САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Архитектура ЭВМ

Отчет

по домашней работе № 2

«Построение сложных логических схем»

Выполнил: Рассадников Григорий

Номер ИСУ: 334838

студ. гр. М3135(М3138)

Санкт-Петербург

2022

Цель работы: моделирование сложных логических схем.

Инструментарий и требования к работе: работа выполняется в logisim evolution, logisim.

Теоретическая часть

Триггеры – последовательность устройств, которые могут как менять свои состояния, так и сохранять текущие. По функциональному признаку различают RS, D, T, JK. Существуют синхронные и асинхронные триггеры. В асинхронный триггер запись происходит при получении сигнала. Синхронный имеет еще один вход, который позволят синхронизировать работу, и меняет значение только получении единичного импульса синхронизации.

RS триггер имеет три входа, в случае если мы хотим синхронизацию. Входы: Reset, Set, C (синхронизация). И 2 выхода: значение и отрицание значения. Рисунок 1 – таблица истинности для RS триггер. Рисунок 2 – схема RS триггера.

S	R	Q	Режим
0	0	Q	Хранение
0	1	0	Установка 0
1	0	1	Установка 1
1	1	-	Неопределенность

Рисунок 1– таблица истинности для RS триггер

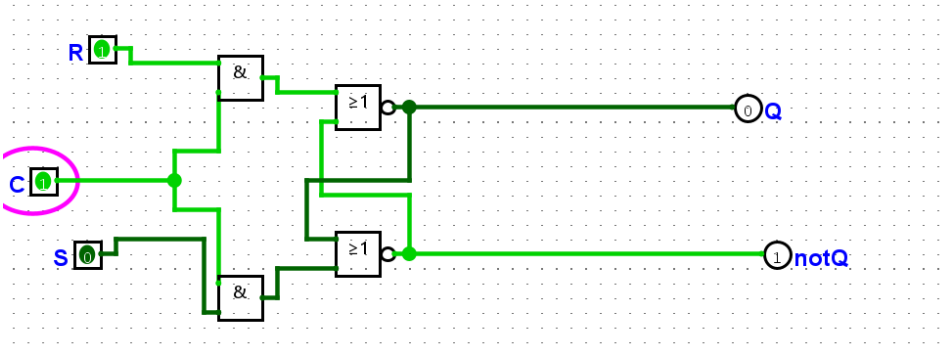

Рисунок 2 – схема RS триггера

JK триггер похож на RS, но не имеет запрещённых комбинаций входа, а именно вход (1,1) меняет значение на противоположное. Reset позволяет установить начальное значение счетчика. Рисунок 3 – таблица истинности для JK триггера. Рисунок 4 – схема RS триггера.

J	K	Q	Режим
0	0	Q	Хранение
0	1	0	Установка 0
1	0	1	Установка 1
1	1	¬Q	Инвертирование

Рисунок 3 – таблица истинности для JK триггера

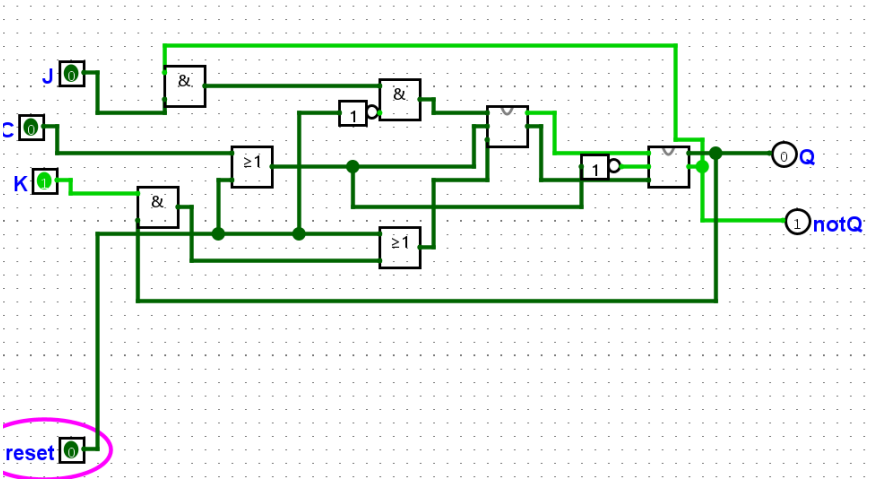

Рисунок 4 – схема JK триггера

D триггер позволяет записать значение, которое подается на вход. Рисунок 5 – таблица истинности для D триггера.

C	D	Q	!Q
0	0	Q	!Q
0	1	Q	!Q
1	0	0	1
0	1	1	0

Рисунок 5 – таблица истинности для D триггера

Счетчик — устройство, на выходах которого получается двоичный или двоично-десятичный код, определяемый числом поступивших импульсов. По существу, счетчик представляет собой совокупность соединенных определенным образом триггеров. Основной параметр счетчика — модуль счета. Это максимальное число единичных сигналов, которое может быть сосчитано счетчиком.

Счётчики используются для построения таймеров или для выборки инструкций из ПЗУ в микропроцессорах. Они могут использоваться как делители частоты в управляемых генераторах частоты.

Простейший вид счётчика — двоичный может быть построен на основе Т триггера. Т-триггер изменяет своё состояние на прямо противоположное при поступлении на его вход синхронизации импульсов. Для реализации Т триггера можно воспользоваться универсальным D-триггером с обратной связью, как это показано на рисунке 6

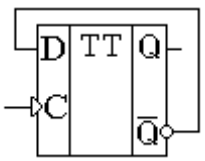

Рисунок 6 – Т-триггер

Асинхронный вычитающий счетчик: счётчики могут не только увеличивать своё значение на единицу при поступлении на вход импульсов, но и уменьшать его. Такие счётчики получили название вычитающих счётчиков. Для реализации вычитающего счётчика достаточно чтобы Т-триггер срабатывал по переднему фронту входного сигнала. Это можно осуществить инвертированием этого сигнала. В схеме, приведенной на рисунке 6, для реализации вычитающего счётчика сигнал на входы последующих триггеров подаются с инверсных выводов предыдущих триггеров. Реализация асинхронного вычитающего счетчика с 4 выходами на рисунке 7. Q0, Q1, Q2, Q3 – выходы счетчика.

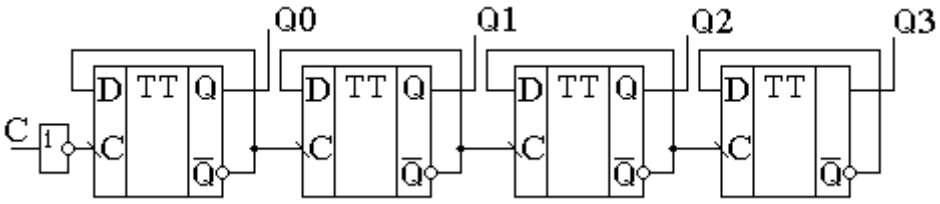

Рисунок 7 – Асинхронный вычитающий счетчик с 4 выходами

Двоичный счетчик считаем по модулю счета кратному степени двойки (зависит от количества триггеров). Чтобы счетчик считал по модулю M необходимо добавить волну сброса, по которой будет идти 0 в случае, если мы совершили M операций с нашим счетчиком после последнего сброса.

Практическая часть (работа выполнялась в Logisim evolution. Возможно, при запуске собьются число входов у логической схемы, такое бывает в некоторых версиях evolution. При запуске в обычном Logisim такого не наблюдается, но могут возникнуть проблемы с возбуждением, которых нет в evolution. Прошу в случае проблем связаться со мной, т. к. у меня все работает, используя evolution)

Составим схемы для RS, JK триггеров. Придумаем свой триггер JKprotiv, который будет отличаться только начальным значением триггера, которое устанавливает reset.

Составим схему для асинхронного вычитающего счетчика по модулю 11, используя JK триггеры. Т. к. просят модуль 11, то наш счетчик принимает значения [0,10]. Поэтому достаточно 4 триггера и сигналов выхода для кодирования значения. Q0, Q1, Q2, Q3 – выходы, от младшего к старшему разрядам. X_1_1 является тактовым генератором, который меняет значение на противоположный. Кнопка RESET приводит схему в начальное состояние 0101. Счетчик уменьшает значение от 10 до 0, меняя значения, которые хранят триггеры. Когда значение во всех триггерах становится равным 0, то вход reset у всех триггеров становится равным 0, т. е. все счетчики приходят в начальное состояние. Начально состояние нашей схемы это 0101, поэтому первый и третий счетчик должны хранить противоположное начальное значение, поэтому там используем счетчик JKprotiv. Реализуем наша схему, на выходах которой будут светодиоды. Используя встроенную функцию Logisim evolution, промоделируем работу счетчика и составим временную диаграмму. Рисунок 8 – временная диаграмма. Рисунок 9 – схема асинхронного вычитающего счетчика по модулю 11.

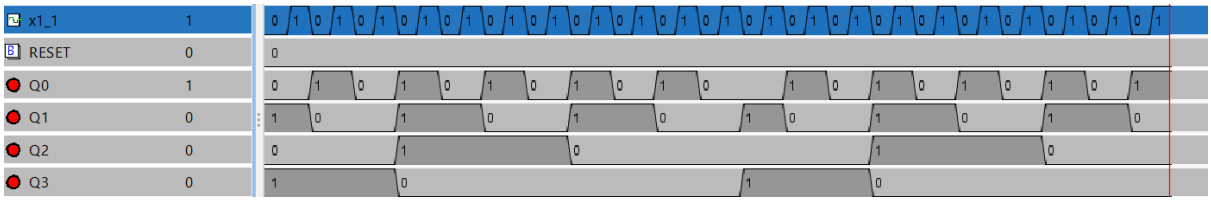

Рисунок 8 – Временная диаграмма

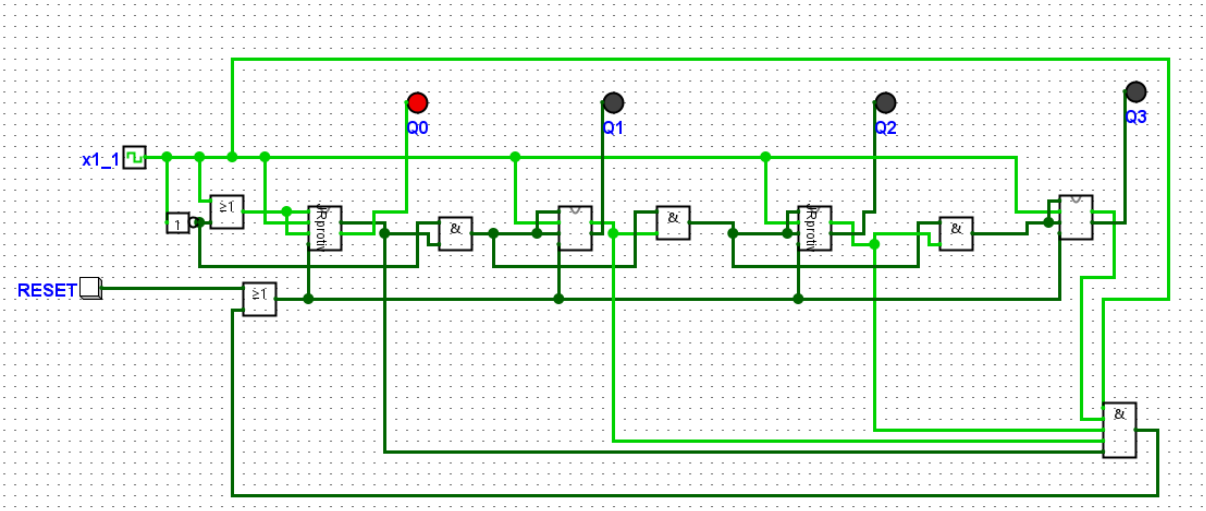

Рисунок 9 – схема асинхронного вычитающего счетчика по модулю 11

Схема корня.

Воспользуемся функцией комбинационного анализа в Logisim evolution. С ее помощью можно загрузить таблицу истинности входных и выходных значений. Так как нам требуется вычислять корень для восьмибитных чисел, то необходимо уметь вычислять корень для целых чисел из диапазона [0;255]. Тогда ответ будет лежать в [0;15] (т. к. тип округления к 0), что означает, что для вывода результата достаточно 4 бита. Составим таблицу истинности для всех возможный комбинаций 8 входных бит используя Python (а именно перебрав все числа из [0;255], вычислив для них корень и записав ответ). Полученную таблицу скопируем в текстовый файл и преобразуем к входному формату для комбинационного анализа. В результате получим построенную схему корня, результат которой, соответствует таблице истинности. Разберем поэтапно ходы выполнения:

1. Открываем окно для комбинационного анализа. Рисунок 10 – комбинационный анализ.

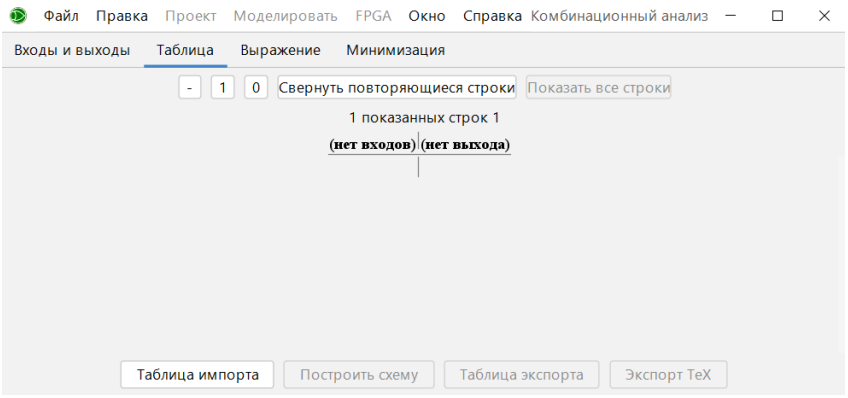

Рисунок 10 – комбинационный анализ

2. Напишем программу для составления таблицы истинности. Перебреем входные биты i7..i0 (представляя числа в двоичном представлении этих бит, получим что мы перебираем значения от 0 до 255), которые принимают значения 0 или 1. Переведем число из двоичного представления в десятичное, вычислим корень, назад переведем в двоичный формат, напечатаем результат out3…out0. Опустим детали реализации, главная цель, понять какую идею реализует этот код. Рисунок 11 – программа для составления таблицы истинности.

```
for i7 in range(0,2):
    for i6 in range(0,2):
        for i5 in range(0,2):
            for i4 in range(0,2):
                for i3 in range(0,2):
                    for i2 in range(0,2):
                        for i1 in range(0,2):
                            for i0 in range(0,2):
                                x=str(i7)+str(i6)+str(i5)+str(i4)+str(i3)+str(i2)+str(i1)+str(i0)
                                x=int(x,2)
                                print(i7,"", i6,"",
                                      i5,"", i4,"",
                                      i3,"", i2,"",
                                      i1,"", i0,"",
                                      "| ",end="")

                                s=int(x**0.5)
                                s=bin(s)
                                s=s[2:]
                                s="000"+s
                                print(s[-4],"",s[-3],"",s[-2]," ",s[-1])
```

Рисунок 11 – программа для составления таблицы истинности

3. Рисунок 12 – Начало результата программы.

```
========== RESTART: C:\Users\Grigorii\itmo\sem-
0  0  0  0  0  0  0  0  | 0  0  0  0
0  0  0  0  0  0  0  1  | 0  0  0  1
0  0  0  0  0  0  1  0  | 0  0  0  1
0  0  0  0  0  0  1  1  | 0  0  0  1
0  0  0  0  0  1  0  0  | 0  0  1  0
0  0  0  0  0  1  0  1  | 0  0  1  0
0  0  0  0  0  1  1  0  | 0  0  1  0
0  0  0  0  0  1  1  1  | 0  0  1  0
0  0  0  0  1  0  0  0  | 0  0  1  0
0  0  0  0  1  0  0  1  | 0  0  1  1
0  0  0  0  1  0  1  0  | 0  0  1  1
```

Рисунок 12 – Начало результата программы

4. Скопируем этот результат, и вставим его в текстовый файл. Преобразуем входной файл, согласно правилам, указанным в руководстве пользователя Logisim. In7…in0 – входные значения, out3…out0 – выходные значения. Рисунок 13 – входной файл для комбинационного анализа.

table_true.txt – Блокнот

Файл Правка Формат Вид Справка

```
in7 in6 in5 in4 in3 in2 in1 in0 | out3 ou2 out1 out0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
0  0  0  0  0  0  0  0  | 0  0  0  0
0  0  0  0  0  0  0  1  | 0  0  0  1
0  0  0  0  0  0  1  0  | 0  0  0  1
0  0  0  0  0  0  1  1  | 0  0  0  1
0  0  0  0  0  1  0  0  | 0  0  1  0
0  0  0  0  0  1  0  1  | 0  0  1  0
0  0  0  0  0  1  1  0  | 0  0  1  0
0  0  0  0  0  1  1  1  | 0  0  1  0
0  0  0  0  1  0  0  0  | 0  0  1  0
0  0  0  0  1  0  0  1  | 0  0  1  1
```

Рисунок 13 – входной файл для комбинационного анализа

5. Импортируем этот файл, и построим таблицу. Рисунок 14 – таблица истинности в Logisim для построения схемы корня.

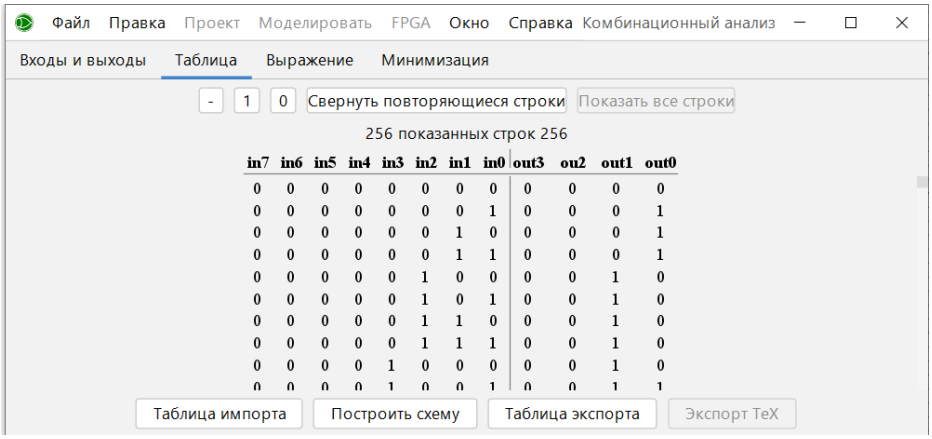

Рисунок 14 – таблица истинности в Logisim для построения схемы корня

6. В результате получим схему корня. На вход этой схеме подается 8 бит in7…in0 (двоичное представление входного числа от старшего к младшему разряду). Результат: out3…out0 (двоичное представление результата от старшего к младшему разряду). На входе и выходе у схемы – контакты. Нам не очень важно, как устроена, данная схема т. к. Logisim корректно составляет схему по таблице истинности. А т. к. таблицу мы получили благодаря перебору всех значений программой, то наша схема корректна. Ну а корректность доказательства перебора является очевидным. Рисунок 15 – схема корня.

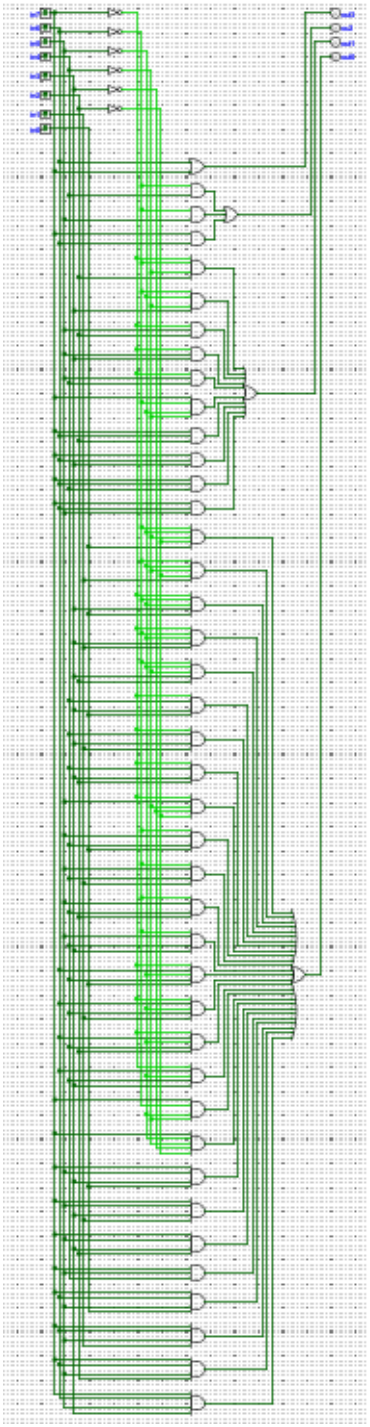

Рисунок 15 – схема корня

Листинг кода вспомогательной программы

Sqrt.py

```
for i7 in range(0,2):
    for i6 in range(0,2):
        for i5 in range(0,2):
            for i4 in range(0,2):
                for i3 in range(0,2):
                    for i2 in range(0,2):
                        for i1 in range(0,2):
                            for i0 in range(0,2):

      x=str(i7)+str(i6)+str(i5)+str(i4)+str(i3)+str(i2)+str(i1)+str(i0)
                                x=int(x,2)
                                print(i7,"", i6,"",
                                      i5,"", i4,"",
                                      i3,"", i2,"",
                                      i1,"", i0,"",
                                      "| ",end="")

                                s=int(x**0.5)
                                s=bin(s)
                                s=s[2:]
                                s="000"+s
                                print(s[-4],"",s[-3],"",s[-2]," ",s[-1])
```