Архитектура вычислительных систем. Домашнее задание №1. Вариант №3.

Комиссаров Данил Андреевич

March 2025

1 Полный отчет

Третий вариант задания. Полный сумматор (Full Adder) двух 3-битных входов.

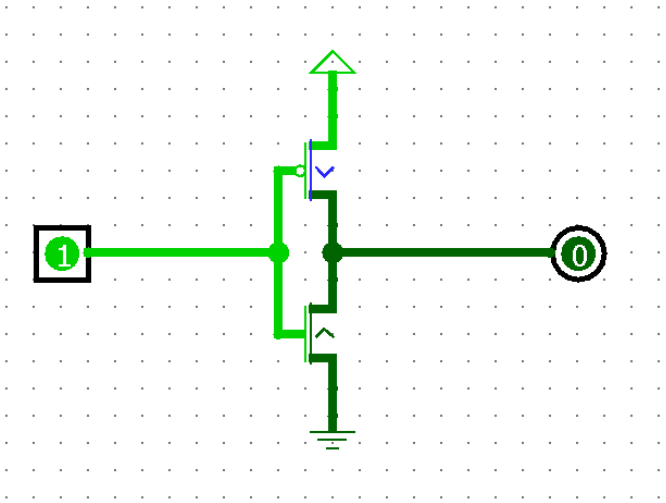
1.1 Подсчет транзисторов для базовых элементов

Доступные элементы для составления схемы:

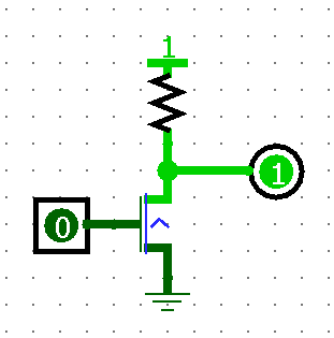
1. NOT Gate
2. AND Gate
3. NAND Gate
4. OR Gate
5. NOR GATE
6. XOR GATE
7. Probe
8. Pin
9. Соединительные провода

Сразу же подсчитаем количество транзисторов в каждом логическом элементе:

- Известно, что оптимальная реализация NOT Gate использует 2 транзистора.

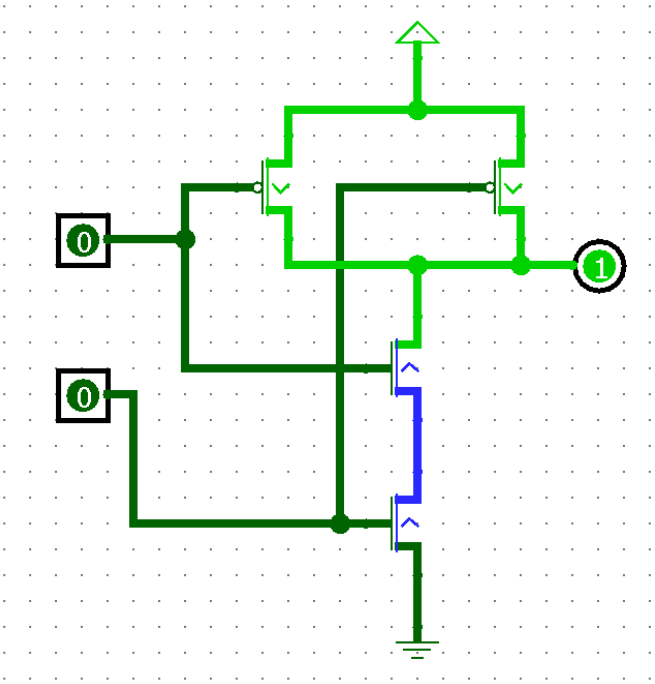


Можно, конечно же, реализовать инвертор и на одном транзисторе при помощи одного транзистора и подтягивающего резистора.



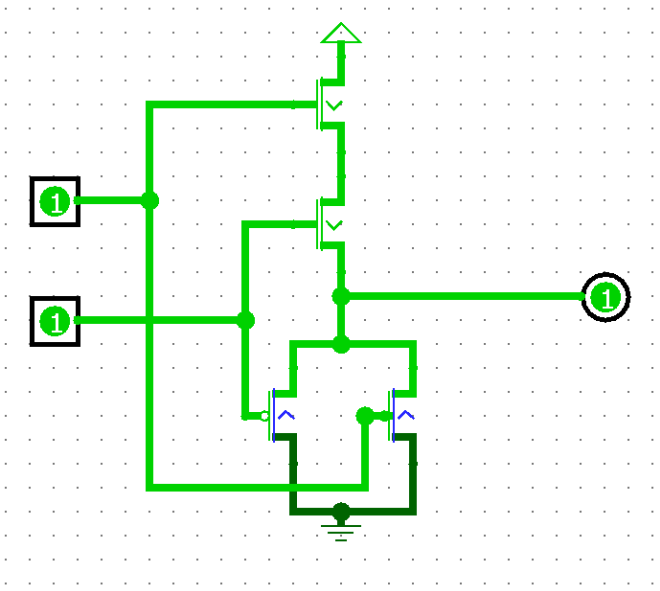
Но это не соответствует КМОП логике из-за протекающего тока через подтягивающий резистор в стабильном состоянии. Жаль. Идем дальше.

- Известно, что оптимальная реализация NAND Gate использует 4 транзистора.



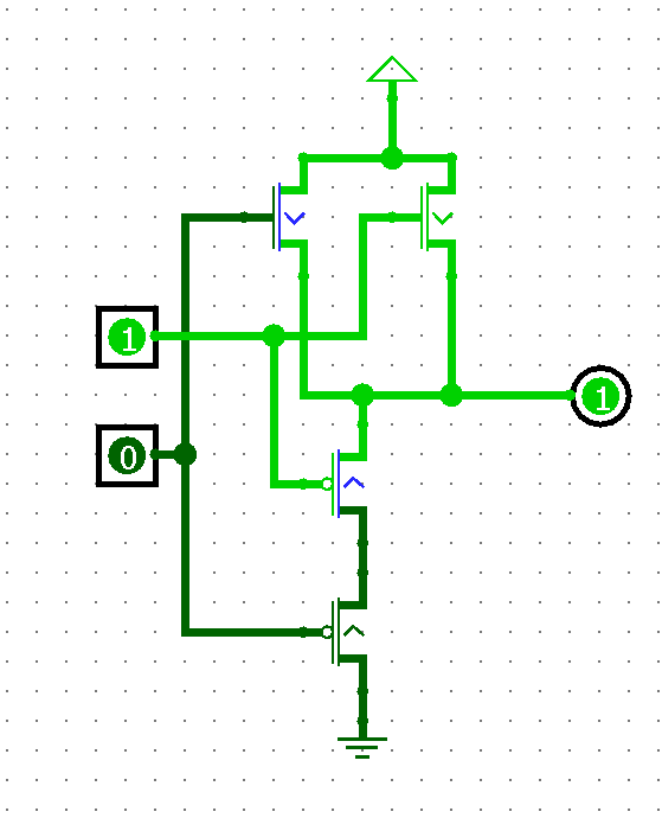
Элемент NAND известен также как "Штрих Шеффера". Интересен он тем, что образует функционально-полный логический базис для пространства булевых функций от двух переменных, так как удовлетворяет теореме Поста о полной системе функций. В электронике это означает, что для реализации любой логической схемы достаточно одного типового элемента. С другой стороны, такой подход увеличивает громоздкость и тем самым снижает их надёжность. В этом задании я поставлю себе задачу реализовать конечное решение с использованием наименьшего количества транзисторов.

- Элемент AND Gate реализуется точно так же, как и предыдущий, с точностью до замены всех типов транзисторов на противоположные.

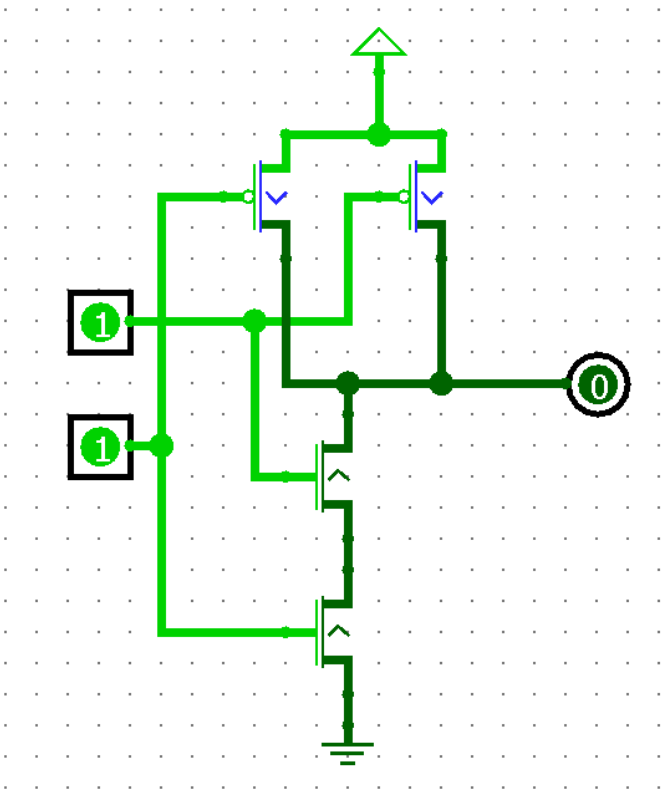


Впрочем, это верно и для любых других пар противоположных логических элементов.

- Известно, что оптимальная реализация OR Gate использует 4 транзистора.



- Аналогично NOR Gate.



- Теперь добрались до самого сложного: XOR Gate.

  У нас нет элемента исключающего ИЛИ (XOR). Нужно реализовать его на базе других логических элементов. Как это сделать? Какое минимальное количество элементов придется использовать?
  Для начала вспомним таблицу истинности для XOR.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

  Что делать? Нужно избавиться от самого оператора XOR. Для этого можем расписать функцию по таблице истинности двумя способами. Существуют такие вещи как **СКНФ**(совершенная конъюнктивная нормальная форма) и **СДНФ**(совершенная дизъюнктивная нормальная форма). Впрочем эти вещи эквивалентны друг другу, т.к. равны одной и той же функции. Поэтому не принципиально, каким методом пользоваться. Можно даже показать это:

$$a \text{ xor } b = (\overline{a} \wedge b) \vee (a \wedge \overline{b}) = \overline{a} \cdot b + a \cdot \overline{b} =$$

$$= (\overline{a} \cdot b + a) \cdot (a \cdot b + \overline{b}) =$$

$$= (1 \cdot (b + a)) \cdot ((\overline{a} + \overline{a}) \cdot (b + a) \cdot (b + \overline{b})) =$$

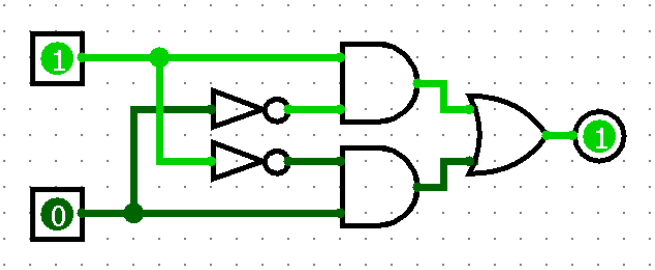
$$= (a + b) \cdot (a + b) = (a + b) \cdot (\overline{a} \cdot b)$$

Итак, есть две формы:

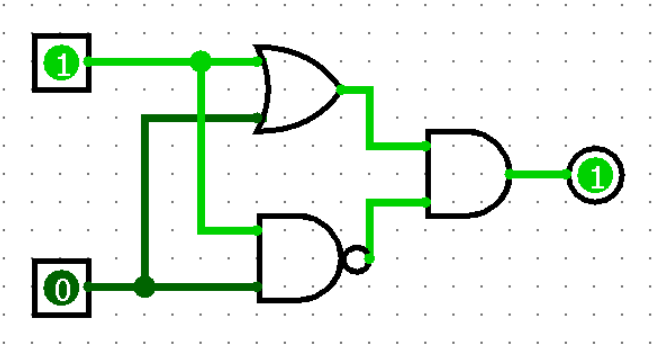
$$a \text{ xor } b = \overline{a} \cdot b + a \cdot \overline{b}$$

$$a \text{ xor } b = (a + b) \cdot (\overline{a} \cdot b)$$

Обратите внимание, чаще всего везде используется первая формула. Но проблема в том, что для её построения на логических элементах нам нужно 5 логических элементов:



Если же мы возьмем вторую формулу, то нам удастся реализовать XOR, используя только три элемента: NAND, OR, AND.



А что если собрать только на NAND?

$$a = \overline{a} + a = a \cdot a = (a \text{ nand } a)$$

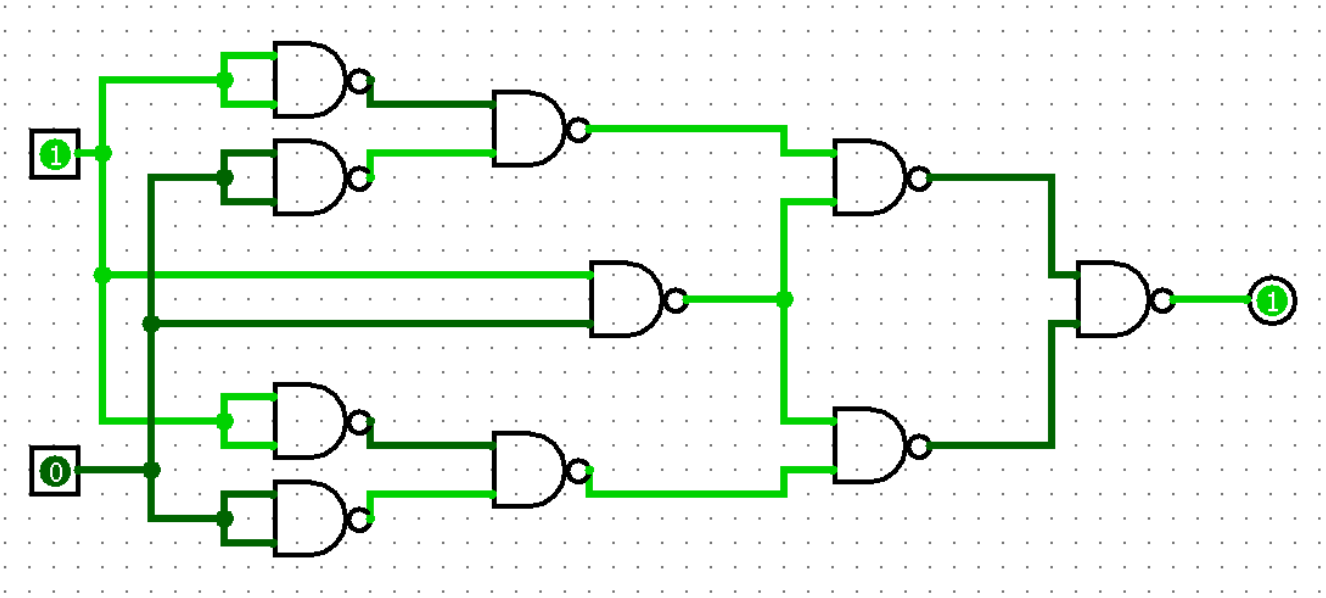
$$a + b = a \cdot b = (\overline{a} + \overline{a}) \cdot (\overline{b} + \overline{b}) = (a \text{ nand } a) \text{ nand } (b \text{ nand } b)$$

$$a \cdot b = a \cdot b = a \cdot b \cdot a \cdot b = (a \text{ nand } b) \text{ nand } (a \text{ nand } b)$$

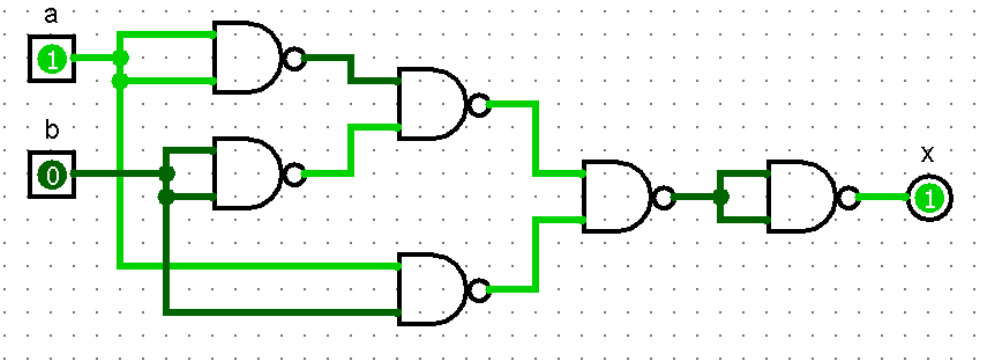
Выражаем XOR через NAND:

$$a \text{ xor } b = (a + b) \cdot \overline{(a \cdot b)} =$$

$$= \overline{\overline{(a \cdot a)} \cdot \overline{(b \cdot b)}} \cdot \overline{(a \cdot b)} = \overline{\overline{\overline{(a \cdot a)} \cdot \overline{(b \cdot b)}} \cdot \overline{(a \cdot b)}} \cdot \overline{(a \cdot b)}$$



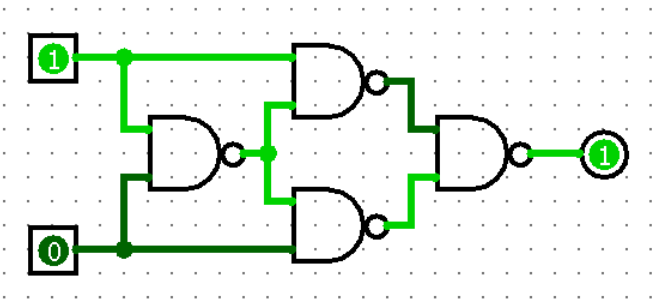
Получилось очевидно громоздко, и даже встроенный анализ Logisim дает результат намного лучше.



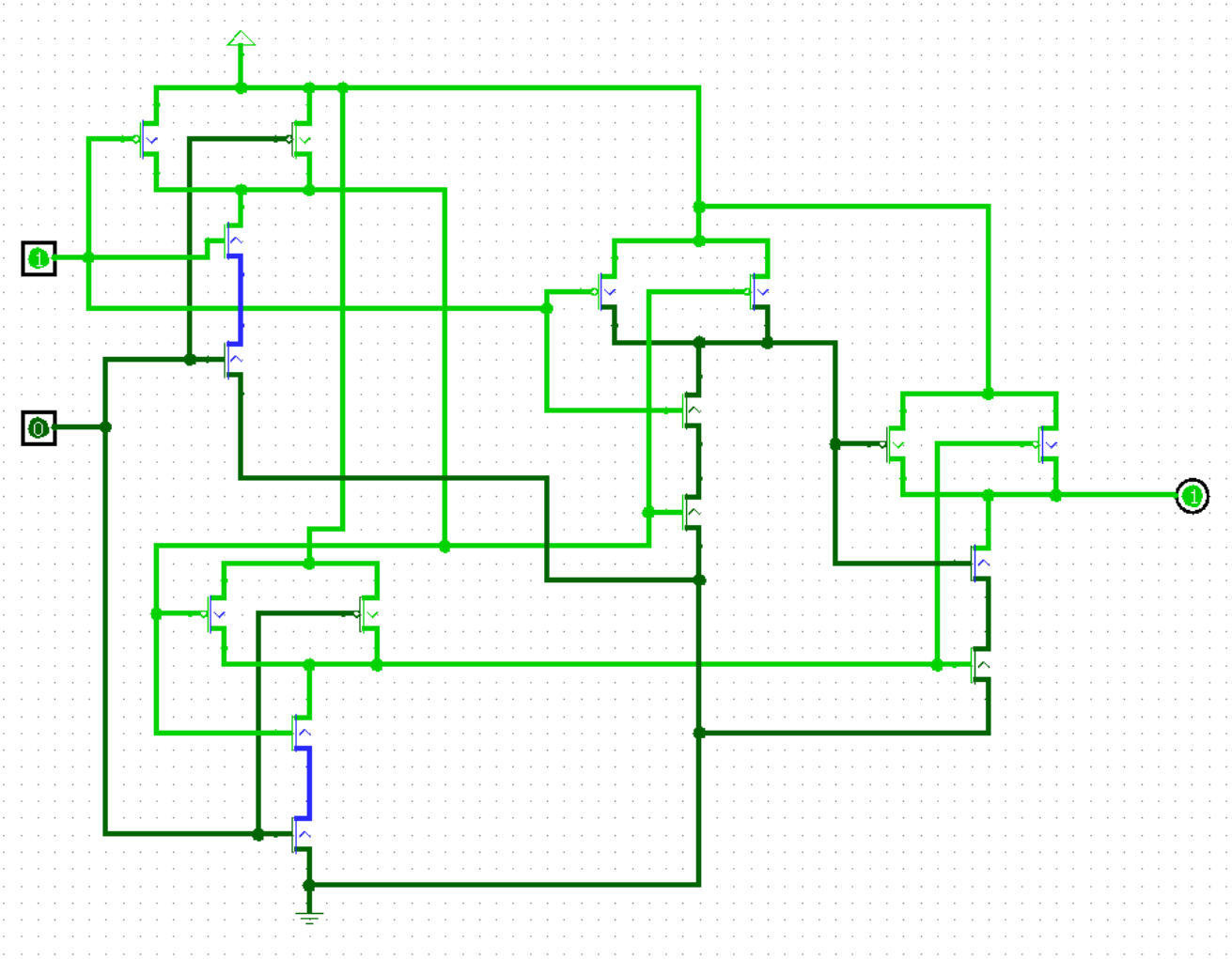
Впрочем, очевидно и то, как можно еще сильнее сократить формулу.

$$a \text{ xor } b = (a+b)\cdot(\overline{a\cdot b}) = a\cdot(\overline{(a\cdot b)}) + b\cdot(\overline{(a\cdot b)}) =$$

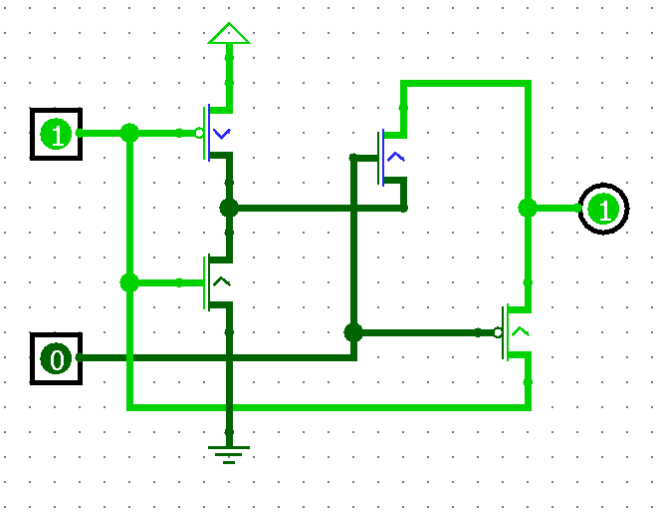
$$= a\cdot(\overline{(a\cdot b)})\cdot b\cdot(\overline{(a\cdot b)})$$



Теперь это можно и расписать в транзисторах.

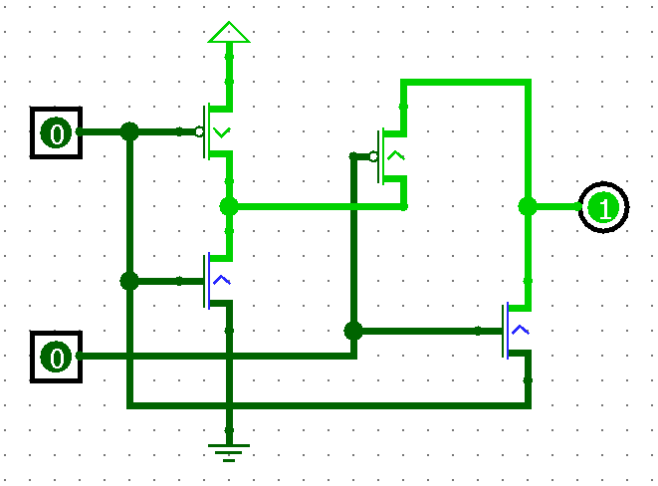


Раскрытие элементов позволяет снова увидеть некоторые паттерны в схеме и сократить схему до конечного:



Дальнейшее упрощение не представляется возможным.

- Элемент NXOR Gate аналогичен.

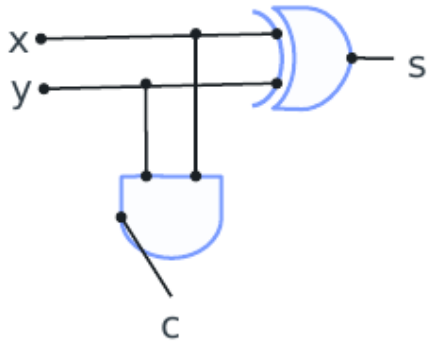


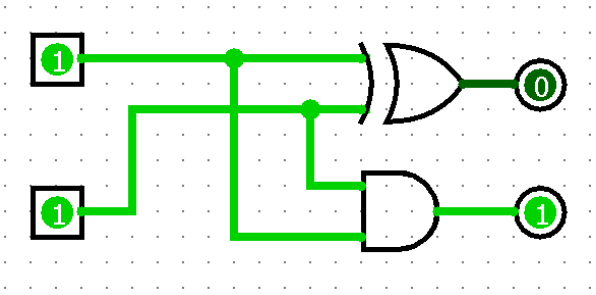
Логический элемент	Кол-во т-ров
NOT Gate	2
AND Gate	4
NAND Gate	4
OR Gate	4
NOR Gate	4
XOR Gate	4

Подсчитали наконец-то. Теперь можно и начинать составлять схему сумматора.

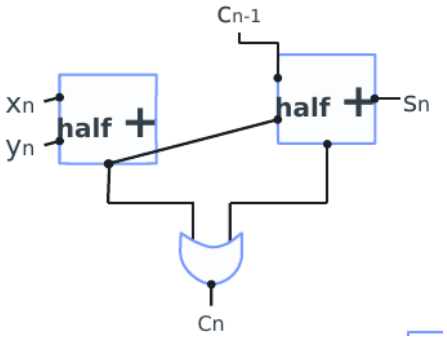
1.2 Составление схемы сумматора

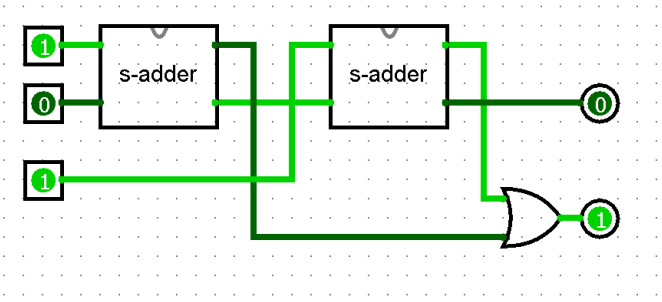
Опираться придется на материалы из лекции. Сначала собирается полусумматор:



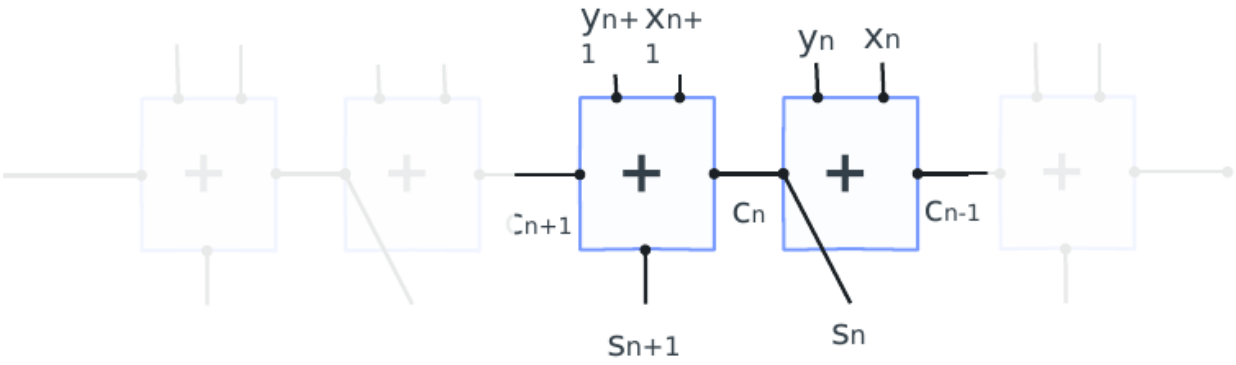


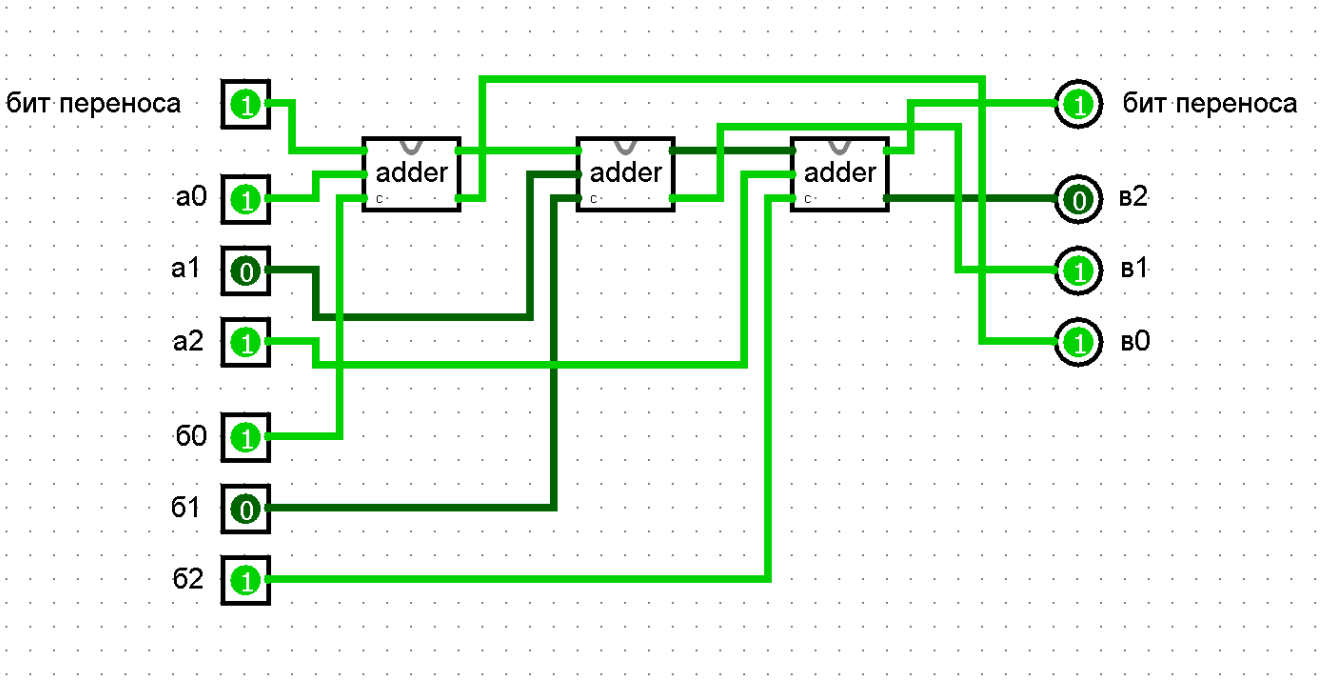
Тривиально.
Далее собираем из них полные сумматоры.





А теперь просто каскадируем их.





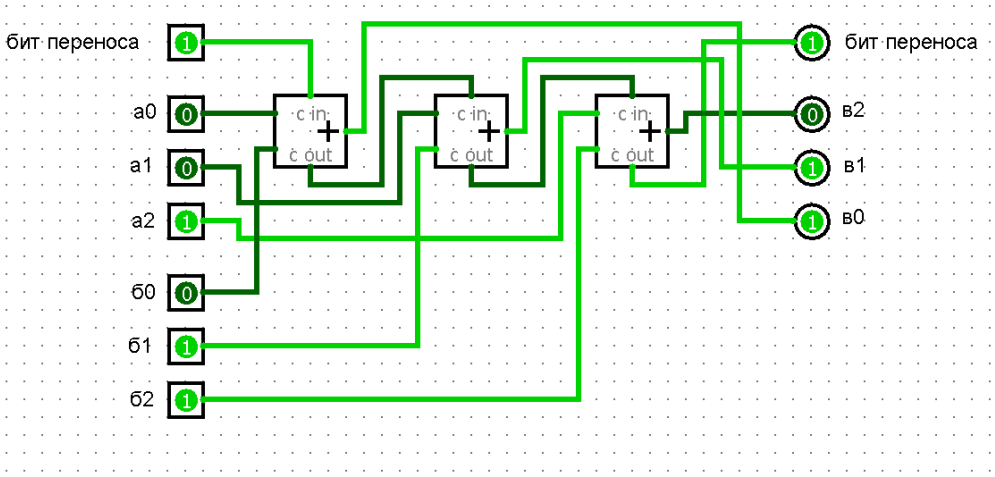
Итого:

Logisim: статистика 3-bit adder

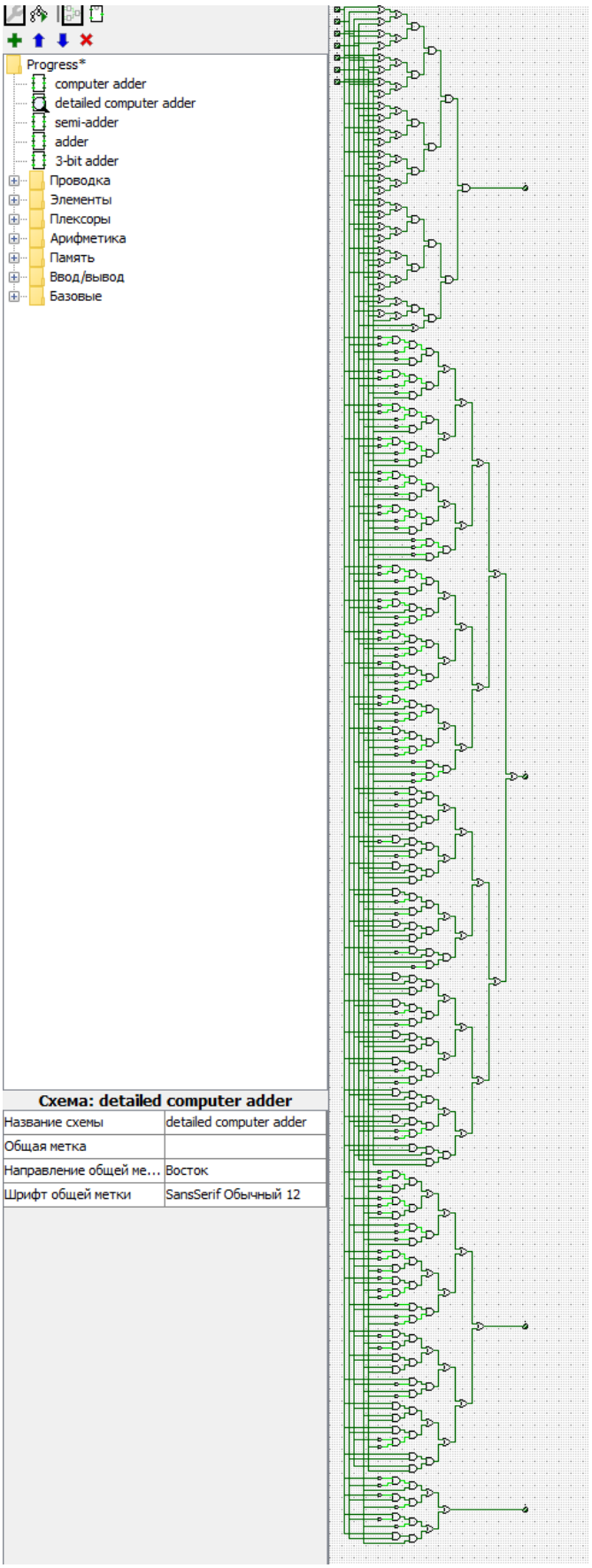
Компонент	Библиотека	Непос...	Уника...	Рекур...
semi-adder	Progress	0	2	6
adder	Progress	3	3	3
Контакт	Проводка	11	20	50
Элемент И	Элементы	0	1	6
Элемент ИЛИ	Элементы	0	1	3
Элемент Исключающее ИЛИ	Элементы	0	1	6
Метка	Базовые	11	11	11
ВСЕГО (без подсхем проекта)		22	34	76
ВСЕГО (с подсхемами)		25	39	85

Компонент	**Рекур.**	**Кол-во т-ров.**
Элемент И	6	24
Элемент ИЛИ	3	12
Элемент Исключающее ИЛИ	6	24
Итого		**60**

Здорово. Но помимо этого можно и доверить построение схемы самому Logisim. Воспользуемся встроенным сумматором (приходится расписывать однобитными элементами, т.к. к сожалению анализ схемы с многобитными элементами недоступен).



А теперь компьютер раскроет схему.

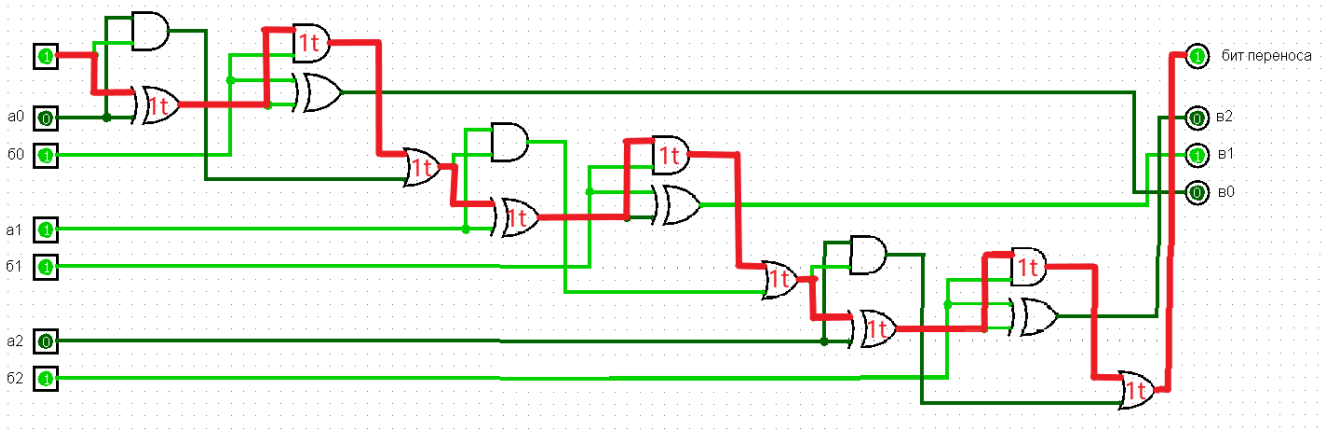


Получилось довольно громоздко.

Logisim: статистика detailed computer adder

Компонент	Библиотека	Непос...	Уника...	Рекур...
Контакт	Проводка	11	11	11
Элемент НЕ	Элементы	98	98	98
Элемент И	Элементы	166	166	166
Элемент ИЛИ	Элементы	82	82	82
ВСЕГО (без подсхем проекта)		357	357	357
ВСЕГО (с подсхемами)		357	357	357

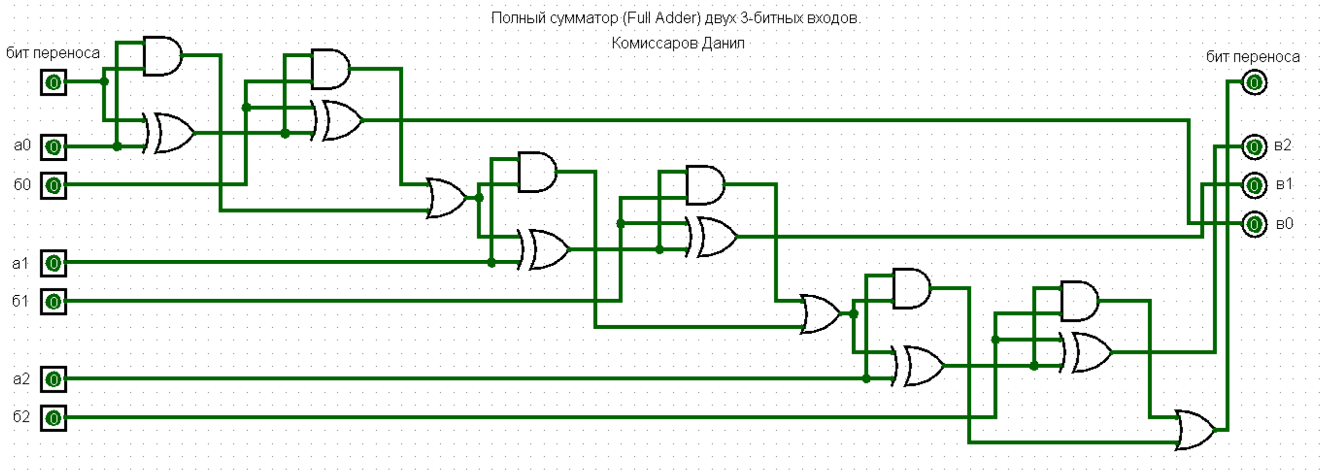
И статистика говорит о том же. Предлагаю тогда остановиться на предыдущем результате.
Тогда разобьем схему на элементы, разрешенные в задании, чтобы посчитать критический путь (предполагается, что каждый элемент из списка занимает одинаковое время).

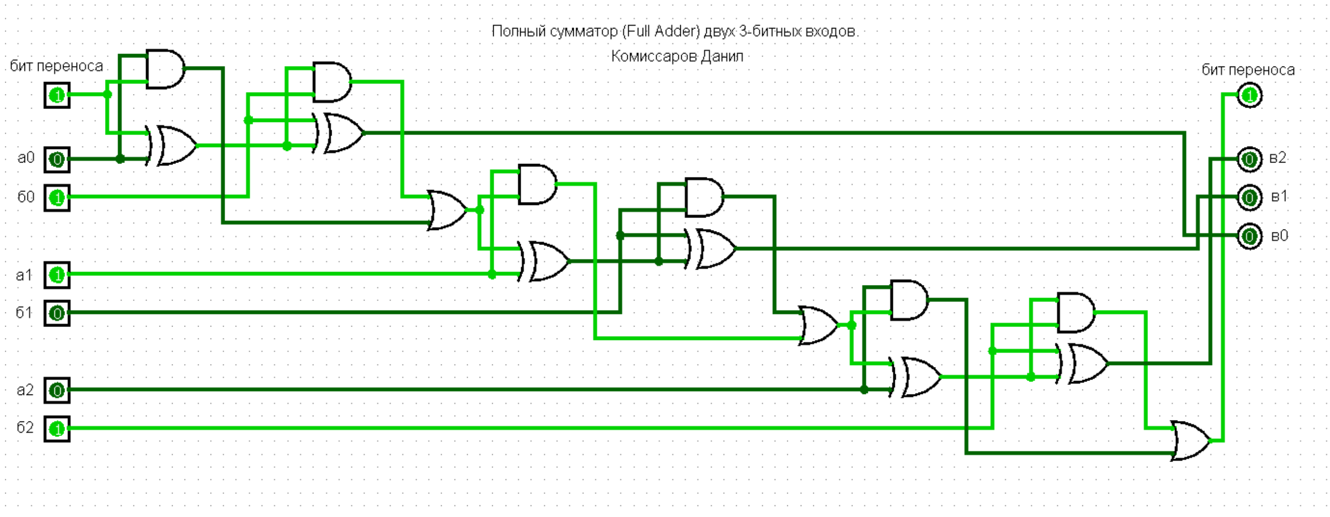


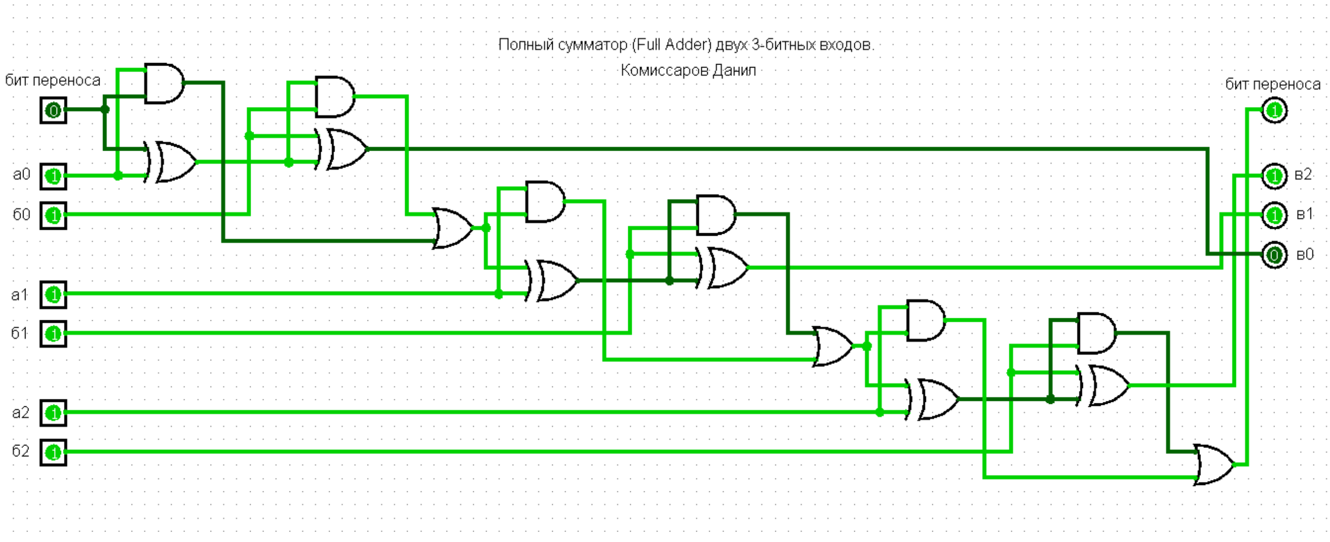
Итого 9t (хотя интересно, что в предыдущем варианте, предоставленный компьютером, критическое время составляло всего 7t). Помимо прочего, можно уточнить, что это за время t такое. Если вернуться к подсекции "Подсчет транзисторов для базовых элементов то можно заметить, что каждый из базовых элементов имеет критическое время в 1 транзистор (это только в том случае, если принять, что не требуется ожидания для сигнала при прохождении через открытый транзистор, т.е. путь коллектор-эмиттер не занимает время), т.е. для получения стабильного сигнала на выходе требуется подождать одно характерное время работы транзистора, поэтому и правда есть обоснование в том, чтобы уравнять характерные времена базовых элементов.
Когда все аспекты рассмотрены, можно занять написание "формального отчета".

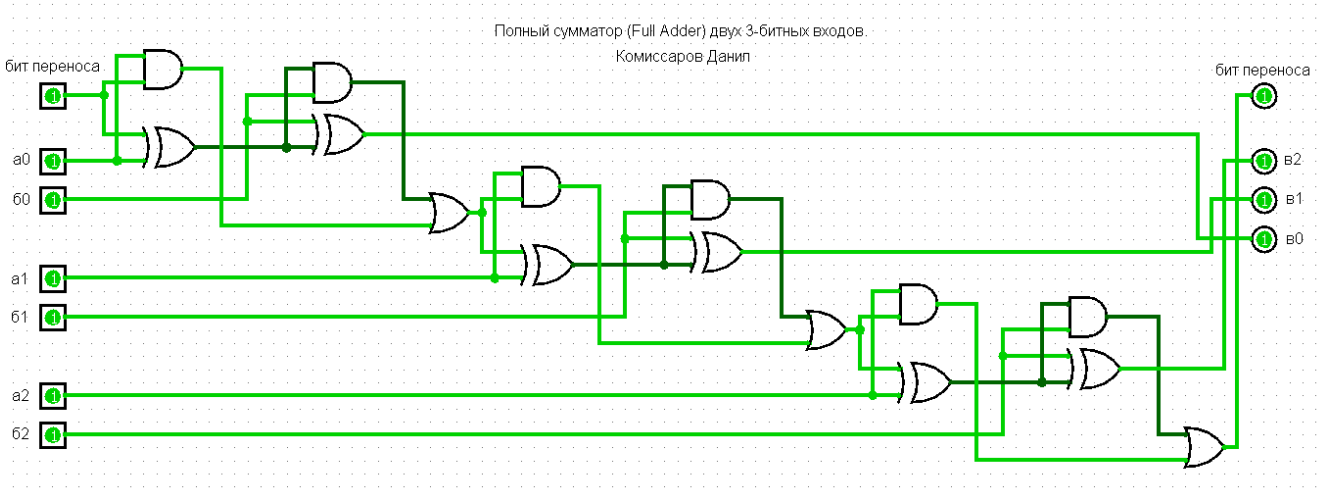
2 Формальный отчет

1. Выполнил: Комиссаров Данил Андреевич.
2. Студент группы Б01-304.
3. Выполненная схема - Полный сумматор (Full Adder) двух 3-битных входов.
4. Контакты: komissarov.da@phystech.edu
5. Полный сумматор — логическая схема, которая производит сложение двух трехбитных чисел и одного однобитного числа, обозначаемых А(а0 - младший бит, а1 - средний бит, а2 - старший бит), Б(б0 - младший бит, б1 - средний бит, б2 - старший бит) и входной бит переноса. На выход подаются трехбитное число В(в0 - младший бит, в1 - средний бит, в2 - старший бит) и выходной бит переноса., где В — это сумма по модулю 8 (а в общем случае 2^n), а выходной бит переноса - это флаг, который говорит о переполнении выхода.

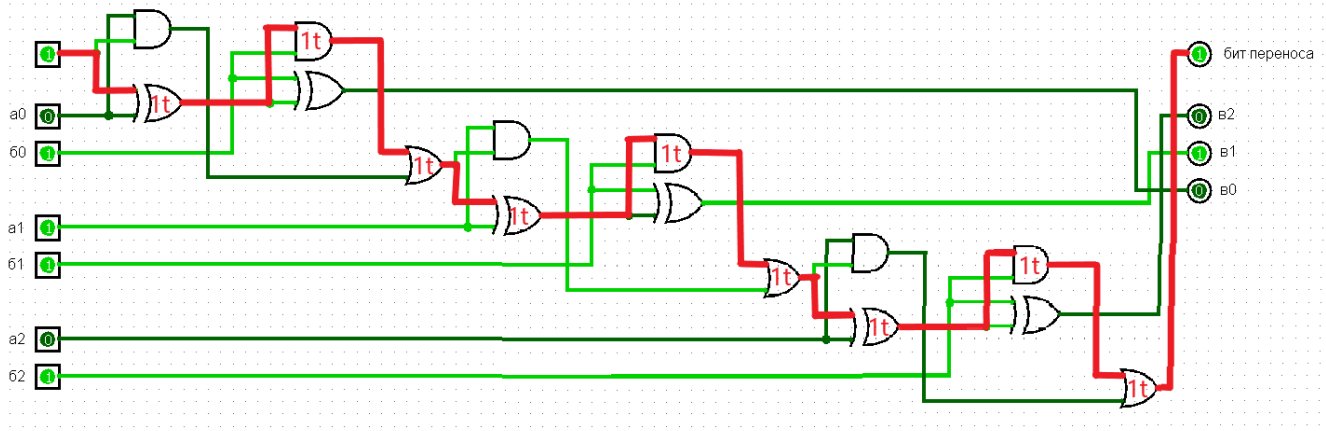

Полный сумматор (Full Adder) двух 3-битных входов.
Комиссаров Данил
бит переноса
a0
б0
a1
б1
a2
б2
бит переноса
в2
в1
в0


Полный сумматор (Full Adder) двух 3-битных входов.
Комиссаров Данил
бит переноса
a0
б0
a1
б1
a2
б2
бит переноса
в2
в1
в0


Полный сумматор (Full Adder) двух 3-битных входов.
Комиссаров Данил
бит переноса
a0
б0
a1
б1
a2
б2
бит переноса
в2
в1
в0



6.



7. Критическое время составляет 9t

8. Схема состоит из 60-ти полевых транзисторов.

На этом пожалуй можно и закончить. Наверное я уделил слишком много внимания несущественным аспектам, в то время, когда некоторые фундаментальные принципы были упущены, но это было интересно.