Лабораторная работа №1	M3136	2022
Построение логических схем в среде моделирования	Багринцев Михаил Алексеевич	

Цель работы: моделирование логических схем на элементах с памятью.

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

Описание: построить счётчик и регистр сдвига с линейной обратной связью.

Вариант:

1.) Схема счётчика: 3, модуль счёта: 23.

2.) Фибоначчи (8, 4, 3, 2, 0)

Счётчик

Моей первой задачей было составить синхронный суммирующий счётчик. Разница синхронного и асинхронного счётчика состоит в том, что первый должен переключать все разряды одновременно (синхронно) по единому входному счётному сигналу. Так как синхронный счётчик должен переключать все триггеры (о них дальше) одновременно, триггер, который будет переключен, должен знать о состояниях всех триггеров до него. Эту часть мы реализовали. Также в счётчике необходимо использовать двухтактные триггеры, потому что иначе при синхронизации 1 триггеры могут делать лишние переключения. Например, первый триггер,

обозначающий наименьший бит в записи инкрементирующегося числа, хранил 0, но после итерации счётчика начал хранить 1. Тогда, что должен делать второй триггер? Он просто увеличит значение своего бита на 1 по модулю 2. То есть нам нужно разделять этапы хранения и смены значения триггера. Это и идея двухтактных триггеров. Я не стал изобретать велосипед и использовал JK-триггер.

Триггеры

JK-триггер построен на синхронном RS-триггере, поэтому начнём со второго (схема rsTriggerWithSync).

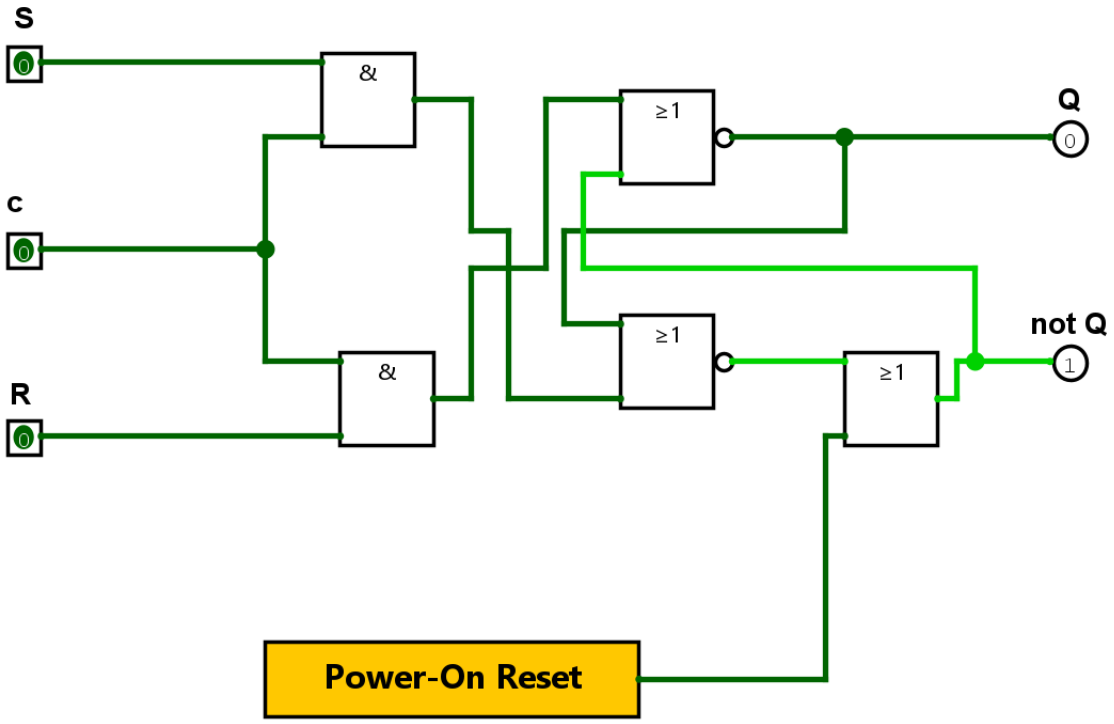

Рисунок 1 – Синхронный RS-триггер в среде моделирования Logisim evolution

Из рисунка 1 видно, что при c (параметр синхронизации) равном нулю схема выключена и хранит последнее значение в Q.

Ниже, для лучшего понимания принципов устройства, приведена таблица истинности работы синхронного RS-триггера. (Пусть предыдущее значение Q было Q’)

S	0	0	1	1	0	0	1	1
R	0	1	0	1	0	1	0	1
c	0	0	0	0	1	1	1	1
Q	Q’	Q’	Q’	Q’	Q’	0	1	-
not Q	not Q’	not Q’	not Q’	not Q’	not Q’	1	0	-

Таблица 1 – Таблица истинности синхронного RS-триггера (- означает неопределённое поведение)

Если у RS-триггера нет входа c, то он называется асинхронным. Для RS-триггера асинхронность не страшна, но она играет большую роль, когда мы хотим инвертировать значение. Если без синхронизации (имеется в виду, что никакая часть схемы не использует идею синхронизации) подавать 1 на контакт несколько тактов подряд, то инверсия произойдет столько раз, сколько тактов мы подали 1. Это плохо контролируемое изменение значения, поэтому мы хотим его избежать. Синхронизация является естественным способом это сделать.

Теперь о работе JK-триггера(схема jkTriggerWithReset).

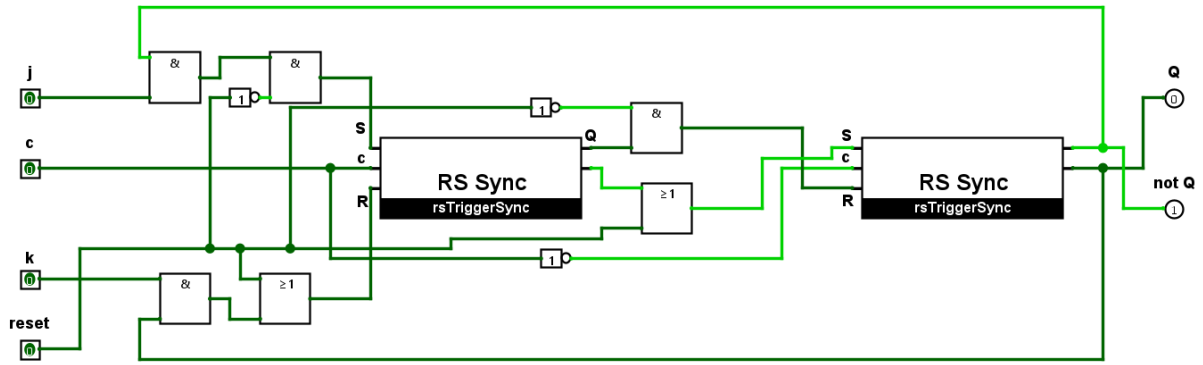

Рисунок 2 - JK-триггер с функцией сброса

В JK-триггере мы видим реализацию идеи двухтактного триггера. Он основан на двух синхронных RS-триггерах. Один хранит предыдущее

значение, другой получает новое. Как видно из рисунка 2, работа JK-триггера, действительно, проходит в 2 такта, значение обновляется по спаду синхронизации. Это связано с тем, что вход синхронизации второго RS-триггера является инверсией входа синхронизации первого. Также значения R и S, которые мы подаём на второй RS-триггер перепутаны. Это сделано для удобного расположения проводов, не допускающих появления (1, 1) на входе RS-триггера (самый нижний и самый верхний провода на рисунке 2). В конкретно этом JK-триггере присутствует модификация - reset контакт. Из названия понятно, что он приводит JK-триггер в нулевое положение (даёт набор (J = 1, K = 0) на вход триггеру и делает нулём его сохранённый бит), убедиться же в этом можно, посмотрев на рисунок 2.
И в завершение, приведём таблицу истинности JK-триггера.

J	x	x	1	1	1	1	0	0	0	0
K	x	x	0	0	1	1	0	0	1	1
c	0	0	1	1	1	1	1	1	1	1
Q(t)	0	1	0	1	0	1	0	1	0	1
Q(t+1)	0	1	1	1	1	0	0	1	0	0

Таблица 2 – Таблица истинности JK-триггера (Q(t) - значение выходного Q на t-ом такте, x - произвольное значение)

Счётчик

Для начала напомню, что в моём варианте нужно сделать синхронный суммирующий счётчик с модулем счёта 23.

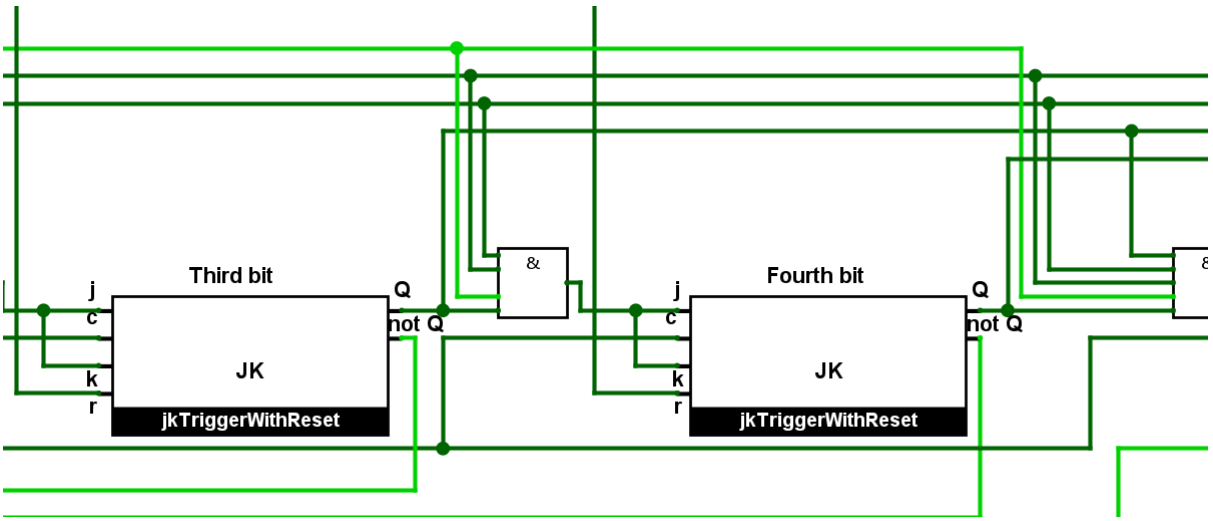

Рисунок 3 - JK-триггеры в счётчике

На рисунке 3 показано как располагаются JK-триггеры в счётчике. Мы осознано обрезали функциональность JK-триггера соединив провода ведущие в J и K входы. Таким образом, у нас есть только 2 метода - сохранить предыдущее значение или инвертировать его. Каждый JK-триггер представляет собой бит выходного числа. Так как модуль счёта 23, всего используется 5 JK-триггеров. Идея работы счётчика такая: при нуле синхронизации каждый триггер с помощью блоков &, которым на вход поданы Q всех триггеров до него, может ответить на вопрос: “Все ли триггеры до меня сейчас хранят единицу?” Если ответ да, то при синхронизации 1, этот триггер получит входное значение (J = 1, K = 1), которое сделает инверсию хранимого в Q значения. Если ответ нет, то этот триггер будет сохранять своё значение, потому что есть триггеры со значением ноль до него, следовательно, сначала они должны стать единичками. Первому триггеру на J=K вход подаётся константная единица. Получается, что триггер инвертирует своё значение только в том случае, если перед ним стоят одни единички. Такой подход обеспечивает переключение всех триггеров по одному такту, что и характеризует синхронный счётчик. Также на рисунке 3 можно увидеть, что все триггеры имеют вход reset, провода которые к ним идут начинаются на рисунке 4.

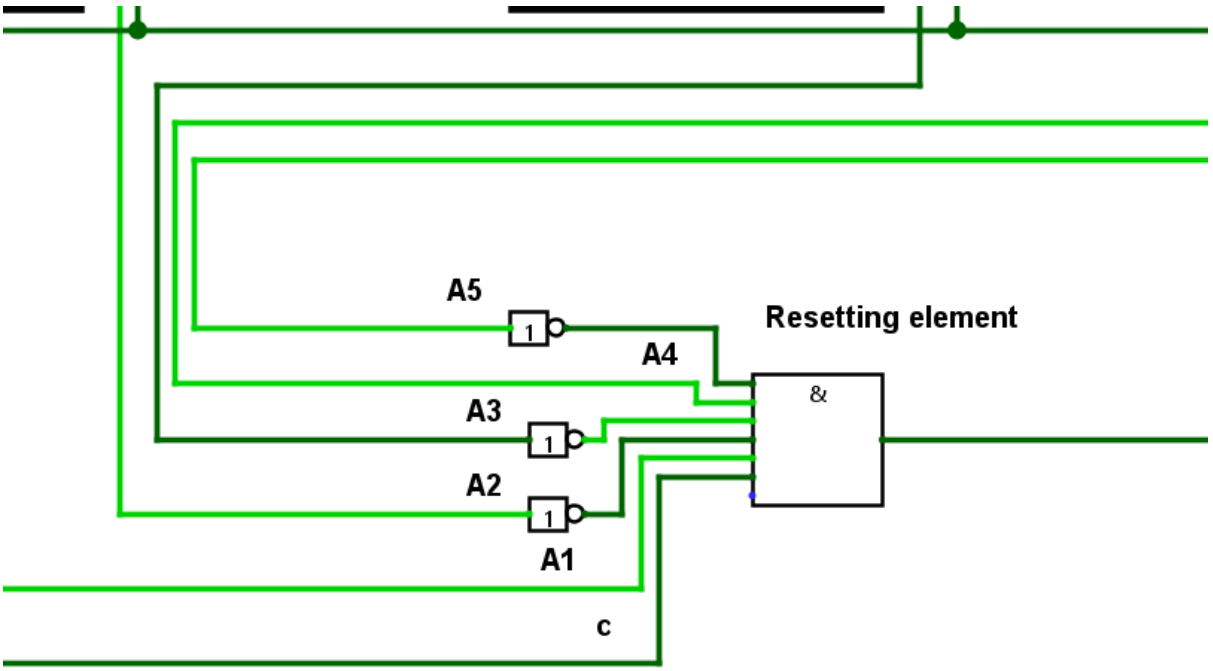

Рисунок 4 - сбрасывающий элемент счётчика

Провода, проведённые к входам reset, подают единицу только когда наши триггеры кодируют значение 10110 (посмотрите на рисунок 3 - в resetting element подаются not Q выходы триггеров), что соответствует числу 22 в десятичной системе счисления. То есть мы обнуляем все триггеры в момент записи новых значений при увеличении значения счётчика с 22 до 23 - логика очевидна.

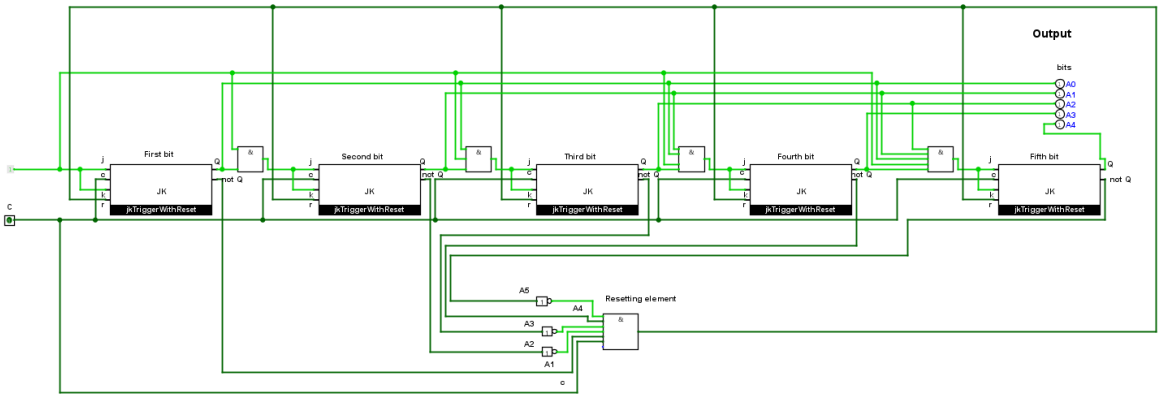

Рисунок 5 - синхронный суммирующий счётчик(схема main)

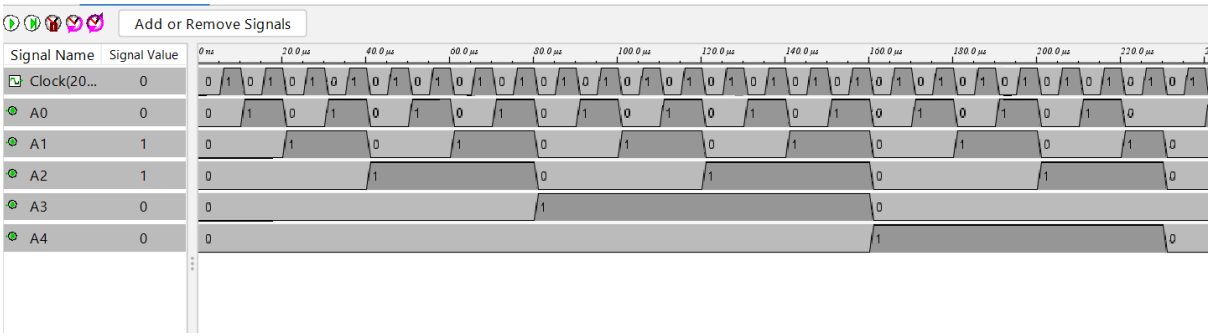

Рисунок 6 - временная диаграмма работы счётчика

Из рисунка 6 видно, что счётчик действительно работает по спаду синхронизации, что вызвано использованием двухтактных JK-триггеров. Выход A4 принимает значение 0 чаще чем значение 1. Это связано с тем, что счётчик не доходит до 31, а сбрасывается после 22.

Регистр сдвига с линейной обратной связью

Триггеры

Для хранения одного бита я использовал синхронный D-триггер. Он основан на уже нам знакомом из предыдущей схемы JK-триггере.

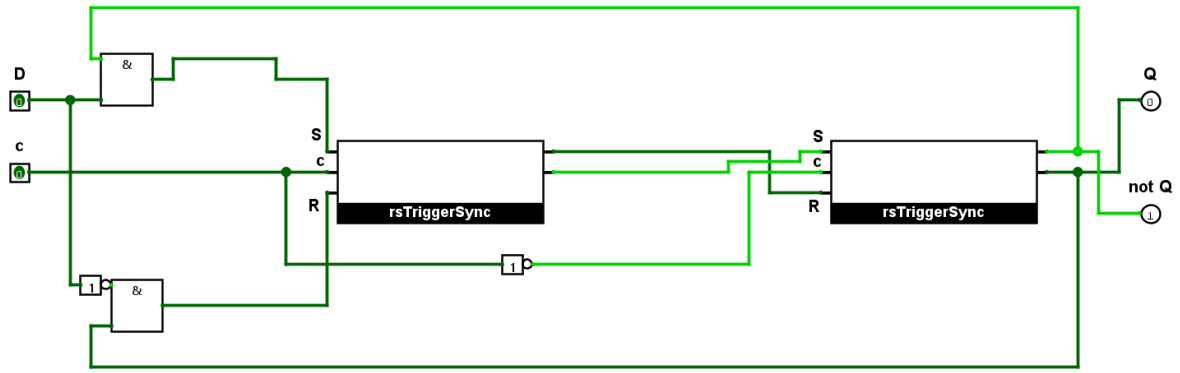

Рисунок 7 - синхронный D-триггер (схема dTriggerSync)

Отличие лишь в том, что мы подаём сигнал D на вход J, а на вход K инверсию D. В таком случае мы не используем инверсию (J=1, K=1) и сохранение предыдущего значения (J=0, K=0) - каждый раз устанавливаем новое состояние триггера. Почему так? В регистре сдвига с линейной обратной связью нам нужно, как понятно из названия, лишь сдвигать биты вправо - то есть посмотреть какое значение было у предыдущего триггера и на новом такте перенять это значение.

Регистр

Напомню, что в моём варианте необходимо было сделать регистр типа конфигурации Фибоначчи с (8, 4, 3, 2, 0) значимыми битами. То есть общий XOR значений D-триггеров с номерами (8, 4, 3, 2) записываем в самый первый (нулевой) D-триггер. Далее предлагаю рассмотреть работу работу регистра, опираясь на картинку 8.

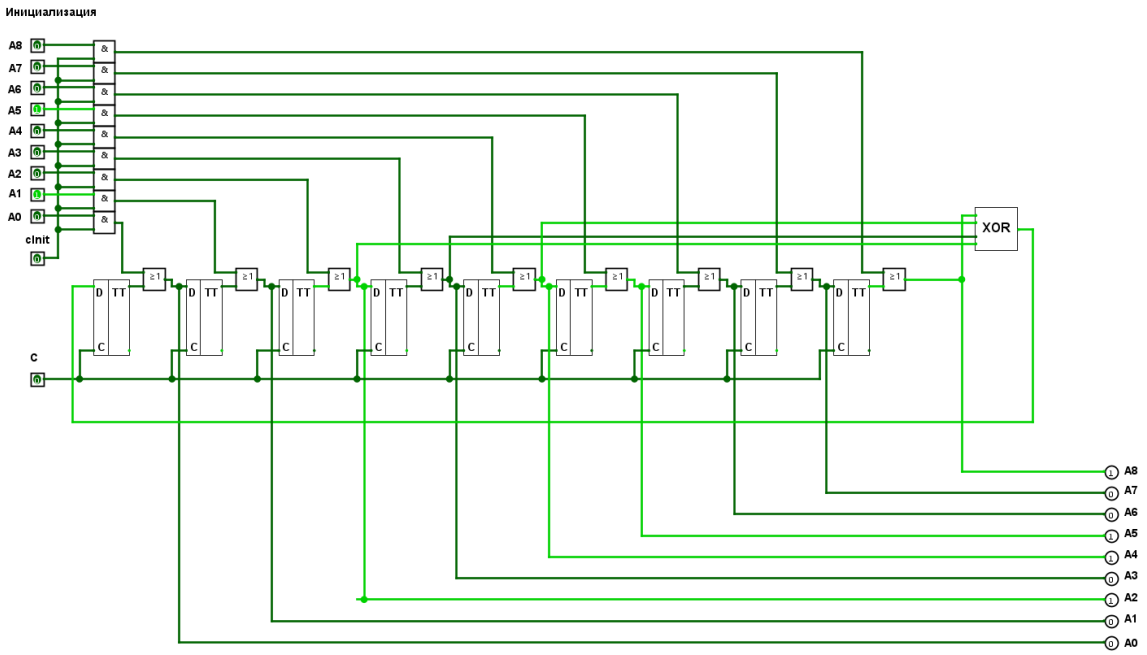

Рисунок 8 - регистр сдвига с линейной обратной связью (схема main)

В моём триггере пользователю предоставляется возможность выбрать начальный набор значений D-триггеров, потому что если запуститься с

набора (0, 0, … , 0) на триггерах, то XOR всегда будет давать 0 и при смене значения на входе C ничего происходить не будет. Пользователю нужно выбрать набор в блоке “Инициализация”, потом включить подачу битов на D-триггеры подачей 1 на контакт cInit. Чтобы D-триггеры запомнили значения, на контакт C, отвечающий за синхронизацию, необходимо подавать 1. Далее для корректной работы регистра сдвига от пользователя требуется перестать подавать 1 на контакт cInit. Теперь вы можете генерировать псевдослучайные последовательности битов.

Из-за использования JK-триггеров, схема работает по спаду синхронизации, то есть новое значение получаете при переключении контакта C с единицы на ноль. При C равном единице происходит запись - каждый D-триггер передаёт своё значение в последующий, за исключением последнего. D-триггеры с индексами 8, 4, 3, 2 участвуют в генерации значения для D-триггера с индексом 0. Мы делаем операцию XOR от их выходных значений при C=1 и передаём результат на D вход D-триггера с индексом 0. Кстати, говоря о XOR для четырёх элементов, стоит сказать, что я использую свой XOR, потому что XOR для 4-х входов “из коробки” при входных значениях (1, 1, 1, 0) даёт результат 0.

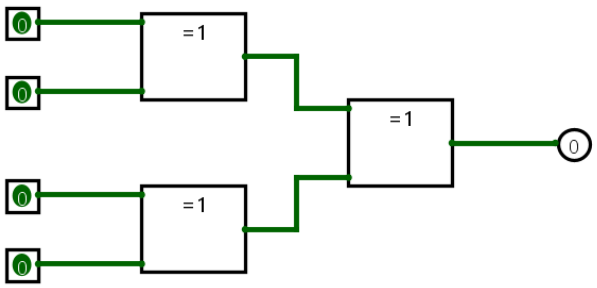

Рисунок 9 - XOR для 4-х входов (схема XORfor4inputs)

На рисунке 9 представлен XOR, который используется в схеме на рисунке 8.