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)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

Дисциплина: Теория цифровых автоматов

по теме Диагностика неисправностей комбинационных схем с одним выходом

Выполнил: ст. группы ВТ-32
Воскобойников И. С.

Проверил: Рязанов Ю. Д.

Белгород 2020

Цель работы: научиться строить диагностические тесты и алгоритмы распознавания неисправностей комбинационных схем с одним выходом.

З а д а н и е

При выполнении лабораторной работы нужно решить следующую задачу.

Дано:

1) комбинационная схема с одним выходом, построенная при выполнении лабораторной работы № 3;

2) множество одиночных неисправностей, состоящее из неисправностей «константа 0» и «константа 1» на каждом входе схемы.

Найти: диагностический тест для заданного множества неисправностей.

Построить: алгоритм распознавания неисправностей.

Для решения задачи нужно выполнить следующие задания.

1. Написать программу моделирования исправной схемы и построить таблицу истинности булевой функции, реализуемой исправной комбинационной схемой.

2. Для каждой неисправности написать программу моделирования схемы с этой неисправностью и построить таблицу истинности функции неисправности.

3. Определить, существуют ли в множестве неисправностей необнаружимые и неразличимые неисправности.

4. Составить матрицу функций неисправностей, содержащей попарно различные строки. Столбцы матрицы соответствуют наборам входных сигналов, а строки — векторам-значений функций неисправности. Каждой строке матрицы поставить в соответствие множество подозреваемых неисправностей.

5. Составить диагностическую матрицу, заменив в матрице функций неисправностей каждую функцию неисправности соответствую-щей разностной функцией.

6. По диагностической матрице найти минимальный диагностический тест.

7. В матрице функций неисправностей (см. п. 4) оставить только столбцы, соответствующие наборам входных сигналов, принадлежащим диагностическому тесту.

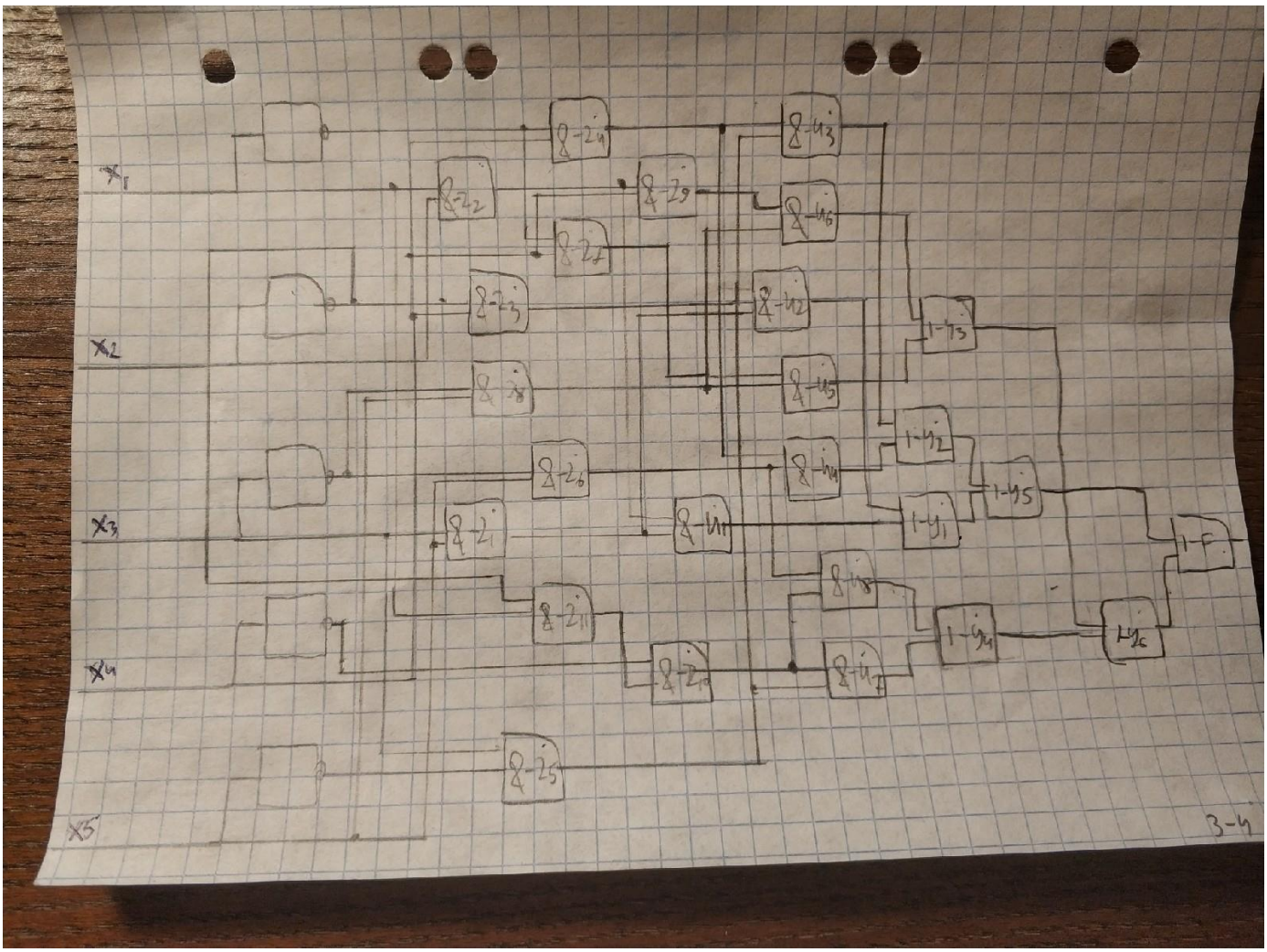
8. По полученной в п. 7 матрице построить алгоритм распознавания неисправностей в виде диагностического дерева.

9. Используя программу моделирования комбинационной схемы с неисправностью и алгоритм распознавания неисправностей написать программу для проведения диагностического эксперимента.

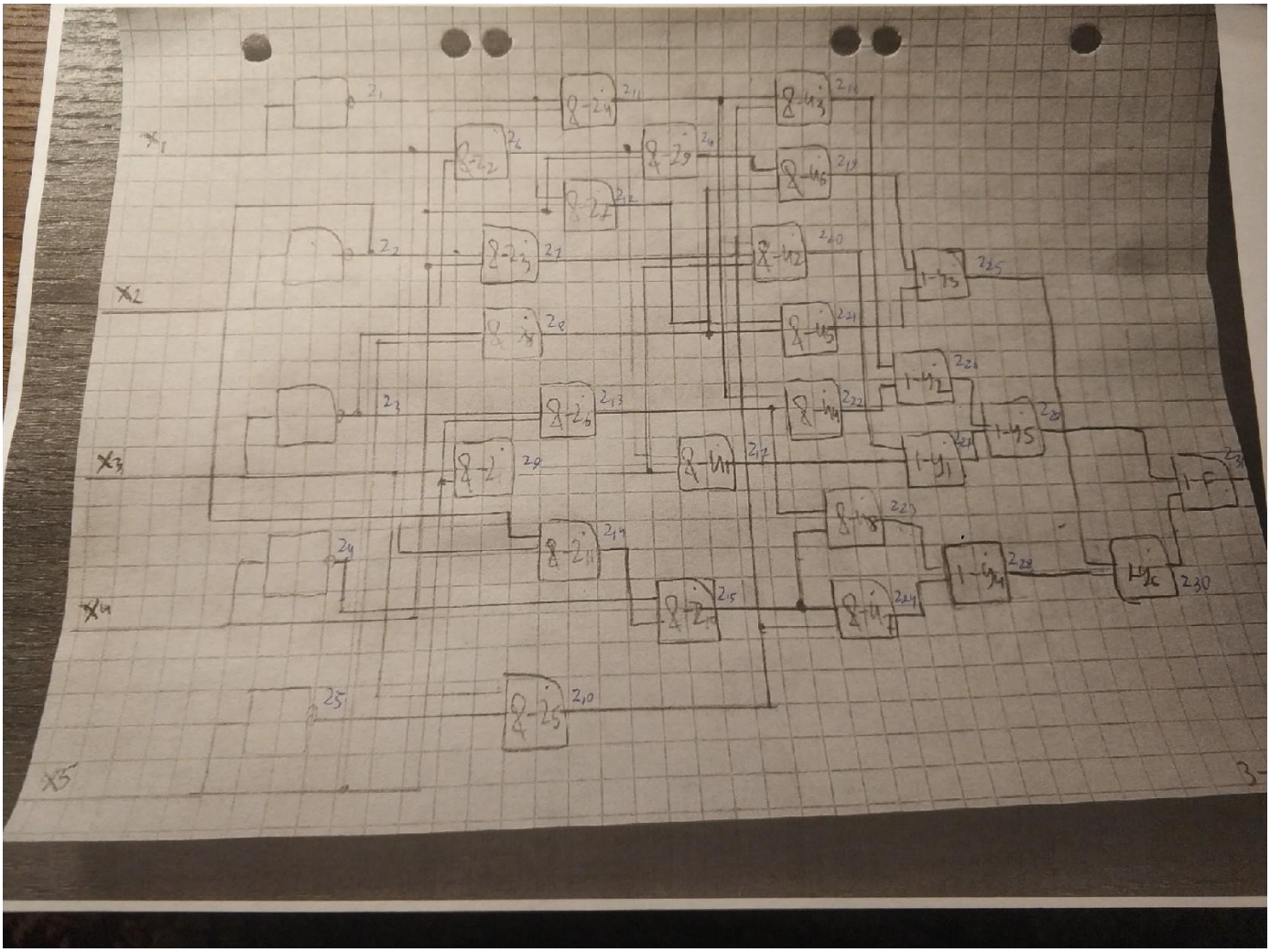
Задание 1: таблица истинности исправной схемы и программа, моделирующая ее работу

```
bool FuncDNF(bool *x)
{
    bool z1 = x[2] && x[4],
         z2 = x[0] && x[1],
         z3 = !x[1] && x[3],
         z4 = !x[0] && x[3],
         z5 = x[2] && !x[4],
         z6 = !x[2] && x[4],
         z7 = !x[0] && !x[1],
         z8 = !x[2] && !x[4],
         z9 = x[3] && z2,
         z11 = x[0] && !x[1],
         z10 = !x[3] && z11;

    bool u1 = z1 && z2,
         u2 = z1 && z3,
         u3 = z4 && z5,
         u4 = z4 && z6,
         u5 = z7 && z8,
         u6 = z8 && z9,
         u7 = z5 && z10,
         u8 = z6 && z10;

    bool y1 = u1 || u2,
         y2 = u3 || u4,
         y3 = u5 || u6,
         y4 = u7 || u8,
         y5 = y1 || y2,
         y6 = y3 || y4;



    return y5|| y6;
}
```

x1	x2	x3	x4	x5	F
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	0
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	1
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	1


x1
x2
x3
x4
x5
3-4

Задание 2: для каждой неисправности написать программу моделирования схемы с этой неисправностью и построить таблицу истинности функции неисправности.



```
bool function(bool *x, int xn, bool flag)
{
    int tx;
    if (xn != 0)
    {
        tx = x[xn - 1];
        if (flag)
            x[xn - 1] = 1;
        else
            x[xn - 1] = 0;
    }
    bool z1 = x[2] && x[4],
         z2 = x[0] && x[1],
         z3 = !x[1] && x[3],
         z4 = !x[0] && x[3],
         z5 = x[2] && !x[4],
         z6 = !x[2] && x[4],
         z7 = !x[0] && !x[1],
```

```
            z8 = !x[2] && !x[4],
            z9 = x[3] && z2,
            z11 = x[0] && !x[1],
            z10 = !x[3] && z11;

    bool u1 = z1 && z2,
            u2 = z1 && z3,
            u3 = z4 && z5,
            u4 = z4 && z6,
            u5 = z7 && z8,
            u6 = z8 && z9,
            u7 = z5 && z10,
            u8 = z6 && z10;

    bool y1 = u1 || u2,
            y2 = u3 || u4,
            y3 = u5 || u6,
            y4 = u7 || u8,
            y5 = y1 || y2,
            y6 = y3 || y4;



    if (xn != 0)
        x[xn - 1] = tx;
    return y5 || y6;
}

void output(int m, int n, bool **table, bool *fres, bool **f)
{
    for (int i = 0; i < m; i++)
    {
        cout << "x" << i + 1 << "  ";
    }
    cout << "f\t";
    for (int i = 0; i < 10; i++)
    {
        cout << "f" << i + 1 << "\t";
    }
    cout << "\n";
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << table[i][j] << "   ";
        }
        cout << fres[i] << "\t";

        cout << function(table[i], 1, false) << "\t";
        cout << function(table[i], 1, true) << "\t";
        cout << function(table[i], 2, false) << "\t";
        cout << function(table[i], 2, true) << "\t";
        cout << function(table[i], 3, false) << "\t";
        cout << function(table[i], 3, true) << "\t";
        cout << function(table[i], 4, false) << "\t";
        cout << function(table[i], 4, true) << "\t";
        cout << function(table[i], 5, false) << "\t";
        cout << function(table[i], 5, true) << "\t";
        cout << "\n";
    }
    cout << check_test(2, true);
    cout << "\n";
}
```

x1	x2	x3	x4	x5	f	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
0	0	0	0	0	1	1	0	1	0	1	0	1	1	1	0
0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	0
0	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1
0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1
0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	1	1	0	0	1	0	0	0	1
0	1	0	1	1	1	1	0	1	1	1	0	0	1	0	1
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	1	0	1	1	0	1	0	1	1	0
0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0
1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1
1	0	0	0	1	1	0	1	1	0	1	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	0	1	1	0	0	1	1	0	1	0
1	0	1	0	1	0	0	0	0	1	1	0	0	1	1	0
1	0	1	1	0	0	1	0	0	0	0	0	1	0	0	1
1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0
1	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0
1	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0
1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1
1	1	1	0	1	1	0	1	0	1	0	1	1	1	0	1
1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1

Задание 3: Определить, существуют ли в множестве неисправностей необнаружимые и неразличимые неисправности.
В данной схеме необнаружимых неисправностей нет, так как ни одна из функций неисправностей, не совпадает с исходной функцией.
Неразличимых неисправностей также нет, так как ни одна функция неисправности не совпадает с другой функцией неисправности.

Задание 4: Составить матрицу функций неисправностей, содержащей попарно различные строки. Столбцы матрицы соответствуют наборам входных сигналов, а строки — векторам-значений функций неисправности. Каждой строке матрицы поставить в соответствие множество подозреваемых неисправностей.

x1	x2	x3	x4	x5	f	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
0	0	0	0	0	1	1	0	1	0	1	0	1	1	1	0
0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	0
0	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1
0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1
0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	1	1	0	0	1	0	0	0	1
0	1	0	1	1	1	1	0	1	1	1	0	0	1	0	1
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	1	0	1	1	0	1	0	1	1	0
0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0
1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1
1	0	0	0	1	1	0	1	1	0	1	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	0	1	1	0	0	1	1	0	1	0
1	0	1	0	1	0	0	0	0	1	1	0	0	1	1	0
1	0	1	1	0	0	1	0	0	0	0	0	1	0	0	1
1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0
1	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0
1	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0
1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1
1	1	1	0	1	1	0	1	0	1	0	1	1	1	0	1
1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1

Задание 5: Составить диагностическую матрицу, заменив в матрице функций неисправностей каждую функцию неисправности соответствующей разностной функцией.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
x0	0	1	0	1	0	1	0	0	0	1
x1	0	1	0	0	0	0	0	1	1	0
x2	0	1	0	1	0	0	0	0	0	0
x3	0	1	0	0	0	0	1	0	0	0
x4	0	1	0	0	1	0	0	1	0	0
x5	0	0	0	0	0	0	0	1	0	0
x6	0	1	0	0	0	0	1	0	0	0
x7	0	0	0	1	0	0	1	0	0	0
x8	0	0	1	0	0	0	0	0	0	0
x9	0	0	0	0	0	0	0	1	0	0
x10	0	1	1	0	0	1	0	0	0	1
x11	0	1	0	0	0	1	1	0	1	0
x12	0	0	0	0	0	0	0	1	0	0
x13	0	1	0	0	0	0	0	0	0	0
x14	0	1	0	0	1	0	1	0	0	1
x15	0	1	1	0	1	0	0	0	1	0
x16	1	0	0	0	0	1	0	0	0	1
x17	1	0	0	1	0	1	0	1	1	0
x18	1	0	0	1	0	0	0	0	0	0
x19	1	0	0	0	0	1	1	0	0	0
x20	1	0	0	1	1	0	0	1	0	1
x21	0	0	0	1	1	0	0	1	1	0
x22	1	0	0	0	0	0	1	0	0	1
x23	0	0	0	0	1	0	1	0	1	0
x24	0	0	0	0	0	0	0	1	0	0
x25	0	0	1	0	0	1	0	0	0	0
x26	1	0	1	0	0	1	1	0	0	1
x27	1	0	0	0	0	1	0	0	1	0
x28	0	0	1	0	0	0	0	0	0	1
x29	1	0	1	0	1	0	0	0	1	0
x30	1	0	0	0	1	0	0	0	0	1
x31	1	0	0	0	1	0	0	0	1	0

Задание 6: Составить диагностическую матрицу, заменив в матрице функций неисправностей каждую функцию неисправности соответствую-щей разностной функцией.
Проверяющий тест: { x^{14}, x^{15}, x^{17}},{ x^{11}, x^{15}, x^{20}},{ x^{0}, x^{21}, x^{17}}
Минимальный диагностический тест: { x^{0}, x^{1} , $x^{2,}$ x^{14}, x^{15}, x^{17}}

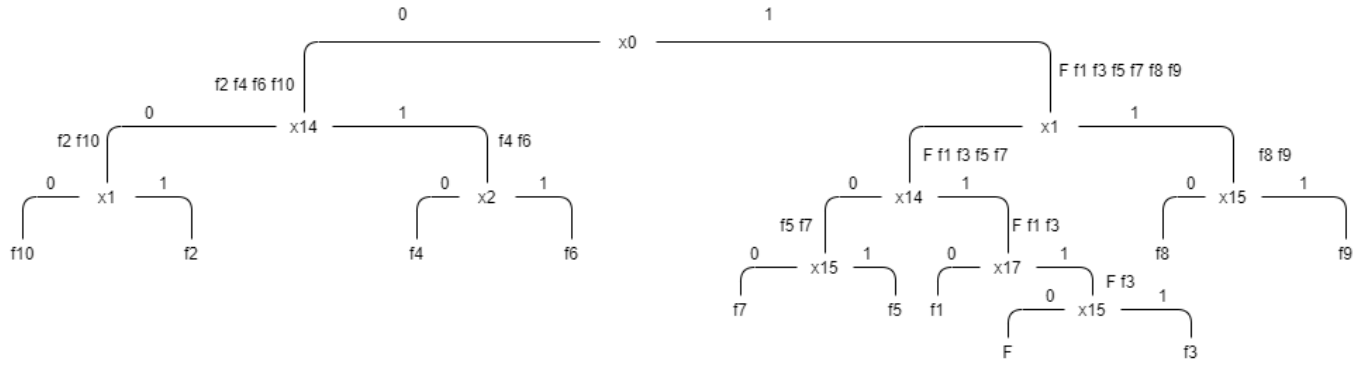
Диагностический тест

Номер набора	Разностные функции									
	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
X^{0}	0	1	0	1	0	1	0	0	0	1
x^{1}	0	1	0	0	0	0	0	1	1	0
X^{2}	0	1	0	1	0	0	0	0	0	0
X^{14}	0	1	0	0	1	0	1	0	0	1
X^{15}	0	1	1	0	1	0	0	0	1	0
X^{17}	1	0	0	1	0	1	0	1	1	0

Задание 7: В матрице функций неисправностей (см. п. 4) оставить только столбцы, соответствующие наборам входных сигналов, принадлежащим диагностическому тесту.

Номер набора	Значение сигналов					Разностные функции										
	X1	X2	X3	X4	X5	F	f1	f2	f3	f4	f5	f6	f7	f8	f9	F10
X^{0}	0	0	0	0	0	1	1	0	1	0	1	0	1	1	1	0
X^{1}	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	0
X^{2}	0	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1
X^{14}	0	1	1	1	0	1	1	0	1	1	0	1	0	1	1	0
X^{15}	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0
X^{17}	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	1

Задание 8: По полученной в п. 7 матрице построить алгоритм распознавания неисправностей в виде диагностического дерева.



9. Используя программу моделирования комбинационной схемы с неисправностью и алгоритм распознавания неисправностей написать программу для проведения диагностического эксперимента.

```
string check_test(int xn, bool flag) {
    bool test[6][5] = {{0, 0, 0, 0, 0},
                       {0, 0, 0, 0, 1},
                       {0, 0, 0, 1, 0},
                       {0, 1, 1, 1, 0},
                       {0, 1, 1, 1, 1},
                       {1, 0, 0, 0, 0}};
    if (function(test[0], xn, flag)) {
        if (function(test[1], xn, flag)) {
            if (function(test[4], xn, flag)) {
                return "error f9 (x5 = 0)";
            } else {
                return "error f8 (x4 = 1)";
            }
        } else {
            if (function(test[3], xn, flag)) {
                if (function(test[5], xn, flag)) {
                    if (function(test[4], xn, flag)) {
                        return "error f3 (x2 = 0)";
                    } else {
                        return "error F";
                    }
                } else {
                    return "error f1 (x1 = 0)";
                }
            } else {
                if (function(test[4], xn, flag)) {
                    return "error f5 (x3 = 0)";
                } else {
                    return "error f7 (x4 = 0)";
                }
            }
        }
    } else {
        if (function(test[3], xn, flag)) {
            if (function(test[2], xn, flag)) {
                return "error f6 (x3 = 1)";
            } else {
                return "error f4 (x2 = 1)";
            }
        } else {
            if (function(test[1], xn, flag)) {
                return "error f2 (x1 = 1)";
            } else {
                return "error f10 (x5 = 1)";
            }
        }

    }
}
```

x1	x2	x3	x4	x5	f	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
0	0	0	0	0	1	1	0	1	0	1	0	1	1	1	0
0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	0
0	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1
0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1
0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	1	1	0	0	1	0	0	0	1
0	1	0	1	1	1	1	0	1	1	1	0	0	1	0	1
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	1	0	1	1	0	1	0	1	1	0
0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0
1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1
1	0	0	0	1	1	0	1	1	0	1	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	0	1	1	0	0	1	1	0	1	0
1	0	1	0	1	0	0	0	0	1	1	0	0	1	1	0
1	0	1	1	0	0	1	0	0	0	0	0	1	0	0	1
1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0
1	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0
1	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0
1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1
1	1	1	0	1	1	0	1	0	1	0	1	1	1	0	1
1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	1
1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	1

error f6 (x3 = 1)