САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Дисциплина: Архитектура компьютера  
  
  
  
  
  
  
  
  
  
  
  
  
Отчет

по лабораторной работе № 1

«**Построение логических схем**»

Выполнил: Скальт Альберт Вячеславович

студ. гр. Б10

Санкт-Петербург

2021

**Цель работы:** моделирование логических схем на элементах с памятью.

**Инструментарий и требования к работе:** работа выполняется в logisim**.**

**Практическая часть.**

**1)** **Построение схемы счетчика.**

**Внимание**. Чтобы счетчик начал работать, нужно его инициализировать, для этого контакт I нужно перевести в 1 и далее в 0. Таким образом проициниализируется триггеры, используемые в схеме.

В начале я изготовил стандартный синхронный по высоте RS триггер с кнопкой инициализации. (Рис. 1)  
Вход I отвечает за инициализацию RS триггера.

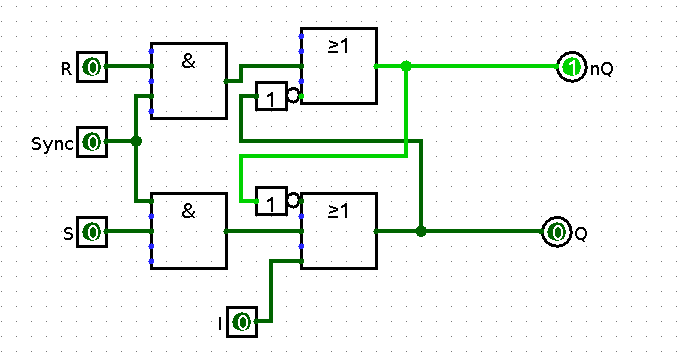


Рис. 1

Далее с помощью двух RS триггеров изготовил синхронный (по заднему фронту) T триггер. (Рис. 2)

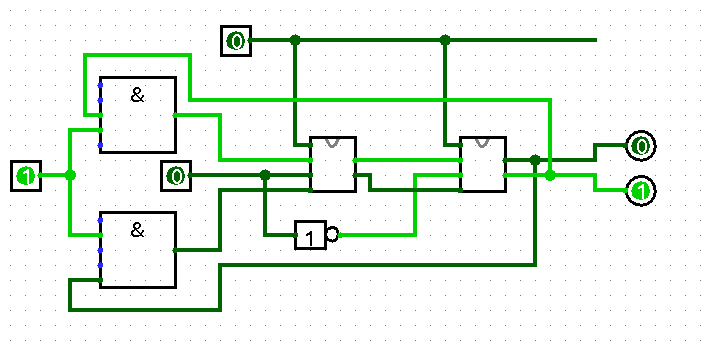


Рис. 2

Такой триггер удобен в схеме счетчика, так как по сути все что надо сделать при прибавлении 1 - это на некотором префиксе числа инверснуть значения битов (пока происходит перенос). Так как модуль у меня равен 15, то для хранения числа достаточно будет 4 битов. Само изображение схемы на рисунке 3.

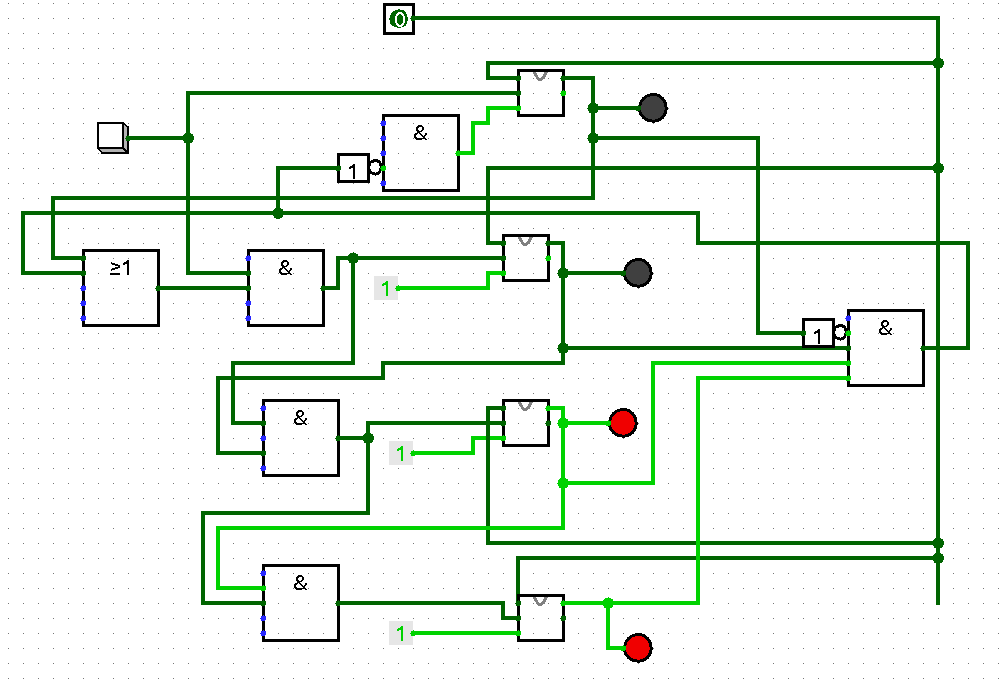


Рис. 3

Данная схема уже использует модуль, но чтобы описать принцип работы такого счетчика, удобно рассматривать его по модулю 16, поэтому, с него и начнем, а далее уже будет объяснено, как был добавлен модуль.

**Работа по модулю 16:**

В такой схеме все работает также, как в схеме на рис. 3, но ко всем триггерам подведена константная единица в Т разъем.

При нажатии на кнопку прибавления синхронизация на триггере, отвечающем за младший разряд, отпускается, происходит вычисление переноса из первого разряда во второй с помощью предыдущего значения младшего бита. Если это значение есть 1, то далее синхронизация отпускается на Т-триггере, отвечающем за второй бит (то есть на нем уже точно будет поменяно значение на противоположное), вычисляется перенос из 2 в 3 разряд, аналогично, если он ненулевой, то отпускается синхронизация на Т-триггере 3 разряда и так далее. В итоге те триггеры, на которых в процессе была опущена синхронизация составляют тот самый префикс числа, на котором значение битов меняется на противоположное. После отжатия кнопки они срабатывают и меняют значение на себе на противоположное.

**Улучшение для работы по модулю 15:**

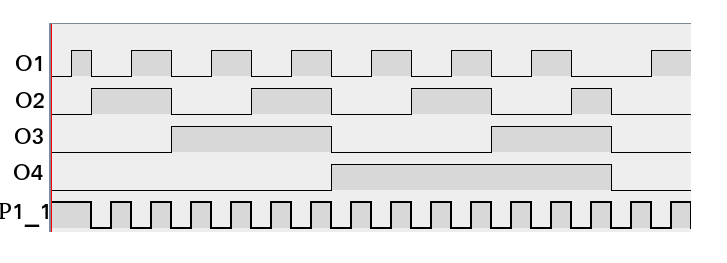
Требовалось по сути обнулять счетчик при попытке добавить к 14 единицу. Я сделал на схеме специальный вывод, который равен 1, если комбинация цифр на выходе сейчас равна 14. Далее модифицировал вычисление переноса с самого младшего на второй разряд. Он будет равен 1, если есть реальный перенос, либо если число равно 14. Тогда все биты, кроме младшего уже точно обнуляются, при прибавлении к 14 единицы. Осталось позаботиться о младшем бите. Для этого я немного поменял вход самого младшего триггера: если число равно 14, то после притока сигнала он больше не меняет значение на противоположное. Таким образом, 14 + 1 = 0 для моего счетчика.  
  
**Временная диаграмма.**Чтобы построить временную диаграмму я добавил вместо кнопки добавления единицы тактовый генератор P1\_1, а также назвал выходы битов O1, O2, O3, O4, O5 для битов, начиная с младшего. Полученная диаграмма на рисунке.  


Диаграмма 1

**2) Построение схемы взятия корня.**

Для того чтобы описать построение этой схемы, мне понадобится математическая лемма. К сожалению, всех нужных мне математических знаков нет в моем текстовом редакторе, поэтому я оформил эту лемму в LaTex. Она записана в файле lemma.pdf.  
  
Исходя из леммы можно вычислять корень числа x по следующему алгоритму:

1. Рекурсивно найти корень для такого же числа, но без двух младших битов.
2. Умножить найденное число на 2.
3. Прибавить к результату 1 и если квадрат числа все еще не будет превышать x, то ответом является полученное число, вернуть его.
4. Вернуть число без прибавленной единицы.

Заметим, что это эквивалентно последовательному вычислению корня в начале из числа, составленного из двух старших битов исходного, далее из числа, составленного из четырех старших битов и так далее, пока не дойдем до 8 бит.

**Построение вспомогательных схем.**

**Возведение в квадрат.**  
Для реализации алгоритма потребуется возведение не более чем 5-битного числа в квадрат. Для этого потребуется сложение не более чем 10-битных чисел. Реализовал я сложение следующим образом, в начале построил вспомогательную подсхему sum\_node (Рис. 4), которая принимает перенос с предыдущего разряда и два текущих бита, вычисляет результат бита суммы и перенос в следующий разряд. Из таких подсхем составил схему для сложения (Рис. 5).

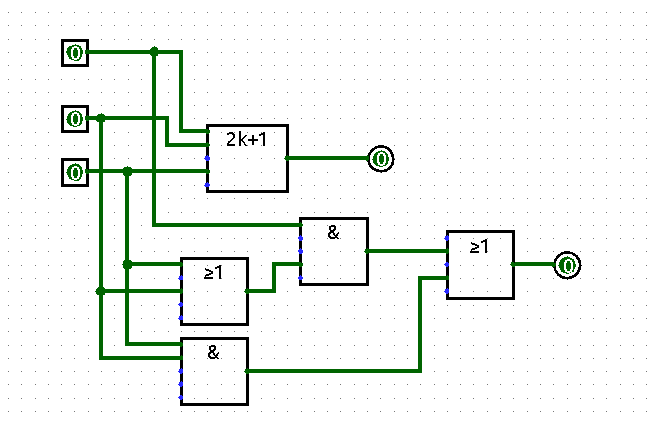


Рис. 4

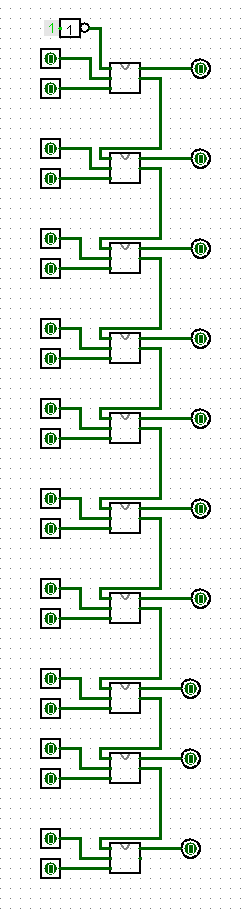


Рис. 5

Далее требовалось составить схему для возведения в квадрат. Для этого я использовал умножение в столбик. Нужно сдвинуть входное число 5-битное число влево 5 раз с приписыванием нуля справа (умножить на 2), а потом сложить эти сдвиги с коэффициентами, равными соответствующим цифрам исходного числа. Вспомогательная схема sqr\_node принимает в себя бит текущего числа, соответствующий сдвигу, а также сдвиг, и возвращает число с нужным коэффициентом (Рис. 6).

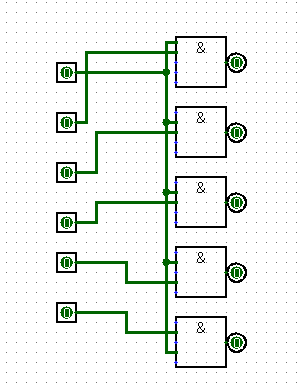
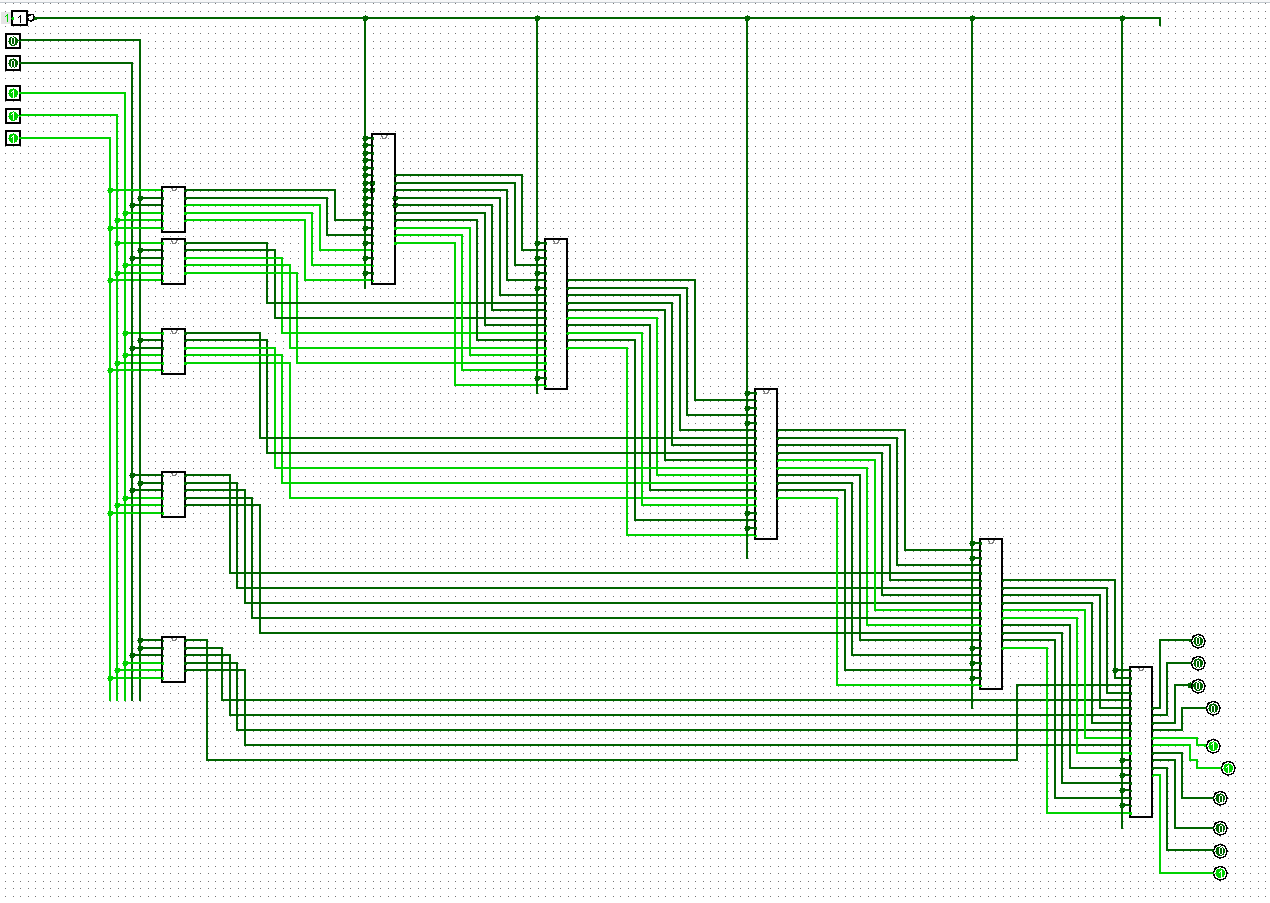


Рис. 6

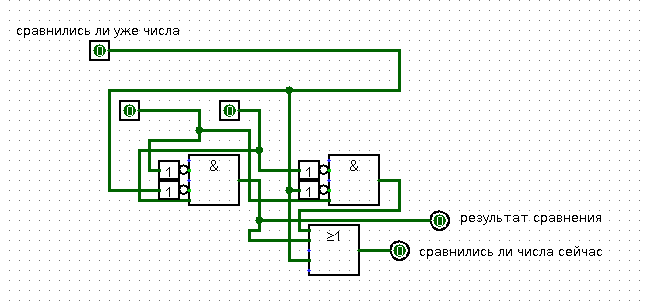
Потом получил нужные 5 сдвигов и сложил их при помощи сумматоров. Полученная схема представлена на рисунке 7.

  
Рис. 7

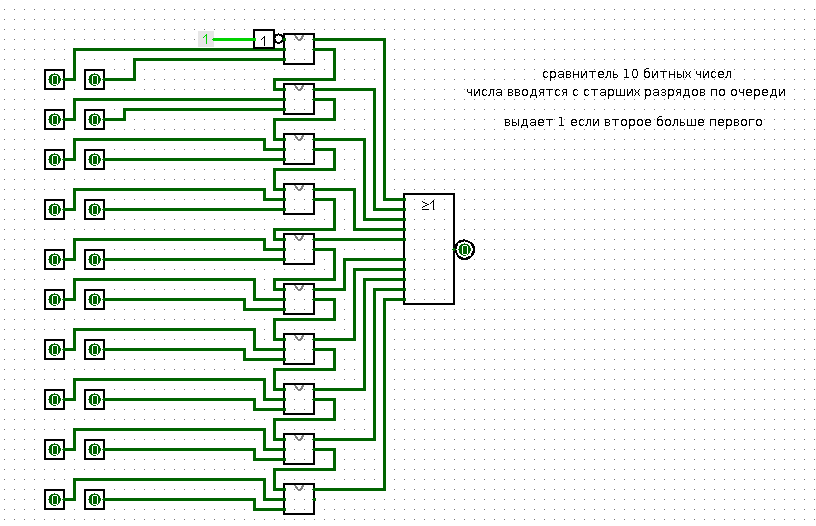
**Сравнение чисел.**

Нужно научиться сравнивать два 10-битных числа. Алгоритм будет следующий: сравним биты попарно, начиная со старших, первый момент отличия и определит результат сравнения.

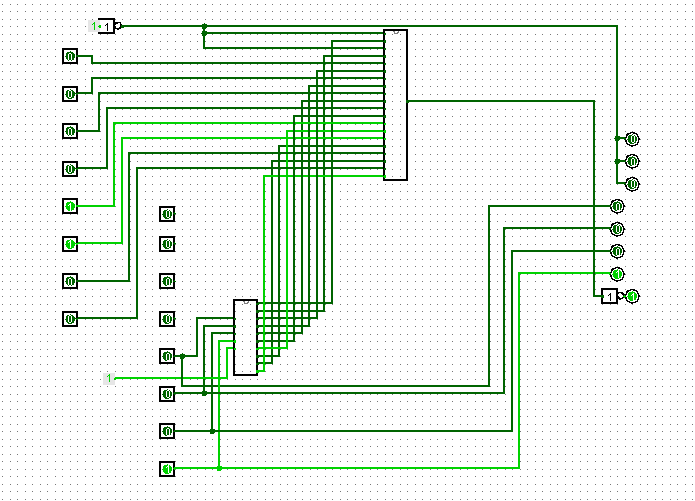
Изготовим схему cmp\_node, которая принимает в себя два сравниваемых бита из одного разряда, а также булевый флаг, равный 1, если на старших битах уже было выполнено сравнение (была пара неравных). Возвращать она будет 1, если второе число больше первого в этом разряде, а также, если до этого еще не было выполнено сравнения. А также новое, пересчитанное значение нашего флага (для следующего разряда). Такая схема представлена на рисунке 8.

  
Рис. 8

Чтобы реализовать сравнения нужно выстроить 10 таких схем и подвести соответствующие сигналы. Схема на рисунке 9.

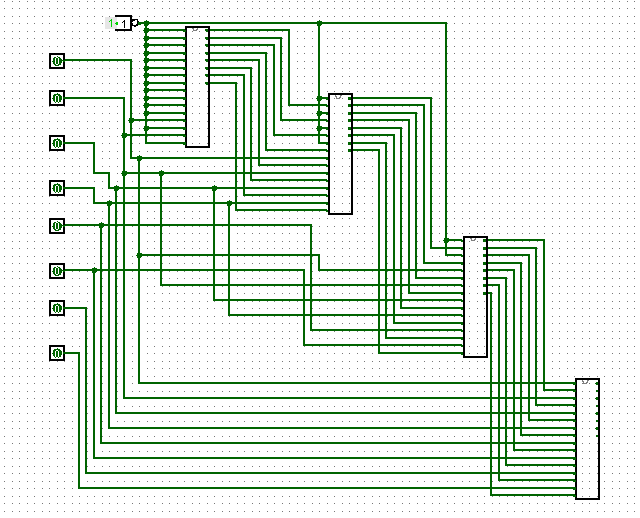
  
Рис. 9

**Схема, выполняющая шаг алгоритма.**После того, как у нас есть все эти подсхемы, можно изготовить подсхему, принимающую в себя результат корня для числа, из которого убрали два младших бита, а также само это число, и возвращающую новое подходящее значение корня. Она представлена на рисунке 10.

  
Рис. 10

Данная схема умножает результат на 2. Далее ставит в младший бит 1 и возводит число в квадрат. Потом с помощью сравнения определяется, оставлять ли младший бит.

**Основная схема взятия корня.**  
Теперь можно изготовить саму схему взятия корня. Для этого нужно проделать вышеизложенную операцию 4 раза: в начале вычислить корень для двух старших битов (для 0 старших битов он равен 0), далее для 4, для 6 и, наконец, для 8. Таким образом, получается схема как на рисунке 11.

  
Рис. 11