**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-31  
Новожен Н.В

Проверил: Рязанов.Ю.Д

**Белгород 2019**

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

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.

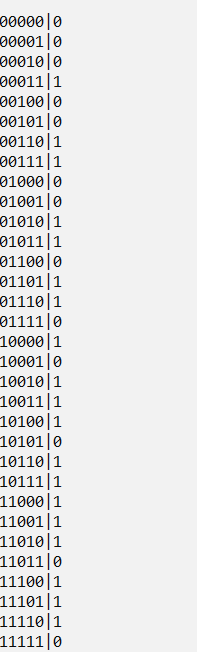
5 Составить диагностическую матрицу, заменив в матрице функций неисправностей каждую функцию неисправности соответствующей разностной функцией.

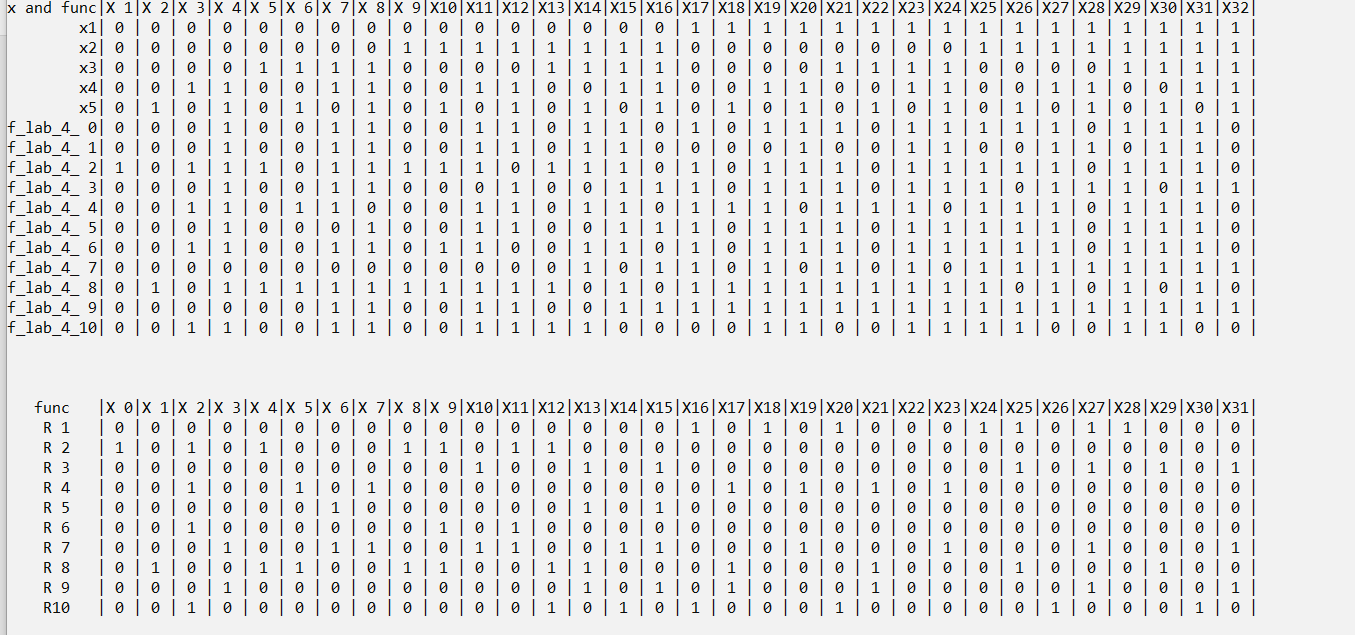
6 По диагностической матрице найти минимальный диагностический тест.

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.





Неразличимых и не обнаружимых неисправностей нет.

По диагностической матрице найти минимальный диагностический тест.

typedef int (\*rrr)(int, int, int, int, int);

rrr\* arr\_fun = nullptr;

void show\_step3\_lab4(int\*\* a, int m, int n)

{

std::cout << " func |";

for (int i = 0; i < n; ++i)

{

std::cout << "X";

std::cout.width(2);

std::cout << i << "|";

}

std::cout << "\n";

int t = 0;

for (int i = 0; i < m; ++i)

{

std::cout << " R"; std::cout.width(2); std::cout << i+1;

std::cout << " | ";

for (int j = 0; j < n; ++j)

{

std::cout << (a[i][j] == 1 ? 1 : 0) << " | ";

}

std::cout << "\n";

}

std::cout << "\n\n\n";

}

void show\_step2\_lab4(int\*\* a, int m, int n,int count\_x)

{

std::cout << "x and func|";

for (int i = 0; i < n; ++i)

{

std::cout << "X";

std::cout.width(2);

std::cout << i+1-1 << "|";

}

std::cout << "\n";

int t = 0;

for (int i = 0; i < m; ++i)

{

if (i < count\_x) {

std::cout.width(9); std::cout << "x" << i+1;}

else { std::cout << "f\_lab\_4\_"; std::cout.width(2); std::cout << t++; }

std::cout << "| ";

for (int j = 0; j < n; ++j)

{

std::cout << (a[i][j] == 1 ? 1 : 0) << " | ";

}

std::cout << "\n";

}

std::cout << "\n\n\n";

}

void FillArrFun(rrr\* arr\_fun, int coun)

{

arr\_fun[0] = &f\_lab\_4;

arr\_fun[1] = &f\_lab\_4\_10;

arr\_fun[2] = &f\_lab\_4\_11;

arr\_fun[3] = &f\_lab\_4\_20;

arr\_fun[4] = &f\_lab\_4\_21;

arr\_fun[5] = &f\_lab\_4\_30;

arr\_fun[6] = &f\_lab\_4\_31;

arr\_fun[7] = &f\_lab\_4\_40;

arr\_fun[8] = &f\_lab\_4\_41;

arr\_fun[9] = &f\_lab\_4\_50;

arr\_fun[10] =&f\_lab\_4\_51;

}

int\*\* viev\_lab4\_tabl1(int\*\* a, int m, int n, rrr\* arr\_fun, int coun)

{

int\*\* matr = new int\* [n + coun];

for (int i = 0; i < n + coun; ++i) {

matr[i] = new int[m];

}

for (int i = 0; i < m; ++i) {

for (int j = 0; j < n; ++j) {

matr[j][i] = a[i][j];

}

}

for (int i = 0; i < coun; ++i) {

for (int j = 0; j < m; ++j) {

matr[i+n][j] = arr\_fun[i](matr[0][j], matr[1][j], matr[2][j], matr[3][j], matr[4][j]);

}

}

return matr;

}

int Xor(int i, int j) { return (i && !j) || (!i && j); }

int\*\* step3\_lab3(int\*\* a, int m, int n, int coun)//return R size =(coun-1,m)

{

int\*\* matr = new int\* [coun-1];

for (int i = 0; i < coun-1; ++i) {

matr[i] = new int[n];

}

int t = m - coun;//f

for (int i = 0; i < coun-1; ++i) {

for (int j = 0; j < n; ++j) {

matr[i ][j] =Xor(a[t][j],a[t+i+1][j]);

}

}

return matr;

}

int\*\* get4\_lab4(int\*\* a, int m, int n,int k ,int x1, int x2, int x3, int x4, int x5, int x6, int x7, int x8, int x9, int x10)

{

int\*\* matr = new int\* [m];

for (int i = 0; i < m; ++i) {

matr[i] = new int[k];

}

int kk=0;

for (int i = 0; i < n; ++i) {

if( (kk < k)&&(x1 == i || x2 == i || x3 == i || x4 == i || x5 == i || x6 == i || x7 == i || x8 == i || x9 == i || x10==i) )

{

for (int j = 0; j < m; ++j) {

matr[j][i] = a[j][i];

}

kk++;

}

}

return matr;

}

void GetVector4(int\* vect, int pos, int\*\* matr, int m, int n)

{

static int s = 0;

for (size\_t i = 0; i < 2; i++)

{

vect[pos] = i;

if (pos == n - 1)

{

for (int j = 0; j < n; ++j)

{

matr[s][j] = vect[j];

//std::cout << vect[j];

}

//std::cout << '\n';

++s;

}

else GetVector4(vect, pos + 1, matr, m, n);

}

}

int\*\* GetTableTrue\_lab4(int m, int n)

{

int\*\* matr = new int\* [m];

for (int i = 0; i < m; ++i) {

matr[i] = new int[n +1 /\*+1+ 1 + 1 + 1 + 1\*/];

}

auto vect = new int[n];

GetVector4(vect, 0, matr, m, n);

delete[] vect;

return matr;

}

void fun\_lab4(int \*\* a,int m,int n,rrr pf) {

for (size\_t i = 0; i < m; i++)

{

a[i][n] = pf(a[i][0], a[i][1], a[i][2], a[i][3], a[i][4]);//это f1

for (int j = 0; j < n; ++j)

{

std::cout << (a[i][j] ? 1 : 0);

}

std::cout << "|" << (pf(a[i][0], a[i][1], a[i][2], a[i][3], a[i][4]) ? 1 : 0) << '\n';

}

}

std::string Is\_Error\_Lab4(int \*\*a,int m,int n)

{

std::string s;

if (a[15][n]==1)

{

if (a[6][n] == 1)

{

if (a[25][n] == 1)

return s += "f9 ||x5=0";

else

return s += "f3 ||x2=0";

}

else

{

if (a[7][n] == 1)

return s += "f5 ||x3=0";

else

return s += "f7 ||x4=0";

}

}

else

{

if (a[9][n] == 1)

{

if (a[0][n] == 1)

return s += "f2 ||x1=1";

else

{

if (a[2][n] == 1)

return s += "f6 ||x3=1";

else

return s += "f8 ||x4=1";

}

}

else

{

if (a[25][n] == 1)

{

if (a[7][n] == 1)

return s += "f10 ||x5=1";

else

return s += "f4 ||x2=1";

}

else

return s += "f1 ||x1=0";

}

}

}

void proverka( int m, int n)

{

auto mt = GetTableTrue\_lab4(m, n);

for (size\_t i = 0; i < 10; i++)

{

fun\_lab4(mt, m, n, arr\_fun[i+1]);

std::cout << "f " << i + 1 << "==?? " << Is\_Error\_Lab4(mt, m, n) << std::endl;

}

dl(mt, m);

}

void viev\_lab4(int\*\* a, int m, int n)

{

for (size\_t i = 0; i < m; i++)

{

for (int j = 0; j < n; ++j)

{

std::cout << (a[i][j] ? 1 : 0);

}

std::cout << "|" << (f\_lab\_4(a[i][0], a[i][1], a[i][2], a[i][3], a[i][4]) ? 1 : 0) << '\n';

}

int k = 11;

arr\_fun = new rrr[k];

FillArrFun(arr\_fun, k);

auto matr =viev\_lab4\_tabl1(a, m, n, arr\_fun, k);

dl(a, m);

int n1 = m;

int m1 = n+k;

show\_step2\_lab4(matr, m1, n1,n);

auto Arr = step3\_lab3(matr, m1, n1, k);

dl(matr, m1);

show\_step3\_lab4(Arr, k-1, n1);

proverka(m, n);

dl(Arr, k-1);

delete[] arr\_fun;

}

