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 №7
дисциплина «Теория цифровых автоматов»
по теме «Диагностика неисправностей многовыходных комбинационных схем»

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

Белгород 2019

Лабораторная работа №7

«Диагностика неисправностей многовыходных комбинационных схем»

Цель работы:научиться строить диагностические тесты и алгоритмы распознавания неисправностей многовыходных комбинационных схем..

Вариант 9

Задание:

1. Представить в аналитической форме систему булевых функций,реализуемую исправной комбинационной схемой.
2. Представить в аналитической форме систему булевых функций для каждой неисправности.
3. Представить в аналитической форме разностные функции для каждой неисправности.
4. Написать программу, которая строит диагностическую матрицу на основе аналитического представления разностных функций.
5. По диагностической матрице найти минимальный диагностический тест.
6. Написать программу, которая по аналитическому представлению систем булевых функций неисправностей строит таблицу их значений на тестовых наборах. Для компактного представления таблицы рекомендуется значение системы булевых функций неисправности представлять не двоичным вектором, а десятичным числом.
7. По полученной в п. 6 таблице построить алгоритм распознавания неисправностей в виде диагностического дерева.
8. Определить неисправность, которая распознается дольше других,написать программу, которая моделирует схему с этой неисправностью, провести диагностический эксперимент.

Ход работы

№	$x_1x_2x_3x_4x_5$	f1	f2	f3	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F1
1	0000	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	
2	0001	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	
3	0010	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	
4	0011	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	
5	0100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0110	1	1	1	1	1	1	0	0	1	0	0	0	1	1	1	

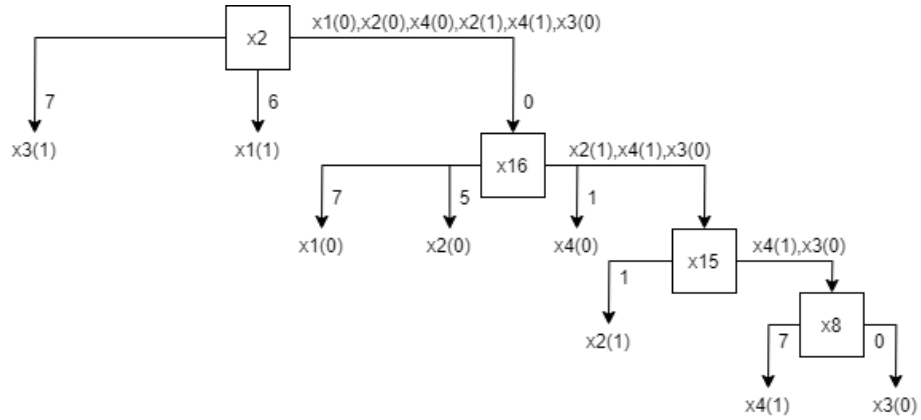
№	$x_1x_2x_3x_4x_5$	f1	f2	f3	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F1
8	0111	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	
9	1000	1	1	0	0	0	0	1	1	0	1	1	0	0	0	0	
10	1001	1	1	0	0	0	0	1	1	0	1	1	0	0	0	0	
11	1010	1	1	1	0	0	0	1	1	1	1	1	1	0	0	1	
12	1011	1	0	1	1	1	1	1	0	1	1	0	1	0	0	0	
13	1100	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
14	1101	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
15	1110	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	
16	1111	0	0	0	1	1	1	0	0	0	1	0	1	0	0	0	

Необнаружимых и недиагностируемых неисправностей нет.

№	f1	f2	f3	||	R1	R2	R3	R4	R5	R6	R7	R8
1	0	0	0	||	0	1	0	0	0	0	0	0
2	0	0	0	||	0	1	0	0	0	1	1	0
3	0	0	0	||	0	1	1	1	1	0	0	1
4	1	1	1	||	0	1	0	0	1	1	1	1
5	0	0	0	||	0	0	0	0	0	1	1	0
6	0	0	0	||	0	0	0	0	0	1	1	0
7	1	1	1	||	0	1	1	1	1	1	0	0
8	1	1	1	||	0	1	1	0	1	1	0	0
9	1	1	0	||	1	1	1	1	1	1	1	1
10	1	1	0	||	1	1	1	1	1	1	1	1
11	1	1	1	||	1	1	0	1	0	1	0	1
12	1	0	1	||	1	1	1	1	1	1	1	1
13	0	0	0	||	0	0	1	0	0	0	1	0
14	0	0	0	||	0	0	1	0	0	0	0	0
15	0	0	1	||	1	1	1	1	1	1	1	1
16	0	0	0	||	1	1	1	1	0	0	0	1

№	f1	f2	f3	||	R1	R2	R3	R4	R5	R6	R7	R8
2	0	0	0	||	0	1	0	0	0	1	1	0
3	0	0	0	||	0	1	1	1	1	0	0	1
4	1	1	1	||	0	1	0	0	1	1	1	1
8	1	1	1	||	0	1	1	0	1	1	0	0
15	0	0	1	||	1	1	1	1	1	1	1	1
16	0	0	0	||	1	1	1	1	0	0	0	1

№	F	F1	F2	F3	F4	F5	F6	F7	F8
2	0	0	6	0	0	0	7	0	0
3	0	0	7	0	7	0	0	0	7
4	7	7	5	7	7	0	7	1	7
8	7	7	0	7	7	0	7	7	7
15	1	7	1	7	1	0	1	1	0
16	7	7	0	5	0	0	0	1	0



Приложение

Содержимое файла funcTest.py

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

args_var = ['x1','x2','x3','x4']
args_f = ['f1','f2','f3']
args_val = [False,True]
table_head = ["№","$x_1x_2x_3x_4x_5$",'f1','f2','f3']

def truth_table(vector,var,val):
    result = []
    F_count = 1
    for i in range(0,len(vector)):
        args = vector[i][0]
        sch = test_schema(args)
        result_item = [
                                str(int(bad_schema(args,var,val,'f1'))),
                                str(int(bad_schema(args,var,val,'f2'))),
                                str(int(bad_schema(args,var,val,'f3'))),
        ]
        result.append(result_item)
    return result

def bad_truth_table(vector,f,var,val,f_n):
    result = []
    for i in range(0,len(vector)):
        result.append([
            i+1,
            vector[i][0],
            int(f(vector[i][0],var,val,f_n)),
        ])
    return result

def f_v9(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(x4 + x5,2) + int(x1 + x2 + x3,2)) < 8
    )

def f_v10(str_val):
    x1 = str_val[0]
    x2 = str_val[1]
    x3 = str_val[2]
    x4 = str_val[3]
    x5 = str_val[4]
    return (
        4 <= (int(x4 + x5,2) + int(x1 + x2 + x3,2)) <= 6
    )

def f_v11(str_val):
    x1 = str_val[0]
    x2 = str_val[1]
    x3 = str_val[2]
```

```
    x4 = str_val[3]
    x5 = str_val[4]
    return (
        5 <= (int(x2 + x3,2) + int(x4 + x5 + x1,2)) <= 8
    )

def f_v12(str_val):
    x1 = str_val[0]
    x2 = str_val[1]
    x3 = str_val[2]
    x4 = str_val[3]
    x5 = str_val[4]
    return (
        -2 <= (int(x1 + x2,2) - int(x3 + x4 + x5,2)) <= 1
    )

def f(f1,f2,str_val):
    if not f2(str_val):
        return '-'
    else:
         return str(int(f1(str_val)))

def test_schema(str_val):
    x1 = bool(int(str_val[0]))
    x2 = bool(int(str_val[1]))
    x4 = bool(int(str_val[2]))
    x5 = bool(int(str_val[3]))

    z1 = not x1 and x4
    z2 = not x2 and x1
    z3 = x1 and not x5
    z4 = not x2 and not x4
    z5 = x4 and not x2
    z6 = x2 and x4

    u1 = z1 and x2
    u2 = z1 and x5
    u3 = z4 and x1
    u4 = z5 and x5
    u5 = z2 and x4
    u6 = z2 and not x5
    u7 = z6 and not x5
    u8 = z3 and x4
    u9 = z3 and not x4

    v3 = u1 or u2
    v4 = u4 or u5
    v6 = u3 or u6
    v8 = u7 or u8

    v1 = v3 or v6
    v2 = v3

    v5 = v1 or u3
    v7 = v4 or v2

    f1 = v4 or v5
    f2 = v1
    f3 = v7 or u8
    return (f1,f2,f3)
```

```
def bad_schema(str_val,bad_var,bad_val,num_f):
    x1 = bad_val if bad_var == 'x1' else bool(int(str_val[0]))
    x2 = bad_val if bad_var == 'x2' else bool(int(str_val[1]))
    x4 = bad_val if bad_var == 'x3' else bool(int(str_val[2]))
    x5 = bad_val if bad_var == 'x4' else bool(int(str_val[3]))

    z1 = not x1 and x4
    z2 = not x2 and x1
    z3 = x1 and not x5
    z4 = not x2 and not x4
    z5 = x4 and not x2
    z6 = x2 and x4

    u1 = z1 and x2
    u2 = z1 and x5
    u3 = z4 and x1
    u4 = z5 and x5
    u5 = z2 and x4
    u6 = z2 and not x5
    u7 = z6 and not x5
    u8 = z3 and x4
    u9 = z3 and not x4

    v3 = u1 or u2
    v4 = u4 or u5
    v6 = u3 or u6
    v8 = u7 or u8

    v1 = v3 or v6
    v2 = v3

    v5 = v1 or u3
    v7 = v4 or v2

    f1 = v4 or v5
    f2 = v1
    f3 = v7 or u8
    if num_f == 'f1':
        return f1
    if num_f == 'f2':
        return f2
    if num_f == 'f3':
        return f3
    return (f1,f2,f3)

def args_to_num(f1,f2,f3):
    return int(f1+f2+f3,2)

def diag_test(var,val):
    table = truth_table(gen_bin_vector(),var,val)
    for i in table:
        i.append(args_to_num(i[0],i[1],i[2]))
    print("Внесем набор X2")
    print(table[1])
    if(table[1][3] == 7):
        print("Обнаружена неисправность x3(1)")
    elif(table[1][3] == 6):
        print("Обнаружена неисправность x1(1)")
    else:
```

```
        print("Возможные неисправности x1(0),x2(0),x4(0),x2(1),x4(1),x3(0)")
        print("Внесем набор X16")
        print(table[15])
        if(table[15][3] == 7):
            print("Обнаружена неисправность x1(0)")
        elif(table[15][3] == 5):
            print("Обнаружена неисправность x2(0)")
        elif(table[15][3] == 1):
            print("Обнаружена неисправность x4(0)")
        else:
            print("Возможные неисправности x2(1),x4(1),x3(0)")
            print("Внесем набор X15")
            print(table[14])
            if(table[14][3] == 1):
                print("Обнаружена неисправность x2(1)")
            else:
                print("Возможные неисправности x4(1),x3(0)")
                print("Внесем набор X8")
                print(table[7])
                if(table[7][3] == 1):
                    print("Обнаружена неисправность x4(1)")
                else:
                    print("Обнаружена неисправность x3(0)")


for var in args_var:
    for val in args_val:
        print("Неисправность "+var+"("+str(int(val))+")")
        diag_test(var,val)
        print()
```