Лабораторная работа №1	M3100	2022
Построение логических схем в среде моделирования	Ерёмин Владимир Ильич	

Цель работы: моделирование логических схем на элементах с памятью.

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

Описание: составить и описать принцип работы двух схем: счетчика и регистра сдвига с линейной обратной связью.

Вариант: 1. Асинхронный суммирующий счетчик. ***(модуль:** 13)* и Регистр сдвига с линейной обратной связью ***(конфигурация:** Фибоначчи (10, 7, 0))*.

1. Асинхронный суммирующий счетчик. (модуль 13)

Проект: counter.circ

1.1. Работа счетчика.

Модуль 13 однозначно говорит о том, что для хранения числа нам потребуется 4 бита.

$$\left(2^3 = 8\right) < 13 < \left(2^4 = 16\right)$$

Рассмотрим пример работы:

0000 0001 0010 0011 0100

Чтобы спроецировать это на схему, нужно было посмотреть на примеры с другой стороны: конкретный бит *инвертируется*, если значение предыдущего бита сменилось с единицы на ноль.

Инвертирование - ключевое слово: исходя из него, мы будем выбирать нужный нам триггер в следующем подпункте.

1.2. Конструирование.

Основа всех дальнейших триггеров - “RS Триггер”.

(“RSTrigger” в проекте)

(POR - гарантирует нормальную работу всех последующих триггеров)

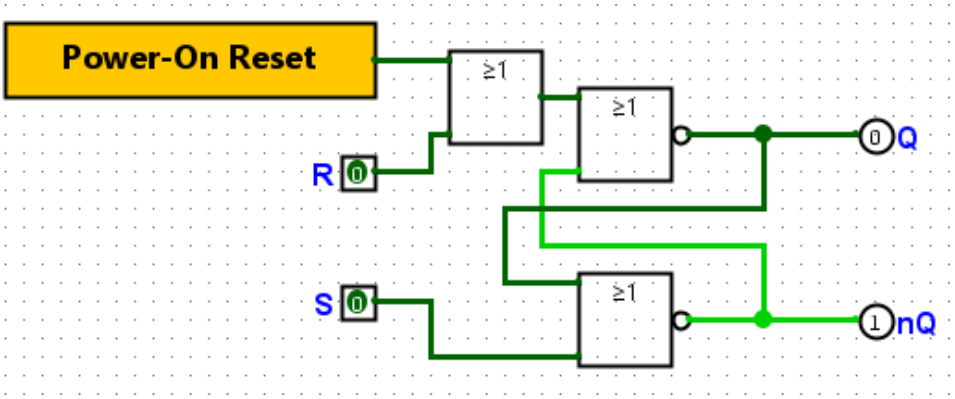

Рисунок 1 - Схема RS триггера.

R	**S**	**Q**	**nQ**
0	0	HOLD	
0	1	1	0
1	0	0	1
1	1	FORBIDDEN	

Таблица 1 - Таблица истинности RS триггера.

В следующих триггерах к уже известным входам добавляется *C (бит синхронизации)*. Когда в состоянии *(1)* - работа происходит в штатном режиме. Когда в состоянии *(0)* - *(в случае с RS триггером) R* и *S* примут значение *(0)* результат сохранится *(исходя из таблицы 1)*.

Также при дальнейшем конструировании на этапе обнуления регистра *(при модуле)* возникнет проблема. Изначально было предположение обнулять триггеры на уровне самого счетчика, но изменение состояния может происходить только при изменении значения бита синхронизации. Обнуление синхронного триггера должно происходить на более низком уровне, а именно на уровне SRS Триггера.

“Синхронный RS Триггер” - Synchronous RS Reset Trigger.

(“SRSRTrigger” в проекте)

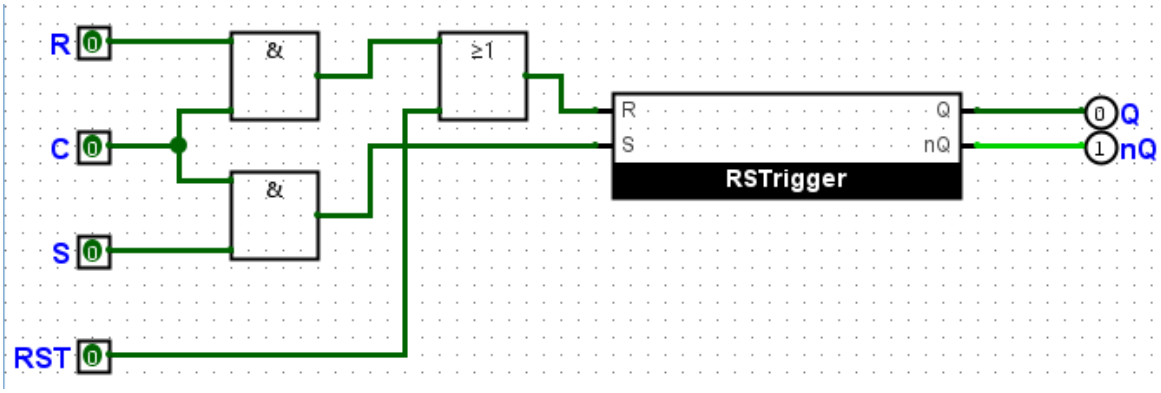

Рисунок 2 - Схема SRSR Триггера.

C	**R**	**S**	**Q**	**nQ**
0	x	y	HOLD	
1	0	0		
1	0	1	1	0
1	1	0	0	1
1	1	1	FORBIDDEN	

Таблица 2 - Таблица истинности SRSR Триггера.

В дополнение к *таблице 2 RST* принудительно ставит *RS Триггер* в состояние *(Q) = 1*, независимо от остальных входов. На безе этого триггера можно построить триггер, который имеет возможность *инвертировать* свое состояние. Тут-то и всплывает слово “инвертирование” из *пункта 1.1.*

“JK Триггер” - JK Reset Trigger

(“JKRTrigger” в проекте)

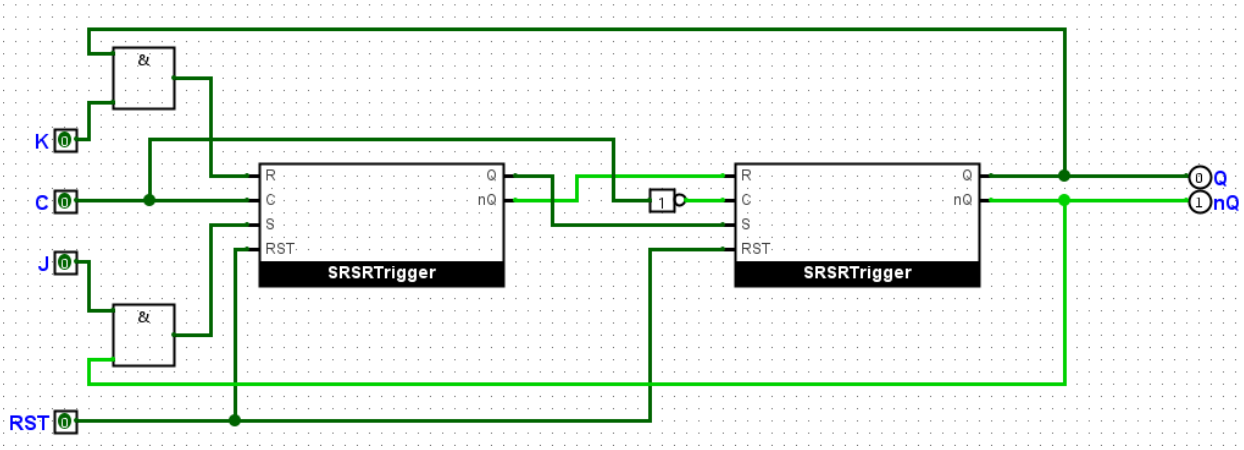

Рисунок 3 – Схема JKR Триггера.

K	**J**	**Q**	**nQ**
0	0	SAVE	
0	1	1	0
1	0	0	1
1	1	TOGGLE	

Таблица 3 - Таблица истинности JKR Триггера.

Смысл входов *RST* и *C* исходит из описания предыдущего триггера.

Данный триггер теперь разрешает состояние *(1, 1)* - Инверсия. Однако теперь триггер совершает работу с отставанием на полтакта. Используем эту особенность во благо нашей идеи: вспомним *(пункт 1.1*), что триггер должен будет инвертироваться, если предыдущий поменял значение с *(1)* на *(0)*. Приведенный *(рисунок 3)* выше триггер, аналогично, меняет свое состояние, когда *C совершает переход.* с *(1)* на *(0)*.

Делаем вывод, что выход *Q* может играть роль *C* для следующего триггера. А на входы *J* и *K* можно константно подавать *(1)* так, как нам имеет значение только инверсия триггера *(ее выполнение будет зависеть от C)*.

“TR Триггер” - Toggle Reset Trigger

(“TRTrigger” в проекте)

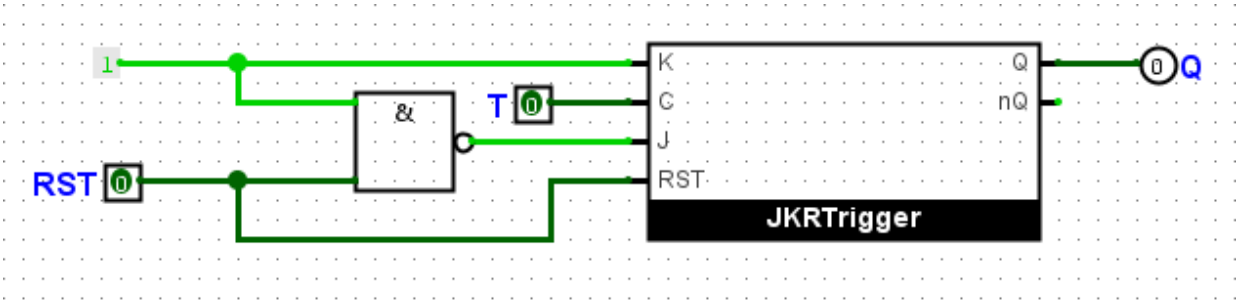

Рисунок 4 - Схема TR Триггера.

T	**Q**
0	HOLD
1	TOGGLE

Таблица 4 - Таблица истинности TR Триггера.

Небольшая модификация *RST* через элемент *NAND*. Она гарантирует правильную работу *RST* - подавая *(1, 0)*, мы избегаем запрещенной для внутренних *RSR Триггеров* ситуации *(1, 0)*.

1.3. Сборка.

Имея на руках, готовые компоненты собрать схему не составляет труда.

“Counter” (по модулю 13)

(“main” в проекте)

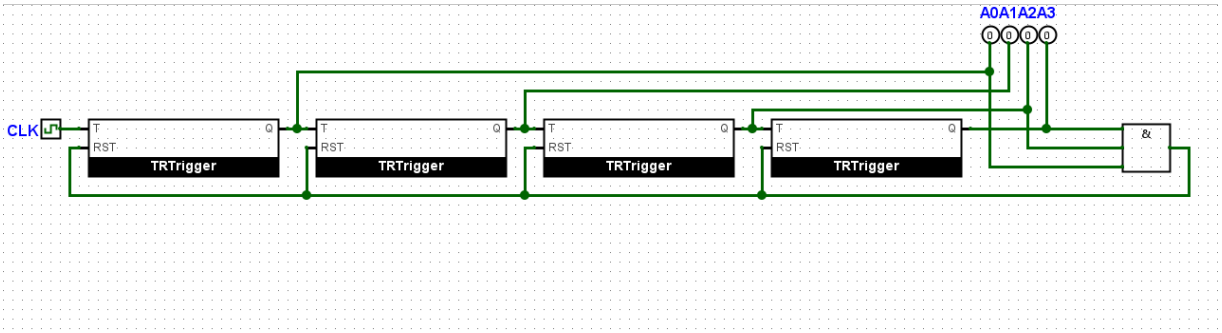

Рисунок 5 - Схема Счетчика.

Модуль реализуется просто: по достижении 13 (1101) - мы сбрасываем все соответствующие триггеры *(для наглядности схемы - сбрасываются все)*. Проверка же осуществляется через оператор *AND:* если *(A0, A2, A3)* приняли значение 1, то идет сигнал на обнуление.

В качестве примера работы привожу временную диаграмму *(рисунок 6)*.

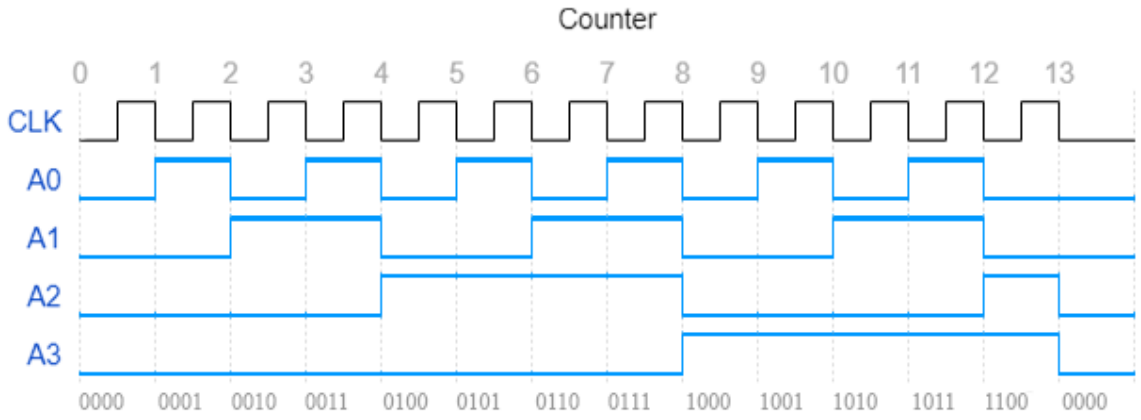

Рисунок 6 - Временная диаграмма для счетчика.

2. Регистр сдвига с линейной обратной связью (конфигурация: Фибоначчи (10, 7, 0))

Проект: lfsr.circ

2.1. Работа регистра сдвига.

Рассмотрим такой регистр на примере двоичного числа.

0001 0010 0100 1000

0101 1010

Исходя из этих примеров очевидно, что операция синхронная так, как все биты должны сдвинуться одновременно. А для определения значения можно опять *(пункт 1.1.)* посмотреть на примеры с другой стороны: если стоящий за конкретным битом бит равен единице, то при сдвиге конкретный бит станет равный единице. Аналогично с нулем.

2.2. Обратная связь.

Обратная связь подразумевает цикличность нашего регистра, а именно:

0100 1000 0001 0010

Очевидно, что нужна инициализация, ибо сдвиги с нулевым регистром смысла не имеют. А 0000 - становится недопустимым состояние.

Конфигурация описывает процедуру обратной связи. В моем случае — это, сумма заданных в конфигурации *(10, 7, 0)* битов по модулю 2. Это в свою очередь подразумевает использования оператора *XOR*.

Исходя из конфигурации нам потребуется ровно 11 битов *(отсчет с 0)*.

2.3. Конструирование.

За основу возьмем уже готовые триггеры (рисунок 1-3). RST можно опустить так, как в данной схеме он не понадобится.

“D Триггер”

(“DTrigger” в проекте)

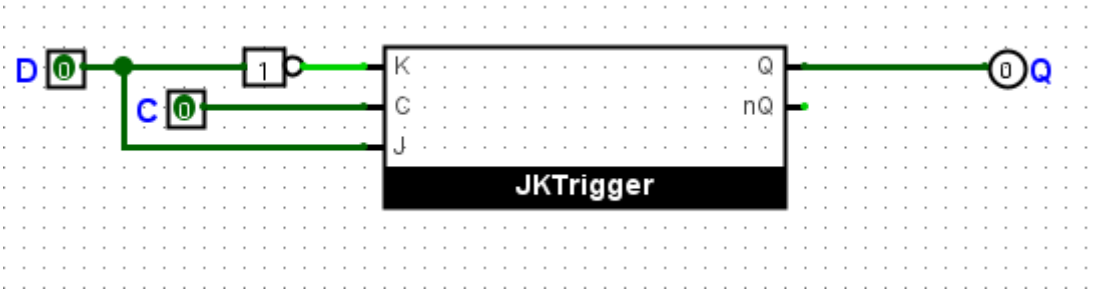

Рисунок 7 - Схема D Триггера.

C	**D**	**Q**
0	x	HOLD
1	0	0
1	1	1

Таблица 5 - Таблица истинности D триггера.

Синхронизация у всех триггеров едина, а значение самого напрямую исходит из значения, идущего до него *(в нашем случае: они цикличны)*.

“LFSR” - Linear Feedback Shift Register.

(“main” в проекте)

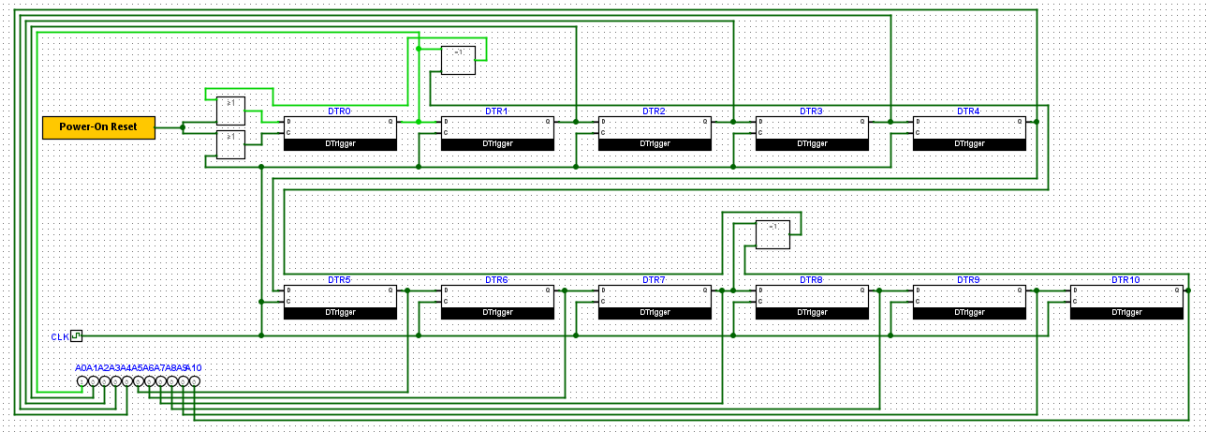

Рисунок 8 - Схема LFSR.

Триггеры и выходы пронумерованы соответственно. Обратная связь идет от *DTR10* до входа *DTR0,* по пути считая сумму со значениями триггеров, заданных в конфигурации *(10, 7, 0)* или *(DTR 10, DTR7, DTR0)*, по модулю 2 то, есть *XOR*.

POR - инициализирует регистр *(пункт 2.2.)*.

Собственно, как выяснилось - эта схема может использоваться, как генератор псевдослучайных чисел, где конфигурация задает SEED.

3. Источники

3.1. Счетчик.

1. https://www.allaboutcircuits.com/textbook/digital/chpt-11/asynchronous-counters/
2. https://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/16-flipflops/40-jkff/jkff.html

3.3. Регистр сдвига с линейной обратной связью.

1. https://en.wikipedia.org/wiki/Linear-feedback_shift_register#Fibonacci_LFSRs