Воскобойников

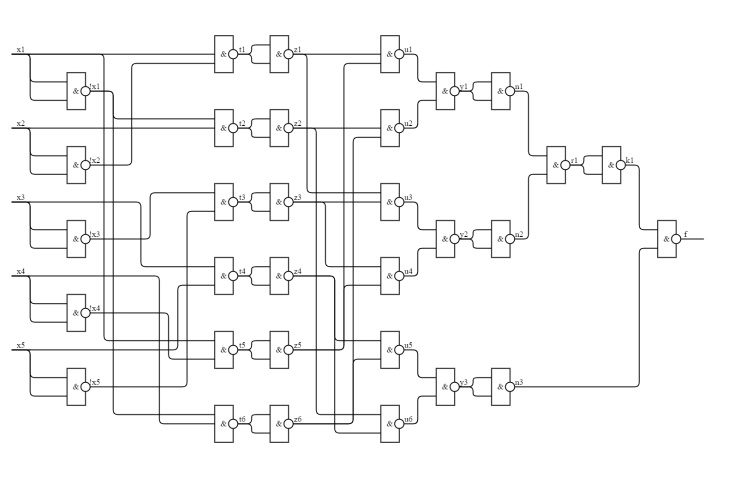
БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ  
УНИВЕРСИТЕТ им. В.Г. ШУХОВА

Кафедра *Программного обеспечения вычислительной техники и автоматизированных систем*

Дисциплина Теория цифровых автоматов

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 2**

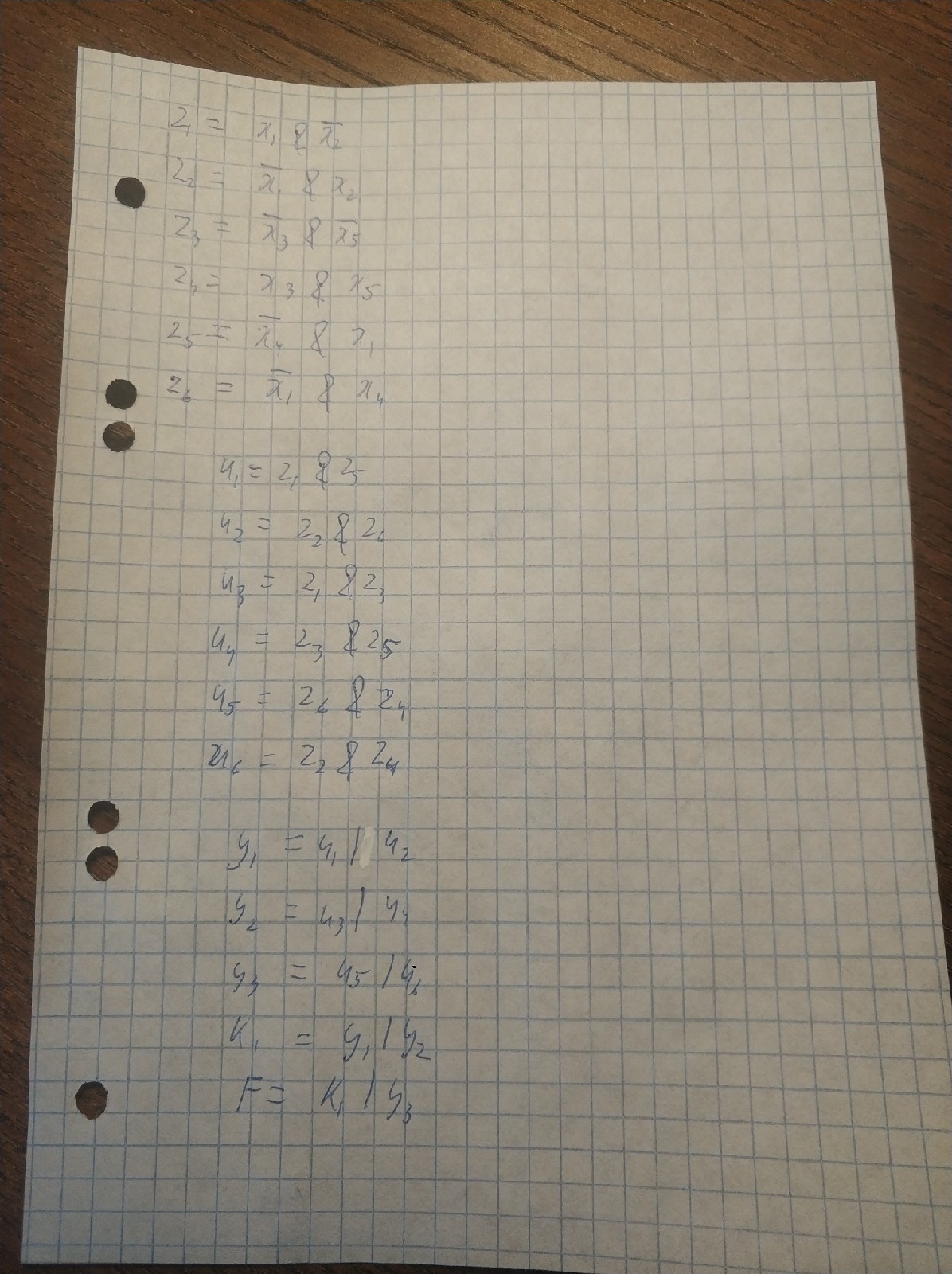
Задана схема:

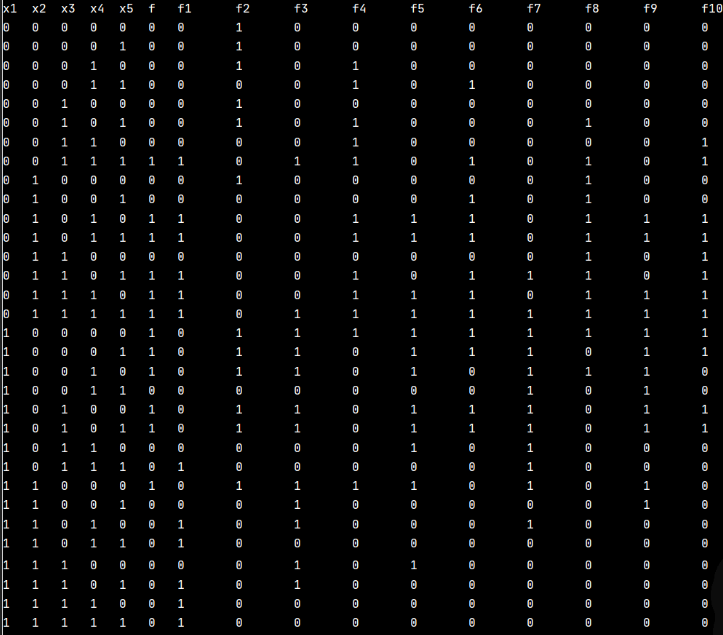


Построить диагностическое дерево для распознавания одиночных константных неисправностей на входах схемы.

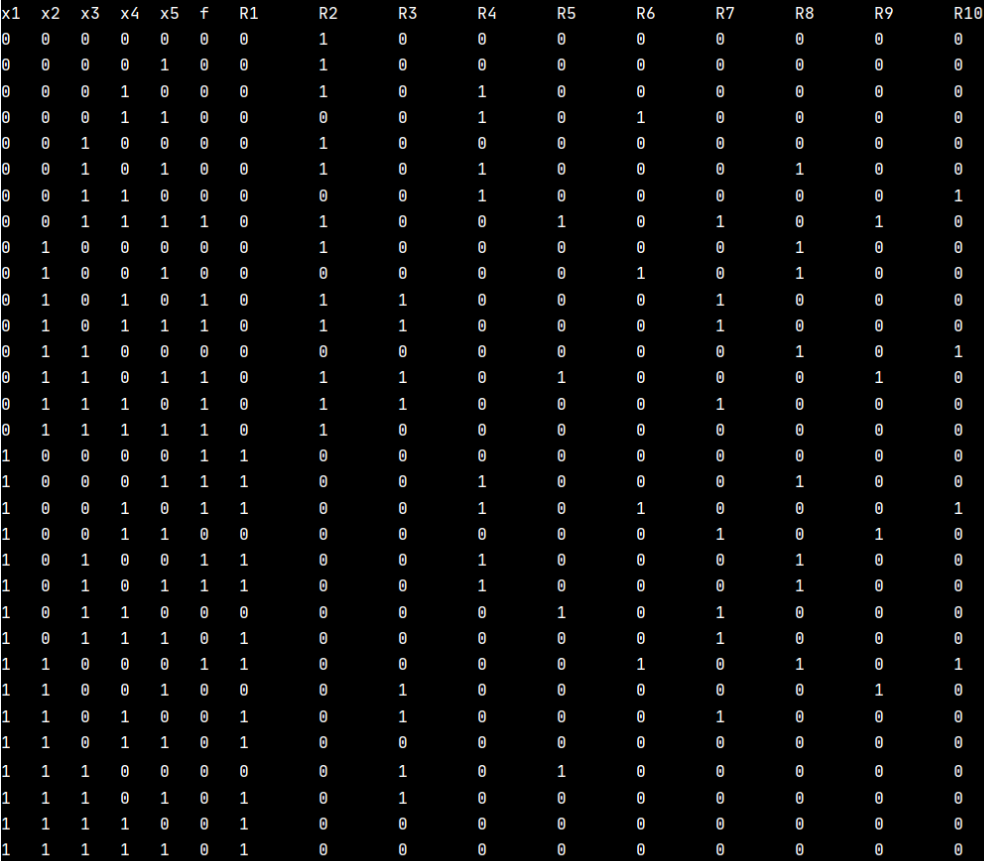
Одобрено на заседании кафедры «17» декабря 2020 г.  
протокол № 4 от 17 декабря 2020 г.

Зав. кафедрой ПОВТАС Поляков В. М.





#include <iostream>  
#include <math.h>  
#include <string>  
  
using namespace std;  
const int NotUsed = system("color F0");  
  
  
bool function(bool \*x, int xn, bool flag) *//моделирование работы схемы МДНФ*{  
 int tx;  
 if (xn != 0)  
 {  
 tx = x[xn - 1];  
 if (flag)  
 x[xn - 1] = 1;  
 else  
 x[xn - 1] = 0;  
 }  
 bool z1 = x[0] &&!x[1],  
 z2= !x[0] && x[1],  
 z3 = !x[2] && !x[4],  
 z4 = x[2] && x [4],  
 z5 = !x[3] && x[0],  
 z6 = !x[0] && x[3];  
  
 bool u1 = z1 && z5,  
 u2 = z2 && z6,  
 u3 = z1 && z3,  
 u4 = z3 && z5,  
 u5 = z6 && z4,  
 u6 = z2 && z4;  
  
 bool y1 = u1 || u2,  
 y2 = u3 || u4,  
 y3 = u5 || u6,  
 k1 = y1 || y2;  
  
  
  
 if (xn != 0)  
 x[xn - 1] = tx;  
 return k1|| y3;  
}  
  
  
  
  
  
void truth\_table(int m, int n, bool \*\*table, bool \*fres, bool \*\*f) *//функция для построения таблицы истинности*{  
 *//заполнение table двоичными векторами* for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)  
 {  
 table[i][j] = ((i >> (m - 1 - j)) & 1);  
 }  
 }  
 *//получение результата функции и запись его в f при значениях, записанных в i-той строке table* for (int i = 0; i < n; i++)  
 {  
  
  
 fres[i] = function(table[i], 0, false);  
  
 }  
}  
  
  
  
void output(int m, int n, bool \*\*table, bool \*fres)  
{  
 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 << "x"<<i<<"\t";* 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 << (function(table[i], 1, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 1, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 2, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 2, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 3, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 3, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 4, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 4, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 5, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 5, true) ^ fres[i]) << "\t";  
\*/* cout << "\n";  
 }  
 cout << check\_test(3, true)<<"\n\n";  
  
}  
  
int main()  
{  
 int m = 5, n = pow(2, m);  
 bool \*\*table = new bool\*[n];  
 for (int i = 0; i < n; i++)  
 {  
 table[i] = new bool[m];  
 }  
 bool \*fres = new bool[n];  
 int mf = 10, nf = 32;  
 bool \*\*f = new bool\*[nf];  
 for (int i = 0; i < nf; i++)  
 {  
 f[i] = new bool[mf];  
 }  
 truth\_table(m, n, table, fres, f);  
 output(m, n, table, fres);  
 *//check\_similar(fres, f);  
 //check\_similar2(fres, f);* system("pause");  
 return 0;  
}



#include <iostream>  
#include <math.h>  
#include <string>  
  
using namespace std;  
const int NotUsed = system("color F0");  
  
  
bool function(bool \*x, int xn, bool flag) *//моделирование работы схемы МДНФ*{  
 int tx;  
 if (xn != 0)  
 {  
 tx = x[xn - 1];  
 if (flag)  
 x[xn - 1] = 1;  
 else  
 x[xn - 1] = 0;  
 }  
 bool z1 = x[0] &&!x[1],  
 z2= !x[0] && x[1],  
 z3 = !x[2] && !x[4],  
 z4 = x[2] && x [4],  
 z5 = !x[3] && x[0],  
 z6 = !x[0] && x[3];  
  
 bool u1 = z1 && z5,  
 u2 = z2 && z6,  
 u3 = z1 && z3,  
 u4 = z3 && z5,  
 u5 = z6 && z4,  
 u6 = z2 && z4;  
  
 bool y1 = u1 || u2,  
 y2 = u3 || u4,  
 y3 = u5 || u6,  
 k1 = y1 || y2;  
  
  
  
 if (xn != 0)  
 x[xn - 1] = tx;  
 return k1|| y3;  
}  
  
  
  
  
  
void truth\_table(int m, int n, bool \*\*table, bool \*fres, bool \*\*f) *//функция для построения таблицы истинности*{  
 *//заполнение table двоичными векторами* for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)  
 {  
 table[i][j] = ((i >> (m - 1 - j)) & 1);  
 }  
 }  
 *//получение результата функции и запись его в f при значениях, записанных в i-той строке table* for (int i = 0; i < n; i++)  
 {  
  
  
 fres[i] = function(table[i], 0, false);  
  
 }  
}  
  
  
void output(int m, int n, bool \*\*table, bool \*fres)  
{  
 for (int i = 0; i < m; i++)  
 {  
 cout << "x" << i + 1 << " ";  
 }  
 cout << "f\t";  
 for (int i = 0; i < 10; i++)  
 {  
 cout << "R" << i + 1 << "\t";  
 }  
 cout << "\n";  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)  
 {  
 cout << table[i][j] << " ";  
 }  
 *// cout << "x"<<i<<"\t";* 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 << (function(table[i], 1, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 1, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 2, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 2, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 3, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 3, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 4, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 4, true) ^ fres[i]) << "\t";  
 cout << (function(table[i], 5, false) ^ fres[i]) << "\t";  
 cout << (function(table[i], 5, true) ^ fres[i]) << "\t";  
  
 cout << "\n";  
 }  
 cout << check\_test(3, true)<<"\n\n";  
  
}  
  
int main()  
{  
 int m = 5, n = pow(2, m);  
 bool \*\*table = new bool\*[n];  
 for (int i = 0; i < n; i++)  
 {  
 table[i] = new bool[m];  
 }  
 bool \*fres = new bool[n];  
 int mf = 10, nf = 32;  
 bool \*\*f = new bool\*[nf];  
 for (int i = 0; i < nf; i++)  
 {  
 f[i] = new bool[mf];  
 }  
 truth\_table(m, n, table, fres, f);  
 output(m, n, table, fres);  
 *//check\_similar(fres, f);  
 //check\_similar2(fres, f);* system("pause");  
 return 0;  
}

Проверяющий тест: { x7, x8, x11 , x27 }

Минимальный диагностический тест: {x7, x8, x11 , x27 , x14 , x3 , x23 , x4 }

