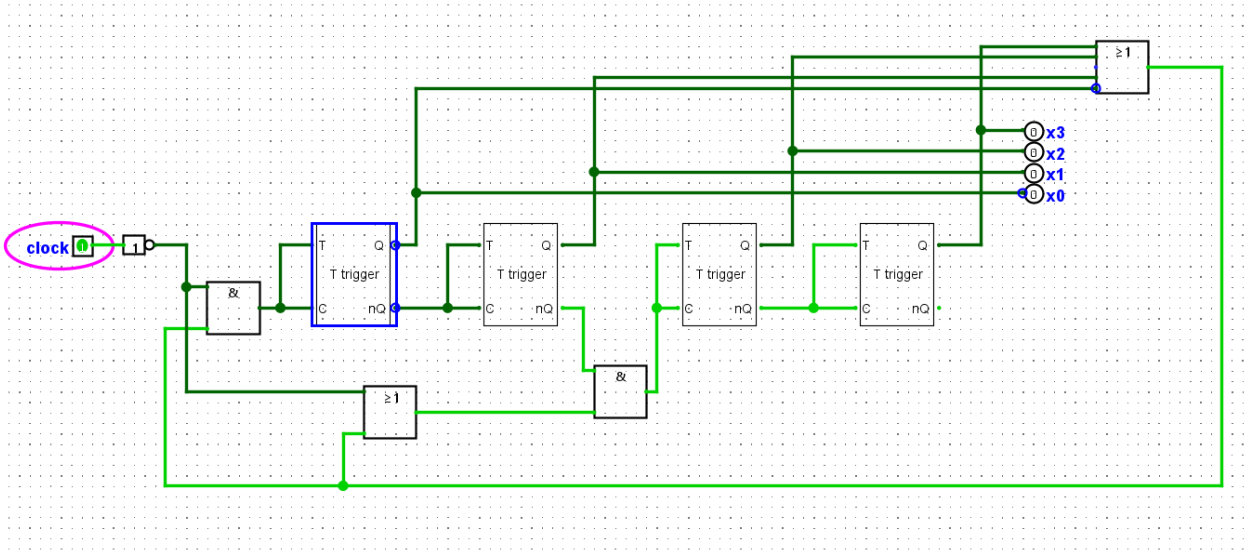


Рисунок 8

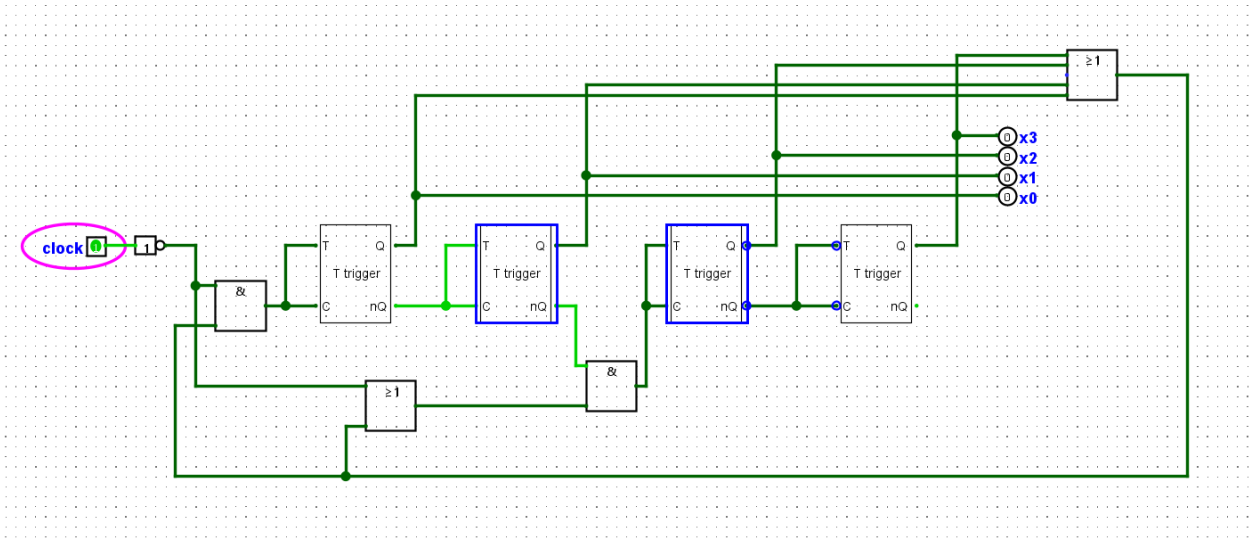


Рисунок 9

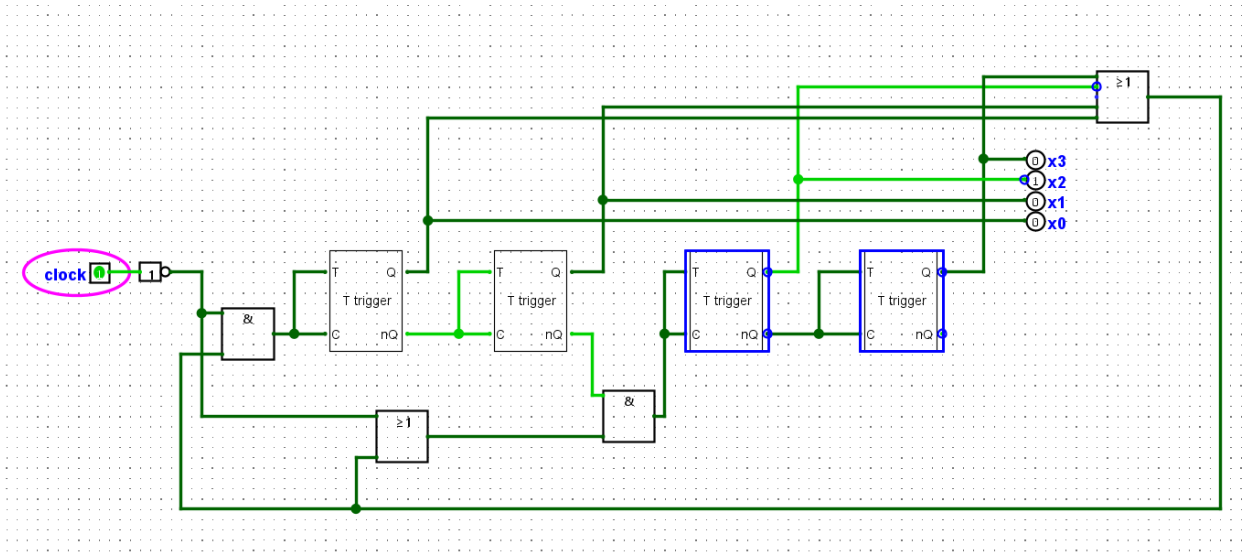


Рисунок 10

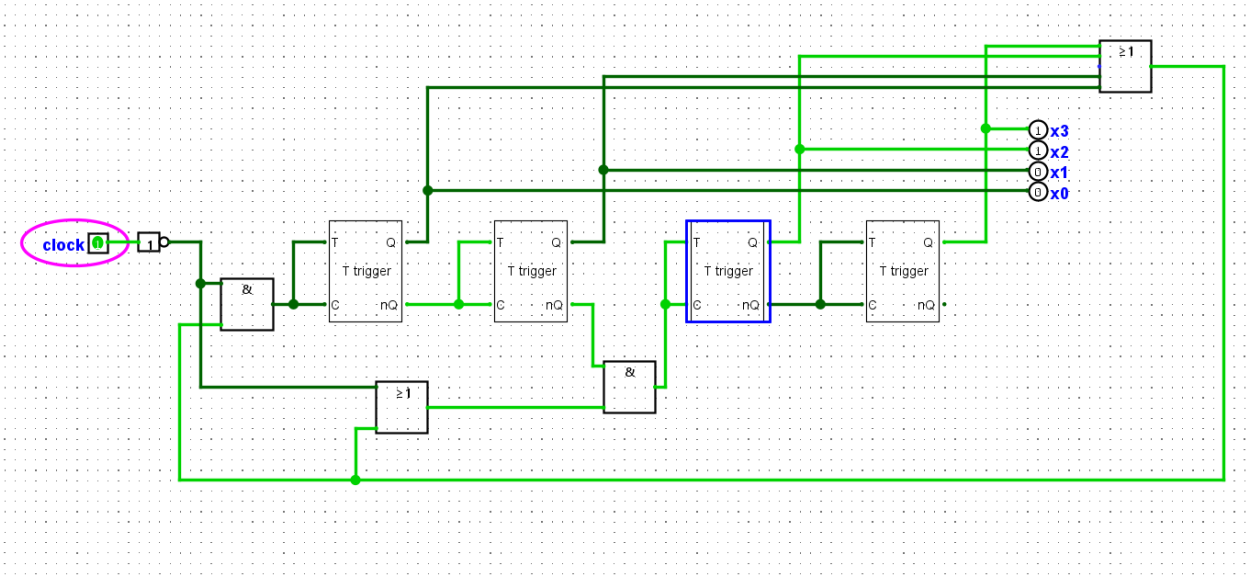


Рисунок 11

Для того чтобы исправить данную проблему, я построил **итоговую** схему (Рисунок 12). В проекте она имеет название «main».

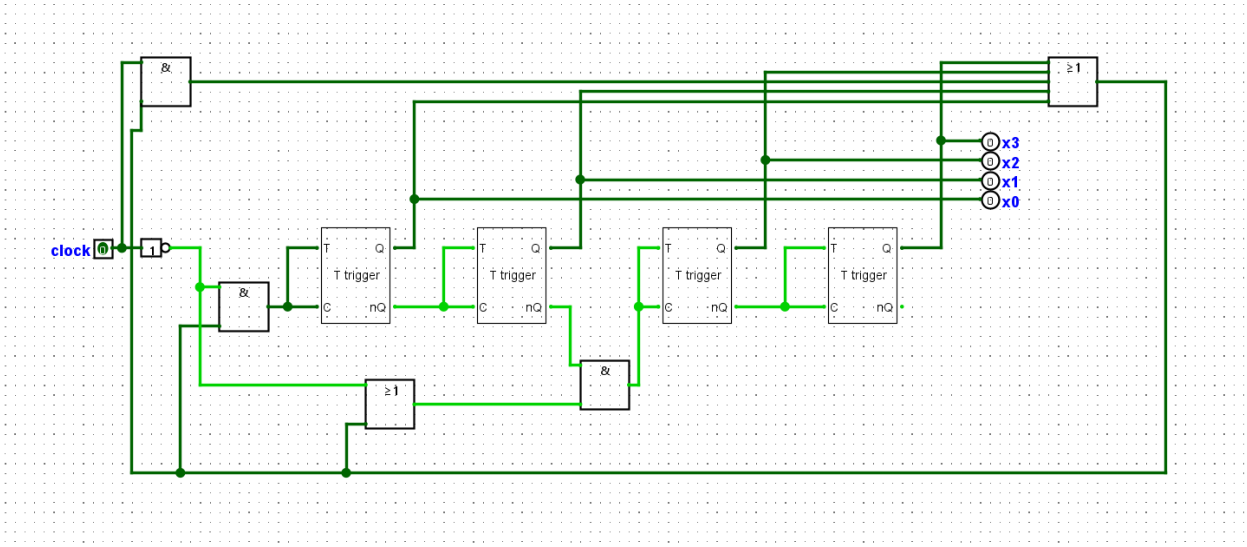
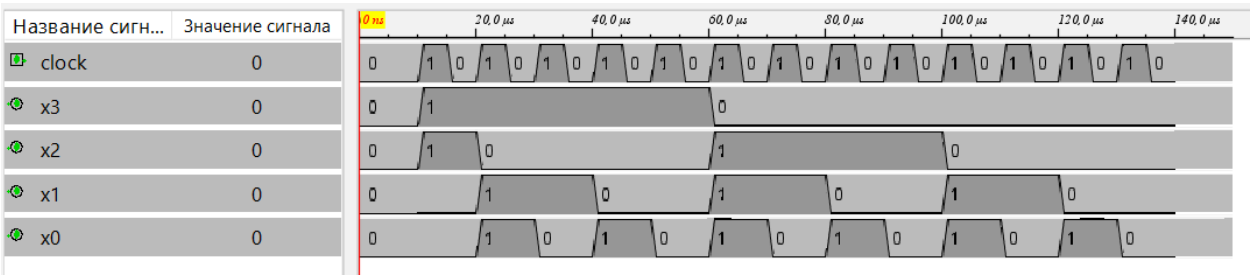


Рисунок 12

Теперь я дополнительно подаю в orAns И от предыдущего значения orAns и значения на входе. При переходе выходного значения от 1 к 0, на 3 и 4 триггеры будет постоянно подаваться 1, благодаря чему они не изменят своего значения (что нам и нужно). Когда же выходное значение будет переходить из 0 в 12, в 3 и 4 триггеры подастся 0, из-за чего они изменят свое значение.

Временная диаграмма итоговой схемы:



Регистр сдвига с линейной обратной связью

Принцип работы:

Регистр сдвига с линейной обратной связью при очередном запросе сдвигает на 1 вправо некоторый набор битов, а в начало ставит бит, равный линейной булевой функции от значений остальных битов (в моем случае 7, 1, 0). Тип конфигурации (в моем случае Фибоначчи) задает линейную булевую функцию от битов.

Первым делом, на основе JK триггера я собрал новый триггер (Рисунок 13). В проекте он имеет название «modJK». В нем вход C отвчает за синхронизацию, вход T записывает значение 1 или 0 в триггер, а вход S позволяет задать значение триггера при необходимсти. Данный триггер срабатывает при переходе C из 1 в 0.

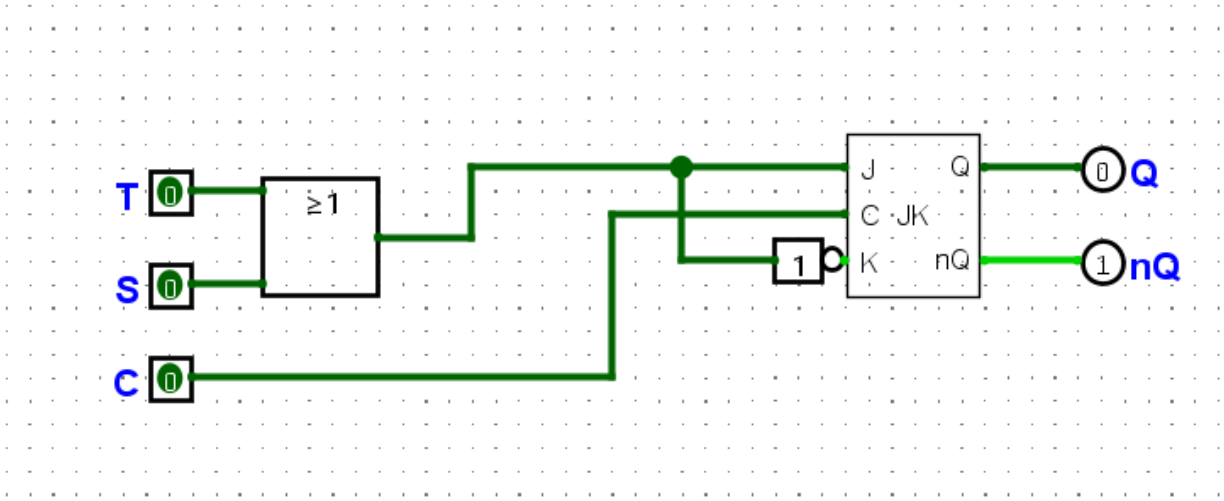


Рисунок 13 – mJK

Использу mJK триггеры, я создал **итоговую** схему (Рисунок 14). В проекте она называется «main».

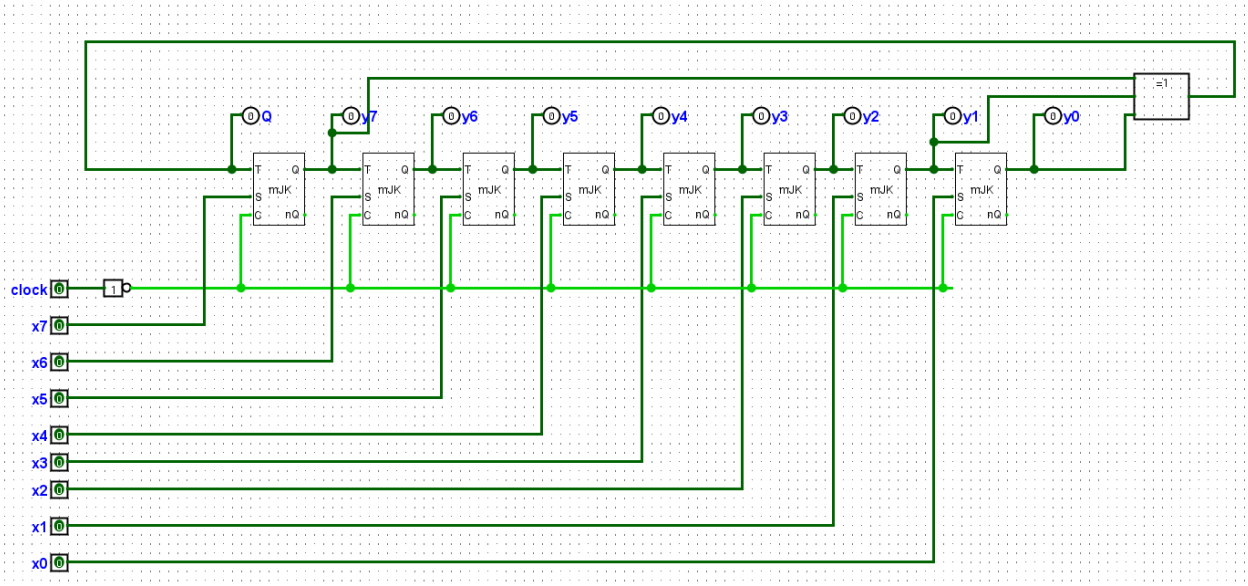


Рисунок 14 – Итоговая схема

Для начала разберемся, какие входы и выходы за что отвечают.

1) Входы x7 – x0 позволяют задать исходное значение для регистра.
2) Выходы y7 – y0 показывают, какие биты записаны в каждом из наших триггеров.
3) Выход Q - очередной бит, сгенерированный нашим регистром, который в результате «пойдет дальше».
4) Вход clock - при значении 1 запускает регистр, при 0 выключает.

Идея схемы: хранить биты регистра в триггерах, а при очередном сдвиге присваивать каждому триггеру значение предыдущего. Заметим, что, чтобы приостановить работу триггеров, достаточно подать в C – 1 (что и делает clock). Следуя из конфигурации, я взял xor из 7, 1, 0 битов и подал его на вход к самому первом триггеру (выход Q).Т.к все битики регистра равняются нулю, нужно задать начальное значение для них. Для этого достаточно подать входы x7 - x0 в S каждого из триггеров.

Инструкция для задания начального значения:

1) Подать нужные 1 и 0 на входы x7 – x0 (Рисунок 15)
2) Перевести clock из 0 в 1 (Рисунок 16). Тогда все значения x7 – x0 «протолкнутся» в триггеры.
3) Перевести входы clock, x7 – x0 в 0 (Рисунок 17)

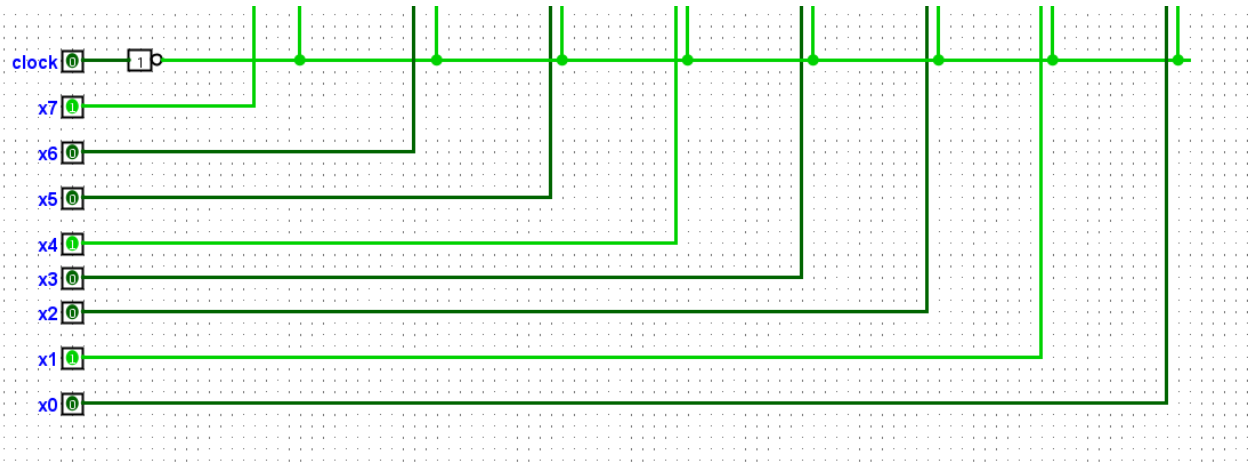


Рисунок 15

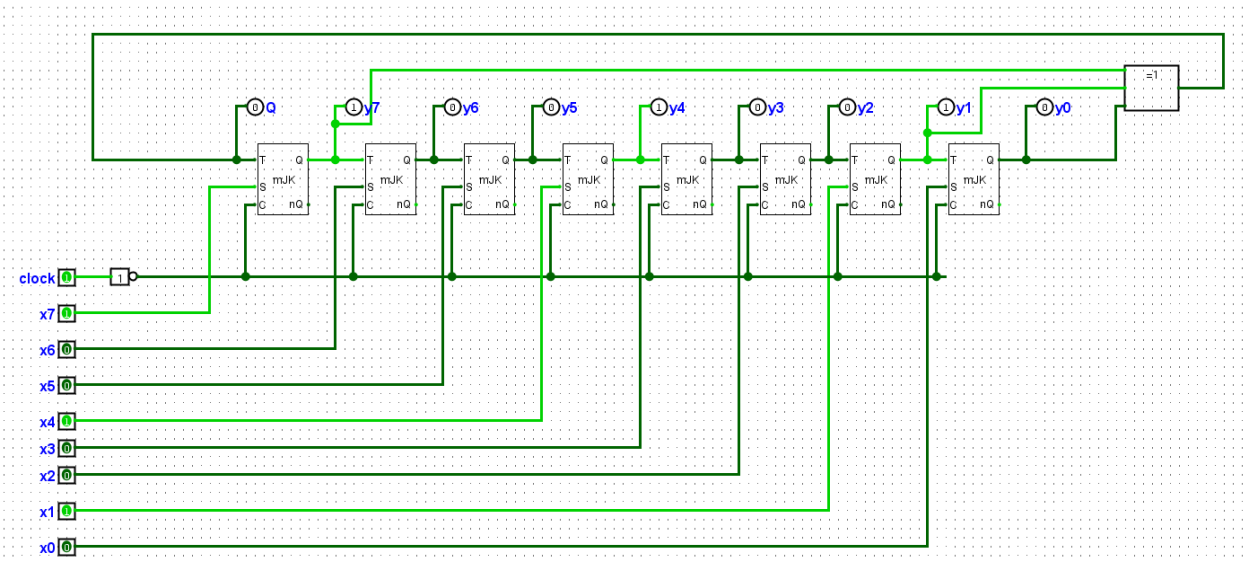


Рисунок 16

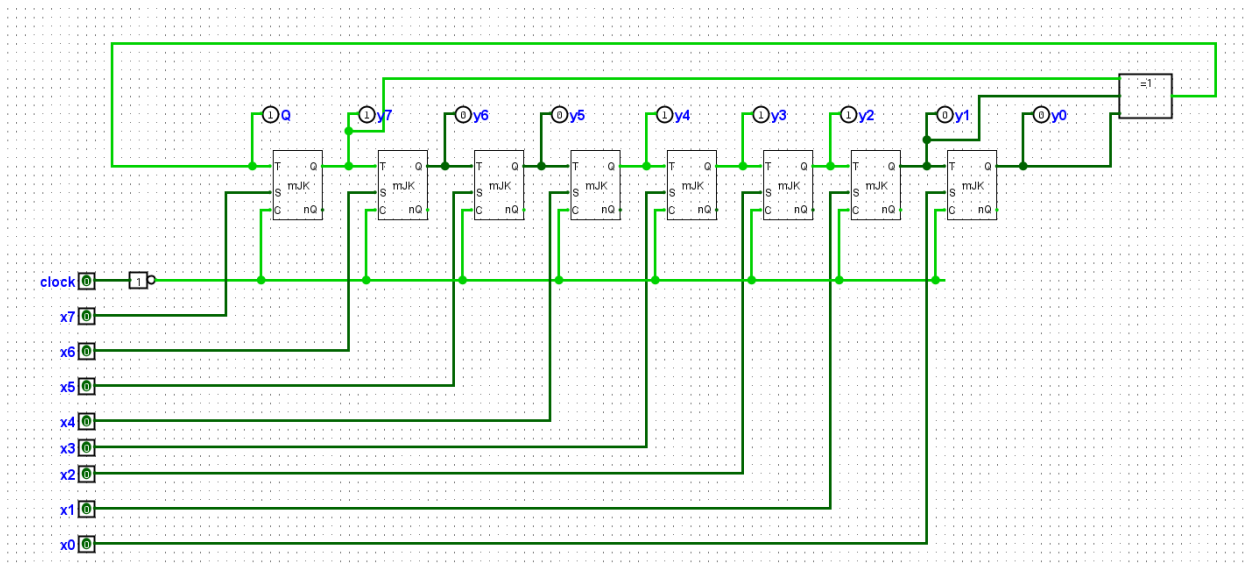


Рисунок 17