МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)

Лабораторная работа №2
дисциплина «Теория цифровых автоматов»
по теме «Синтез и анализ комбинационных схем с одним выходом в монофункциональных базисах»

Выполнил: студент группы ВТ-31 Макаров Д.С.
Проверил: Рязанов Ю.Д.

Белгород 2019

Лабораторная работа №2

«Синтез и анализ комбинационных схем с одним выходом в монофункциональных базисах»

Цель работы:научиться строить эффективные по быстродействию и затратам оборудования комбинационные схемы в монофункциональных базисах..

Вариант 9

Задание:

1. Преобразовать минимальную дизъюнктивную нормальную форму булевой функции (см. лабораторную работу №1, п. 2) в аналитическое выражение, содержащее только операции И-НЕ и НЕ.

2. По полученному выражению построить схему из элементов И-НЕ. Операцию НЕ реализовывать элементом И-НЕ с запараллеленными входами.

3. Комбинационную схему, полученную в лабораторной работе № 1, п. 3, преобразовать в комбинационную схему, состоящую только из элементов И-НЕ, путем замены элементов И, ИЛИ, НЕ на их логические эквиваленты, состоящие только из элементов И-НЕ. После формального построения комбинационной схемы исключить из нее пары последовательных элементов И-НЕ с запараллеленными входами.

4. Преобразовать минимальную конъюнктивную нормальную форму булевой функции (см. лабораторную работу №1, п. 4) в аналитическое выражение, содержащее только операции ИЛИ-НЕ и НЕ.

5. По полученному выражению построить схему из элементов И-НЕ. Операцию НЕ реализовывать элементом ИЛИ-НЕ с запараллеленными входами.

6. Комбинационную схему, полученную в лабораторной работе № 1,п. 5, преобразовать в комбинационную схему, состоящую только из элементов ИЛИ-НЕ, путем замены элементов И, ИЛИ, НЕ на их логические эквиваленты, состоящие только из элементов ИЛИ-НЕ. После формального построения комбинационной схемы исключить из нее пары последовательных элементов ИЛИ-НЕ с запараллеленными входами.

7. Написать программы, моделирующие работу схем, полученных в пунктах 2, 3, 5 и 6, на всех входных наборах и строящие таблицу истинности каждой схемы. Сравнить полученные таблицы истинности с таблицей истинности исходной функции в лабораторной работе № 1

8. Сравнить схемы, построенные в лабораторных работах № 1 и № 2 по Квайну и по быстродействию.

Ход работы

$$3 < (x_4x_5 + x_1x_2x_3) < 8$$

Дизъюнктивная форма

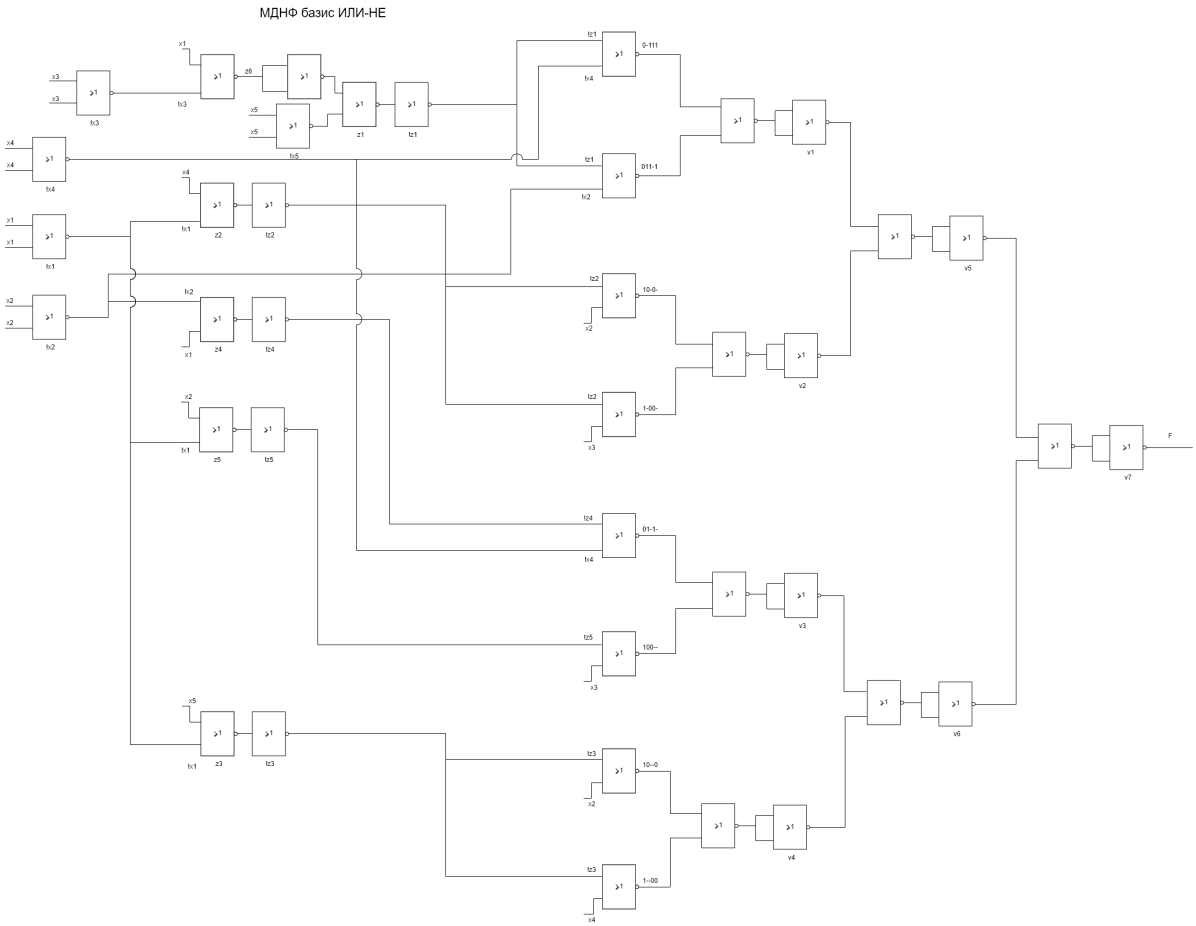

Рис. 1: МДНФ базис ИЛИ-НЕ

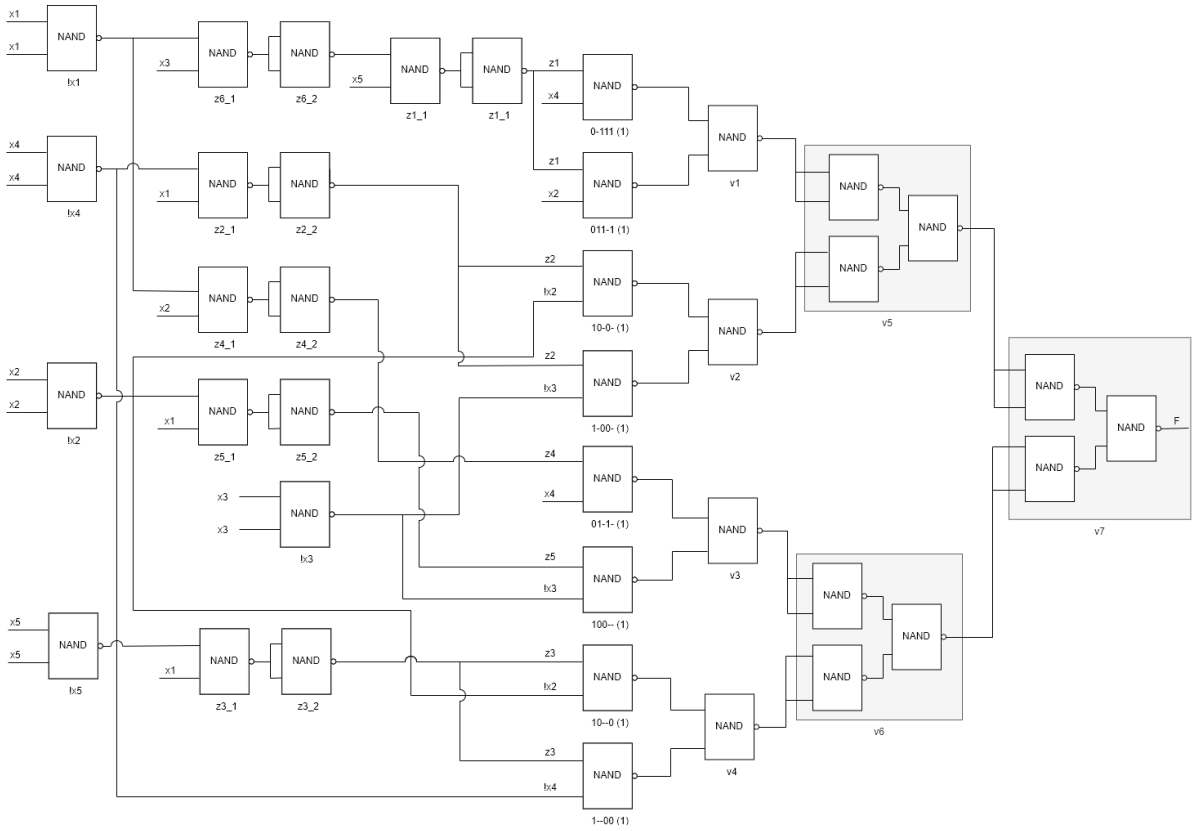

Рис. 2: МДНФ базис И-НЕ

Конъюнктивная форма

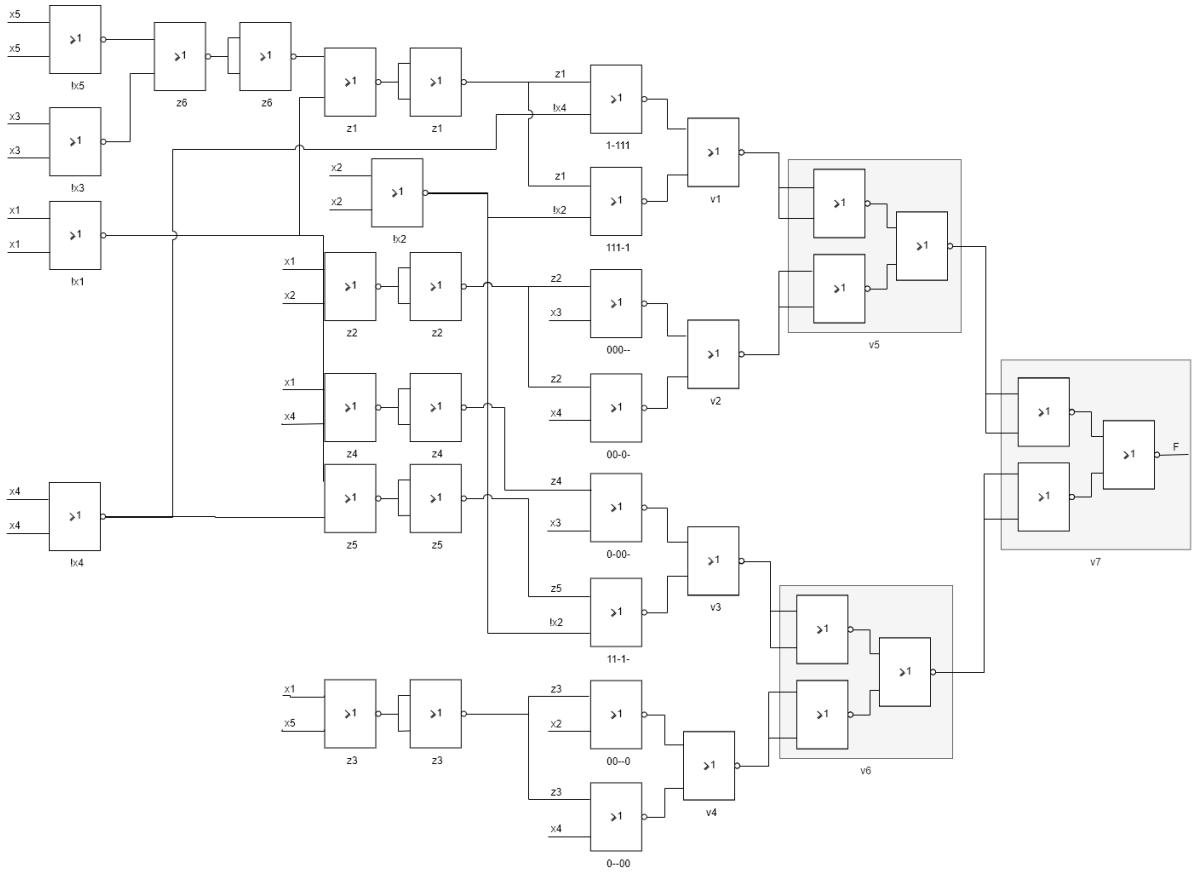

Рис. 3: МКНФ базис ИЛИ-НЕ

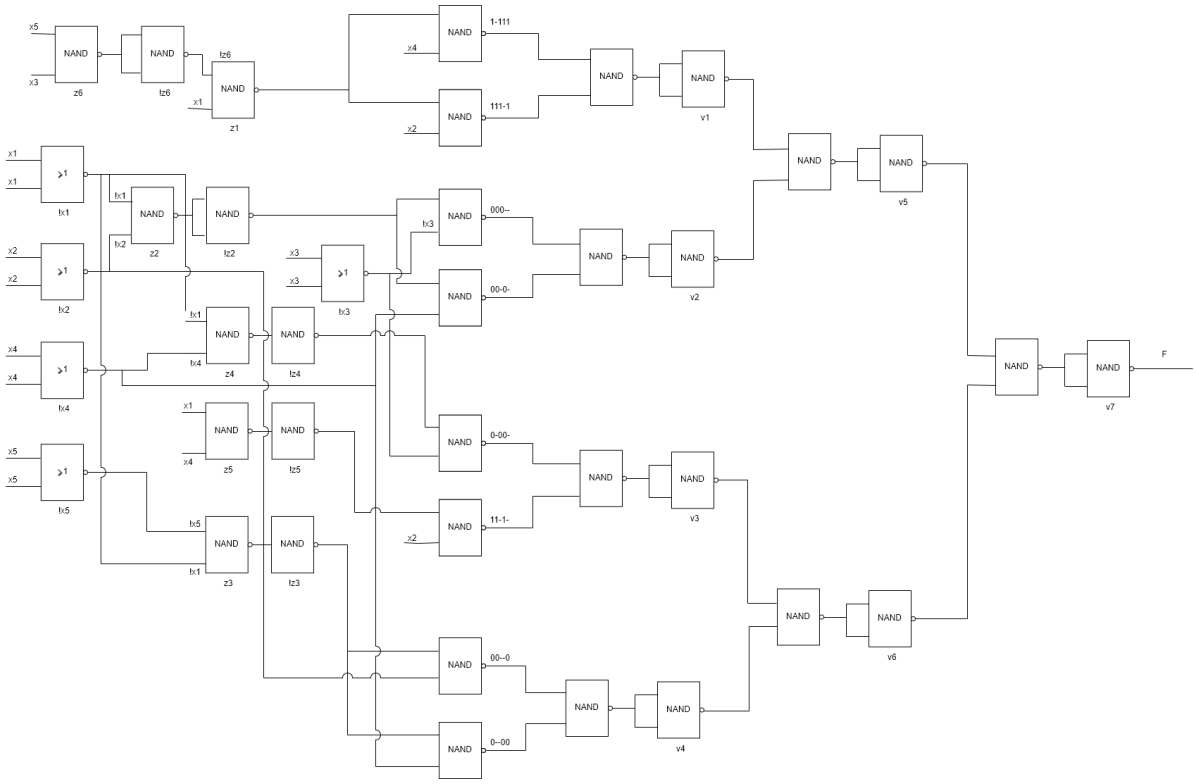

Рис. 4: МКНФ базис И-НЕ

Приложение

Содержимое файла funcTest.py

```
from binVectors import gen_bin_vector_5 as gen_bin_vector
from tabulate import tabulate

def truth_table(vector,f,f_1,f_2,f_3,f_4):
    result = []
    for i in range(0,len(vector)):
        result.append([
            i+1,
            vector[i][0],
            int(f(vector[i][0])),
            int(f_1(vector[i][0])),
            int(f_2(vector[i][0])),
            int(f_3(vector[i][0])),
            int(f_4(vector[i][0]))
        ])
    return result

def nand(a,b):
    return not(a and b)

def nand_not(a):
    return nand(a,a)

def nand_and(a,b):
    return  nand(nand(a,b),nand(a,b))

def nand_or(a,b):
    return nand(nand(a,a),nand(b,b))

def nand_dnf_function(str_val):
    x1 = bool(int(str_val[0]))
    not_x1 = nand_not(x1)
    x2 = bool(int(str_val[1]))
    not_x2 = nand_not(x2)
    x3 = bool(int(str_val[2]))
    not_x3 = nand_not(x3)
    x4 = bool(int(str_val[3]))
    not_x4 = nand_not(x4)
    x5 = bool(int(str_val[4]))
    not_x5 = nand_not(x5)

    z6_t = nand(not_x1,x3)
    z6 = nand(z6_t,z6_t)
    z1_t = nand(z6,x5)
    z1 = nand(z1_t,z1_t)
    z2_t = nand(not_x4,x1)
    z2 = nand(z2_t,z2_t)
    z4_t = nand(not_x1,x2)
    z4 = nand(z4_t,z4_t)
    z5_t = nand(not_x2,x1)
    z5 = nand(z5_t,z5_t)
    z3_t = nand(not_x5,x1)
    z3 = nand(z3_t,z3_t)

    #0-111
    u1 = nand(z1,x4)
```

```
    #011-1
    u2 = nand(z1,x2)
    #10-0-
    u3 = nand(z2,not_x2)
    #1-00-
    u4 = nand(z2,not_x3)
    #01-1-
    u5 = nand(z4,x4)
    #100--
    u6 = nand(z5,not_x3)
    #10--0
    u7 = nand(z3,not_x2)
    #1--00
    u8 = nand(z3,not_x4)

    v1 = nand(u1,u2)
    not_v1 = nand(v1,v1)
    v2 = nand(u3,u4)
    not_v2 = nand(v2,v2)
    v3 = nand(u5,u6)
    not_v3 = nand(v3,v3)
    v4 = nand(u7,u8)
    not_v4 = nand(v4,v4)
    v5 = nand(not_v1,not_v2)
    not_v5 = nand(v5,v5)
    v6 = nand(not_v3,not_v4)
    not_v6 = nand(v6,v6)
    v7 = nand(not_v5,not_v6)
    return v7

def nor(a,b):
    return not(a or b)

def nor_not(a):
    return nor(a,a)

def nor_or(a,b):
    return  nor(nor(a,b),nor(a,b))

def nor_and(a,b):
    return nor(nor(a,a),nor(b,b))

def nor_dnf_function(str_val):
    x1 = bool(int(str_val[0]))
    not_x1 = nor(x1,x1)
    x2 = bool(int(str_val[1]))
    not_x2 = nor(x2,x2)
    x3 = bool(int(str_val[2]))
    not_x3 = nor(x3,x3)
    x4 = bool(int(str_val[3]))
    not_x4 = nor(x4,x4)
    x5 = bool(int(str_val[4]))
    not_x5 = nor(x5,x5)

    z6 = nor(x1,not_x3)
    not_z6 = nor(z6,z6)
    z1 = nor(not_z6,not_x5)
    not_z1 = nor(z1,z1)
    z2 = nor(x4,not_x1)
    not_z2 = nor(z2,z2)
```

```
    z4 = nor(x1,not_x2)
    not_z4 = nor(z4,z4)
    z5 = nor(x2,not_x1)
    not_z5 = nor(z5,z5)
    z3 = nor(not_x1,x5)
    not_z3 = nor(z3,z3)

    #0-111
    u1 = nor(not_z1,not_x4)
    #011-1
    u2 = nor(not_z1,not_x2)
    #10-0-
    u3 = nor(not_z2,x2)
    #1-00-
    u4 = nor(not_z2,x3)
    #01-1-
    u5 = nor(not_z4,not_x4)
    #100--
    u6 = nor(not_z5,x3)
    #10--0
    u7 = nor(not_z3,x2)
    #1--00
    u8 = nor(not_z3,x4)

    v1_t = nor(u1,u2)
    v1 = nor(v1_t,v1_t)
    v2_t = nor(u3,u4)
    v2 = nor(v2_t,v2_t)
    v3_t = nor(u5,u6)
    v3 = nor(v3_t,v3_t)
    v4_t = nor(u7,u8)
    v4 = nor(v4_t,v4_t)
    v5_t = nor(v1,v2)
    v5 = nor(v5_t,v5_t)
    v6_t = nor(v3,v4)
    v6 = nor(v6_t,v6_t)
    v7_t = nor(v5,v6)
    v7 = nor(v7_t,v7_t)
    return v7

def nor_knf_function(str_val):
    x1 = bool(int(str_val[0]))
    not_x1 = nor(x1,x1)
    x2 = bool(int(str_val[1]))
    not_x2 = nor(x2,x2)
    x3 = bool(int(str_val[2]))
    not_x3 = nor(x3,x3)
    x4 = bool(int(str_val[3]))
    not_x4 = nor(x4,x4)
    x5 = bool(int(str_val[4]))
    not_x5 = nor(x5,x5)

    z6_t = nor(not_x5,not_x3)
    z6 = nor(z6_t,z6_t)
    z1_t = nor(z6,not_x1)
    z1 = nor(z1_t,z1_t)
    z2_t = nor(x1,x2)
    z2 = nor(z2_t,z2_t)
    z4_t = nor(x1,x4)
    z4 = nor(z4_t,z4_t)
```

```
    z5_t = nor(not_x1,not_x4)
    z5 = nor(z5_t,z5_t)
    z3_t = nor(x1,x5)
    z3 = nor(z3_t,z3_t)

    #1-111
    u1 = nor(z1,not_x4)
    #111-1
    u2 = nor(z1,not_x2)
    #000--
    u3 = nor(z2,x3)
    #00-0-
    u4 = nor(z2,x4)
    #0-00-
    u5 = nor(z4,x3)
    #11-1-
    u6 = nor(z5,not_x2)
    #00--0
    u7 = nor(z3,x2)
    #0--00
    u8 = nor(z3,x4)

    v1 = nor(u1,u2)
    not_v1 = nor(v1,v1)
    v2 = nor(u3,u4)
    not_v2 = nor(v2,v2)
    v3 = nor(u5,u6)
    not_v3 = nor(v3,v3)
    v4 = nor(u7,u8)
    not_v4 = nor(v4,v4)
    v5 = nor(not_v1,not_v2)
    not_v5 = nor(v5,v5)
    v6 = nor(not_v3,not_v4)
    not_v6 = nor(v6,v6)
    v7 = nor(not_v5,not_v6)
    return v7

def nand_knf_function(str_val):
    x1 = bool(int(str_val[0]))
    not_x1 = nand(x1,x1)
    x2 = bool(int(str_val[1]))
    not_x2 = nand(x2,x2)
    x3 = bool(int(str_val[2]))
    not_x3 = nand(x3,x3)
    x4 = bool(int(str_val[3]))
    not_x4 = nand(x4,x4)
    x5 = bool(int(str_val[4]))
    not_x5 = nand(x5,x5)

    z6 = nand(x5,x3)
    not_z6 = nand(z6,z6)
    z1 = nand(not_z6,x1)
    not_z1 = nand(z1,z1)
    z2 = nand(not_x1,not_x2)
    not_z2 = nand(z2,z2)
    z4 = nand(not_x1,not_x4)
    not_z4 = nand(z4,z4)
    z5 = nand(x1,x4)
    not_z5 = nand(z5,z5)
    z3 = nand(not_x1,not_x5)
```

```
    not_z3 = nand(z3,z3)

    #1-111
    u1 = nand(not_z1,x4)
    #111-1
    u2 = nand(not_z1,x2)
    #000--
    u3 = nand(not_z2,not_x3)
    #00-0-
    u4 = nand(not_z2,not_x4)
    #0-00-
    u5 = nand(not_z4,not_x3)
    #11-1-
    u6 = nand(not_z5,x2)
    #00--0
    u7 = nand(not_z3,not_x2)
    #0--00
    u8 = nand(not_z3,not_x4)

    v1_t = nand(u1,u2)
    v1 = nand(v1_t,v1_t)
    v2_t = nand(u3,u4)
    v2 = nand(v2_t,v2_t)
    v3_t = nand(u5,u6)
    v3 = nand(v3_t,v3_t)
    v4_t = nand(u7,u8)
    v4 = nand(v4_t,v4_t)
    v5_t = nand(v1,v2)
    v5 = nand(v5_t,v5_t)
    v6_t = nand(v3,v4)
    v6 = nand(v6_t,v6_t)
    v7_t = nand(v5,v6)
    v7 = nand(v7_t,v7_t)
    return v7

def function(str_val):
    x1 = str_val[0]
    x2 = str_val[1]
    x3 = str_val[2]
    x4 = str_val[3]
    x5 = str_val[4]
    return (
        3 < (int(str_val[3] + str_val[4],2) + int(str_val[0] + str_val[1] + str_val[2],2)) < 8
    )

table_head = ["№","$x_1x_2x_3x_4x_5$","исходная функция","днф базис И-НЕ","днф базис
↪  ИЛИ-НЕ","кнф базис И-НЕ","кнф базис ИЛИ-НЕ"]
table =
↪  truth_table(gen_bin_vector(),function,nand_dnf_function,nor_dnf_function,nand_knf_function,nor_knf_
print(tabulate(table,table_head,tablefmt="simple"))
```