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 №2

по дисциплине: Теория цифровых автоматов

тема: «Синтез и анализ комбинационных схем с одним

выходом в базисе монофункциональных базисах»

Выполнил: ст. группы ВТ-32

Воскобойников И. С.

Проверил: Рязанов Ю. Д,

Белгород 2020 г.

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

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 ИНЕ. 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 ИЛИ-НЕ. Операцию НЕ реализовывать элементом ИЛИ-НЕ с запараллеленными входами.

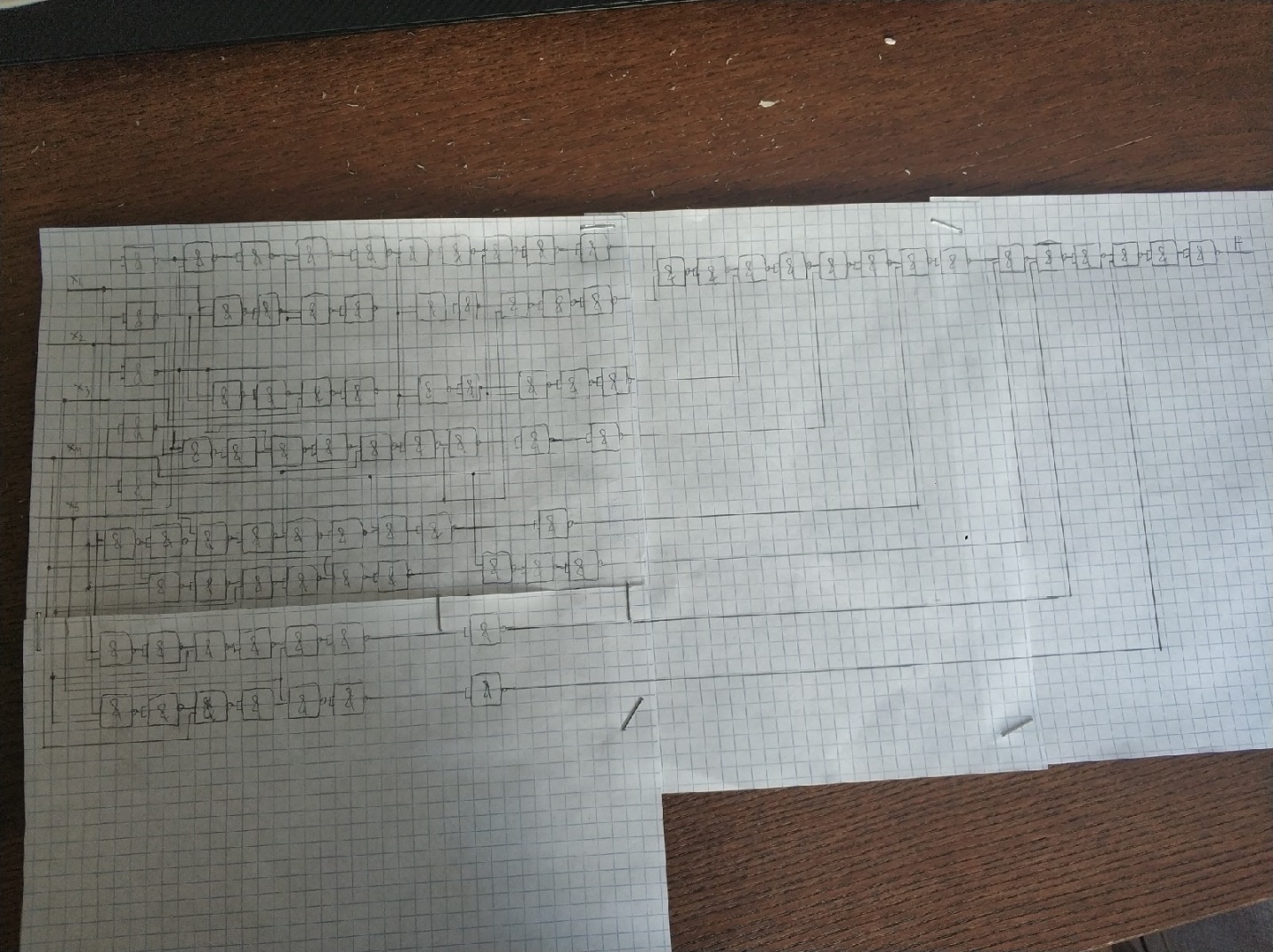
6.Комбинационную схему, полученную в лабораторной работе № 1, п. 5, преобразовать в комбинационную схему, состоящую только из элементов ИЛИ-НЕ, путем замены элементов И, ИЛИ, НЕ на их логические эквиваленты, состоящие только из элементов ИЛИ-НЕ. После формального построения комбинационной схемы исключить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

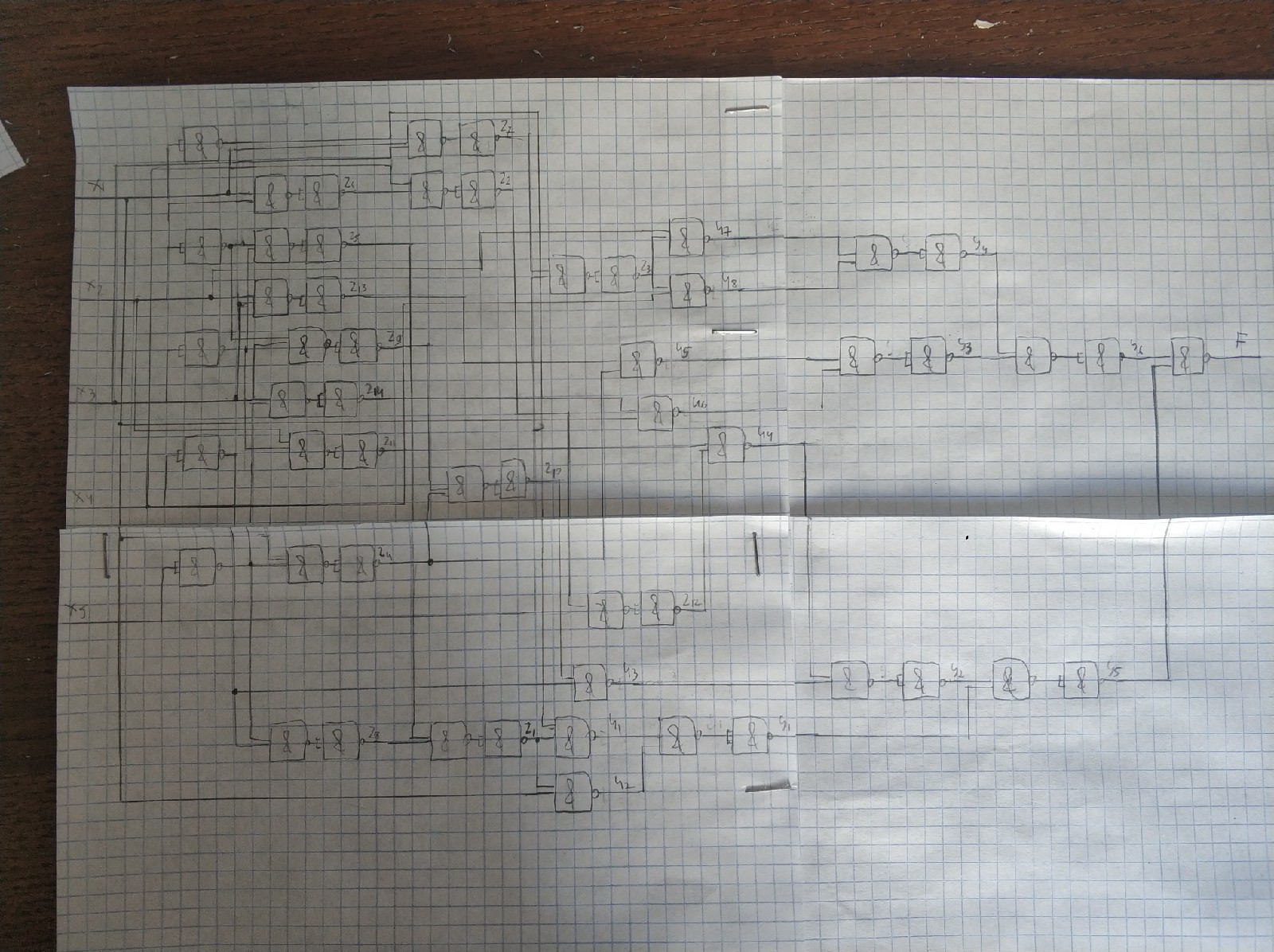
8.Сравнить схемы, построенные в лабораторных работах № 1 и № 2 по Квайну и по быстродействию.

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 ИНЕ. Операцию НЕ реализовывать элементом И-НЕ с запараллеленными входами.

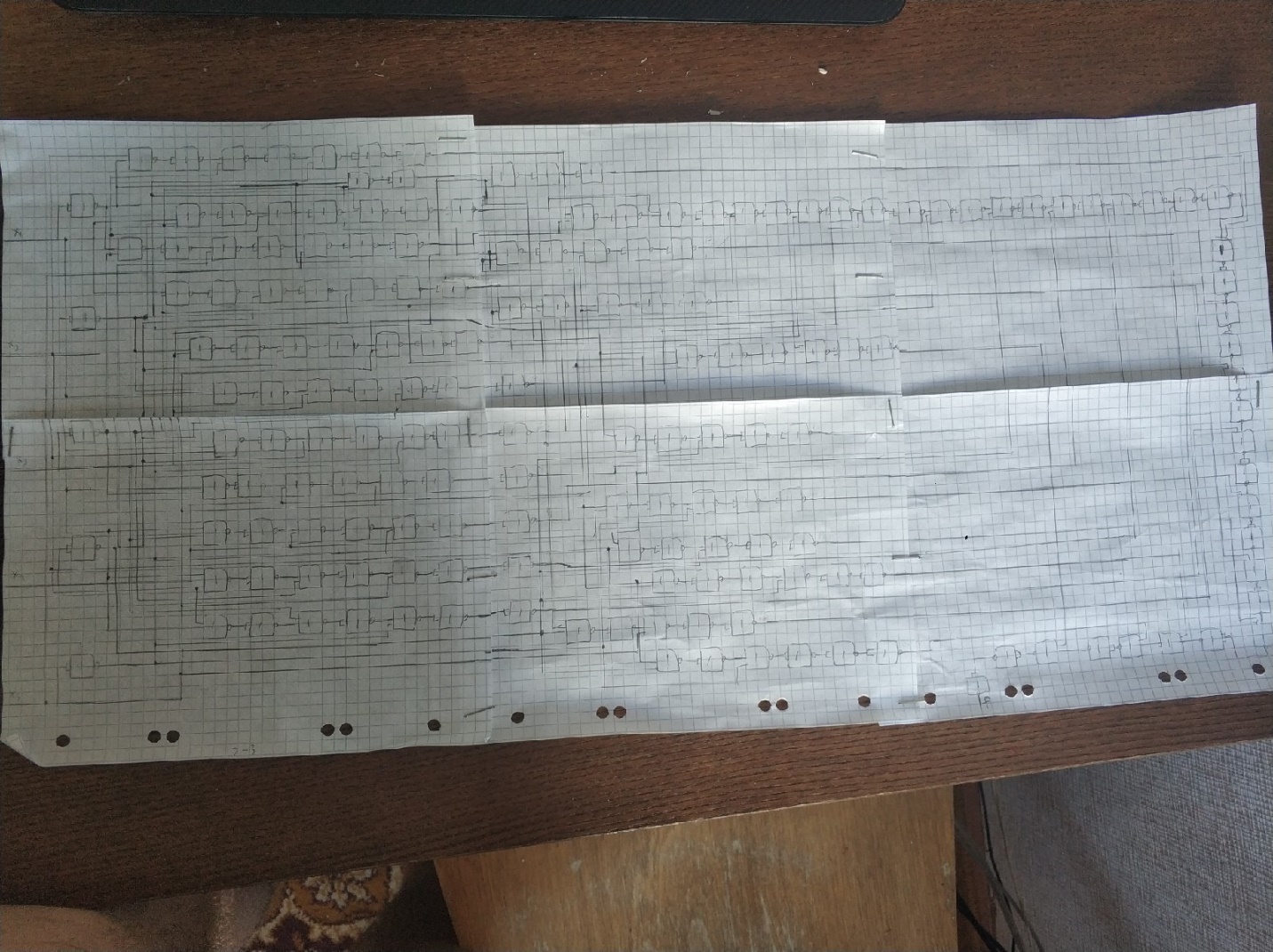


3. Комбинационную схему, полученную в лабораторной работе № 1, п. 3, преобразовать в комбинационную схему, состоящую только из элементов И-НЕ, путем замены элементов И, ИЛИ, НЕ на их логические эквиваленты, состоящие только из элементов И-НЕ. После формального построения комбинационной схемы исключить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

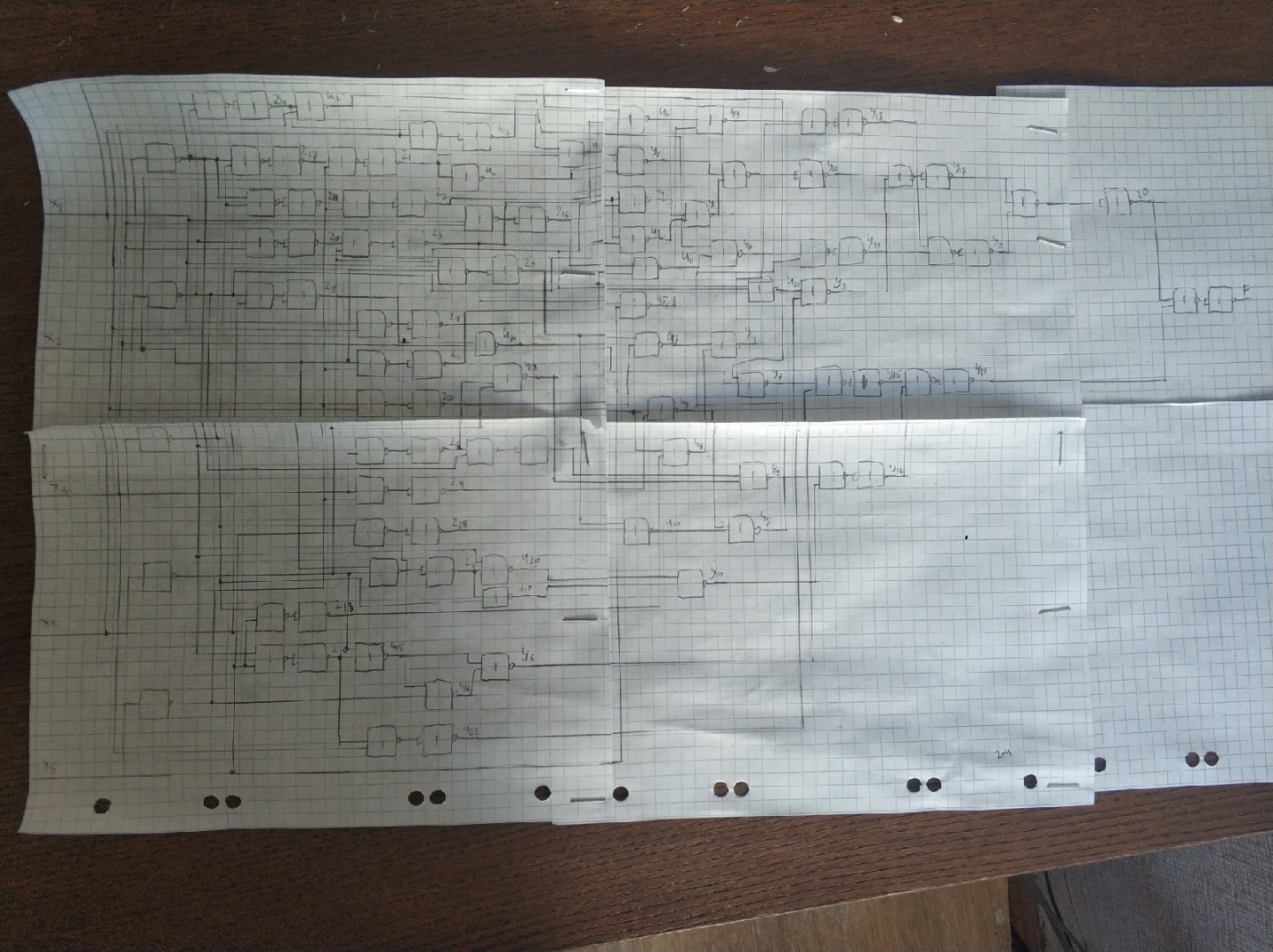


4. Преобразовать минимальную конъюнктивную нормальную форму булевой функции (см. лабораторную работу №1, п. 4) в аналитическое выражение, содержащее только операции ИЛИ-НЕ и НЕ.

5. По полученному выражению построить схему из элементов ИЛИ-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.

int TableValue(int \*X)

{

int Value = (X[1] \* pow(2, 1) + X[2] \* pow(2, 0)) + (X[3] \* pow(2, 2) + X[4] \* pow(2, 1) + X[5] \* pow(2, 0));

if ((5 <= Value) && (Value < 9))

return 1;

else

return 0;

}

int AndNot(int x, int y)

{

return !(x && y);

}

int ANAnd(int x, int y)

{

return AndNot(AndNot(x, y), AndNot(x, y));

}

int ANOr(int x, int y)

{

return AndNot(AndNot(x, x), AndNot(y, y));

}

int OrNot(int x, int y)

{

return !(x || y);

}

int ONAnd(int x, int y)

{

return OrNot(OrNot(x, x), OrNot(y, y));

}

int ONOr(int x, int y)

{

return OrNot(OrNot(x, y), OrNot(x, y));

}

bool FuncDNF(bool \*x)

{

bool z[14] = x[2] && !x[x4],

z[13] = x[1] && !x[2],

z[4] = x[0] && !x[4],

z[5] = !x[1] && !x[2],

z[6] = !x[0] && x[1],

z[7] = x[0] && x[2],

z[8] = !x[3] && !x[4],

z[9] = !x[1] && x[2],

z[10] = z[4] && z[9],

z[11] = x[1] && !x[2],

z[2] = x[3] && z[6],

z[12] = x[4] && z[2],

z[1] = z[5] && z[8],

z[3] = x[4] && z[7];

bool u[1] = !x[0] && z[1],

u2 = x[0] && z[1],

u3 = !x[3] && z10,

u4 = z[11] && z[12],

u5 = x[3] && z[4],

u6 = z[2] && z[14],

u7 = x[1] && z[3],

u8 = x[3] && z[3];

bool y[1] = u[1] || u[2],

y2 = u[3] || u[4],

y3 = u[5] || u[6],

y4 = u[7] || u[8],

y5 = y[1] || y[2],

y6 = y[3] || y[4];

return y[5] || y[6];

}

bool FuncKNF(bool \*x)

{

bool z[4] = !x[0] || !x[1],

z[5] = !x[1] || x[3],

z[6] = x[0] || x[2],

z[7] = x[1] || !x[3],

z[8] = x[0] || x[3],

z[9] = x[0] || x[1],

z[10] = !x[0] || x[4]

z[11] = x[1] || !x[4],

z[12] = !x[3] || !x[4],

z[13] = !x[2] || x[4],

z[14] = x[2] || !x[4],

z[15] = !x[2] || x[4],

z[16] = x[0] || x[2],

z[17] = !x[0] || !x[2],

z[18] = x[0] || !x[2],

z[19] = !x[1] || x[2],

z[1] = !x[4] || z[17],

z[2] = !x[4] || z[18],

z[3] = !x[3] || z[19];

bool u[1] = x[3] || z[1],

u[2] = x[1] || z[1],

u[3] = !x[1] || z[2],

u[4] = !x[3] || z[2],

u[5] = x[4] || z[3],

u[6] = x[0] || z[3],

u[7] = z[9] || z[12],

u[8] = z[8] || z[13],

u[9] = z[8] || z[14],

u[10] = z[9] || z[15],

u[11] = z[11] || z[16],

u[12] = x[2] || z[4],

u[13] = x[3] || z[4],

u[14] = !x[2] || z[5],

u[15] = !x[3] || z[10],

u[16] = !x[4] || z[5],

u[17] = !x[1] || z[6],

u[18] = !x[3] || z[6],

u[19] = !x[2] || z[7],

u[20] = !x[0] || z[7],

u[21] = !x[3] || z[11],

u[22] = x[2] || z[10];

bool y[1] = u[1] && u[2],

y[2] = u[3] && u[4],

y[3] = u[5] && u[6],

y[4] = u[7] && u[8],

y[5] = u[9] && u[10],

y[6] = u[11] && u[12],

y[7 = u[13] && u[14],

y[8] = u[15] && u[16],

y[9] = u[17] && u[18],

y[10] = u[19] && u[20],

y[11] = u[21] && u[22],

y[12] = y[1] && y[2],

y[13] = y[3] && y[4],

y[14] = y[5] && y[6],

y[15] = y[7] && y[8],

y[16] = y[9] && y[10],

y[17] = y[11] && y[12],

y[18] = y[13] && y[14],

y[19] = y[15] && y[16],

y[20] = y[17] && y[18],

return y[19] && y[20];

}

bool OrNotKNF(bool \*x) return OrNot(OrNot(x, y), OrNot(x, y));  
{  
 bool z[20], u[23];  
  
 z[4] = OrNot (OrNot(x[0],x[0]),OrNot(x[1],x[1])), OrNot(x[0],x[0]),OrNot(x[1],x[1])));  
 z[5] = OrNot(OrNot (OrNot(x[1],x[1]),x[3]), OrNot (OrNot(x[1],x[1]),x[3]));  
 z[6] = OrNot(OrNot (x[0],x[2]), OrNot (x[0],x[2]));  
 z[7] = OrNot(OrNot (x[1],OrNot(x[3],x[3])), OrNot (x[1],OrNot(x[3],x[3])));  
 z[8] = OrNot(OrNot (x[0],x[3]), OrNot (x[0],x[3]));  
 z[9] = OrNot(OrNot (x[0],x[1]), OrNot (x[0],x[1]));  
 z[10] = OrNot(OrNot (OrNot(x[0],x[0]),x[4]), OrNot (OrNot(x[0],x[0]),x[4]));  
 z[11] = OrNot(OrNot (x[1],OrNot(x[4],x[4])), OrNot (x[1],OrNot(x[4],x[4])));  
 z[12] = OrNot(OrNot (OrNot(x[3],x[3]),OrNot(x[4],x[4])), OrNot (OrNot(x[3],x[3]),OrNot(x[4],x[4])));  
 z[13] = OrNot(OrNot (OrNot(x[2],x[2]),x[4]), OrNot (OrNot(x[2],x[2]),x[4]));  
 z[14] = OrNot(OrNot (x[2],OrNot(x[4],x[4])), OrNot (x[2],OrNot(x[4],x[4])));  
 z[15] = OrNot(OrNot (OrNot(x[2],x[2]),x[4]), OrNot (OrNot(x[2],x[2]),x[4]));  
 z[16] = OrNot(OrNot (x[0],x[2]), OrNot (x[0],x[2]));  
 z[17] = OrNot(OrNot (OrNot(x[0],x[0]),OrNot(x[2],x[2])), OrNot (OrNot(x[0],x[0]),OrNot(x[2],x[2])));  
 z[18] = OrNot(OrNot (x[0],OrNot(x[2],x[2])), OrNot (x[0],OrNot(x[2],x[2])));  
 z[19] = OrNot(OrNot (OrNot(x[1],x[1]),x[2]), OrNot (OrNot(x[1],x[1]),x[2]));  
 z[1] = OrNot(OrNot (OrNot(x[4],x[4]),z[17]), OrNot (OrNot(x[4],x[4]),z[17]));  
 z[2] = OrNot(OrNot (OrNot(x[4],x[4]),z[18]), OrNot (OrNot(x[4],x[4]),z[18]));  
 z[3] = OrNot(OrNot (OrNot(x[3],x[3]),z[19]), OrNot (OrNot(x[3],x[3]),z[19]));  
  
 u[1] = OrNot(x[3],z[1]);  
 u[2] = OrNot(x[1],z[1]);  
 u[3] = OrNot(OrNot(x[1],x[1]),z[2]);  
 u[4] = OrNot(OrNot(x[3],x[3]),z[2]);  
 u[5] = OrNot(x[4],z[3]);  
 u[6] = OrNot(x[0],z[3]);  
 u[7] = OrNot(z[9],z[12]);  
 u[8] = OrNot(z[8],z[13]);  
 u[9] = OrNot(z[8],z[14]);  
 u[10] = OrNot(z[9],z[15]);  
 u[11] = OrNot(z[11],z[16]);  
 u[12] = OrNot(x[2],z[4]);  
 u[13] = OrNot(x[3],z[4]);  
 u[14] = OrNot(OrNot(x[2],x[2]),z[5]);  
 u[15] = OrNot(OrNot(x[3],x[3]),z[10]);  
 u[16] = OrNot(OrNot(x[4],x[4]),z[5]);  
 u[17] = OrNot(OrNot(x[1],x[1]),z[6]);  
 u[18] = OrNot(OrNot(x[3],x[3]),z[6]);  
 u[19] = OrNot(OrNot(x[2],x[2]),z[7]);  
 u[20] = OrNot(OrNot(x[0],x[0]),z[7]);  
 u[21] = OrNot(OrNot(x[3],x[3]),z[11]);  
 u[22] = OrNot(x[2],z[10]);  
  
  
 int f = 1;  
 for (int i = 1; i <= 20; i++)  
 f = AndNot(AndNot(f, f), AndNot(u[i], u[i]));  
 return f;  
}

bool AndNotDNF(bool \*x)  
{ bool z[15], u[9];  
 z[14] = AndNot(AndNot (AndNot(x[2],AndNot(x[4],x[4]))), AndNot (AndNot(x[2],AndNot(x[4],x[4])))));  
 z[13] = AndNot(AndNot (AndNot(x[1],AndNot(x[2],x[2]))), (AndNot(x[1],AndNot(x[2],x[2])))) ;  
 z[4] = AndNot(AndNot (AndNot(x[0],AndNot(x[4],x[4]))), (AndNot(x[0],AndNot(x[4],x[4]))));  
 z[5] = AndNot(AndNot ( AndNot(AndNot(x[1],x[1]),AndNot(x[2],x[2]))), ( AndNot(AndNot(x[1],x[1]),AndNot(x[2],x[2]))));  
 z[6] = AndNot(AndNot ( AndNot(AndNot(x[0],x[0]),x[1])), ( AndNot(AndNot(x[0],x[0]),x[1])));  
 z[7] = AndNot(AndNot ( AndNot(x[0],x[2])), ( AndNot(x[0],x[2])));  
 z[8] = AndNot(AndNot ( AndNot(AndNot(x[3],x[3]),AndNot(x[4],x[4]))), ( AndNot(AndNot(x[3],x[3]),AndNot(x[4],x[4]))));  
 z[9] = AndNot(AndNot (AndNot(AndNot(x[1],x[1]),x[2])), (AndNot(AndNot(x[1],x[1]),x[2])));  
 z[10] = AndNot(AndNot (AndNot(z[4],z[9])), (AndNot(z[4],z[9])));  
 z[11] = AndNot(AndNot (AndNot(x[1],AndNot(x[2],x[2]))), (AndNot(x[1],AndNot(x[2],x[2]))));  
 z[2] = AndNot(AndNot (AndNot(x[3],z[6])), (AndNot(x[3],z[6])));  
 z[12] = AndNot(AndNot ( AndNot(x[4],z[2])), ( AndNot(x[4],z[2])));  
 z[1] = AndNot (AndNot(z[5],z[8]), AndNot(z[5],z[8]));  
 z[3] = AndNot ( AndNot(x[4],z[7]), AndNot(x[4],z[7]));  
  
 u[1] = AndNot(AndNot(x[1],x[1]),z[1]);  
 u[2] = AndNot(x[0],z[1]);  
 u[3] = AndNot(AndNot(x[3],x[3]),z[10]);  
 u[4] = AndNot(z[11],z[12]);  
 u[5] = AndNot(x[3],z[4]);  
 u[6] = AndNot(z[2],z[14]);  
 u[7] = AndNot(x[1],z[3]);  
 u[8] = AndNot(x[3],z[3]);  
  
 int f = 0;  
 for (int i = 1; i <=6 ; i++)  
 f = ANOr(f, u[i]);  
 return f;  
}  
  
bool DNFImp4(bool x1, bool x2, bool x3, bool x4)  
{  
return AndNot(ANAnd(x1, x2), ANAnd(x3, x4));  
}  
bool DNFImp3(bool x1, bool x2, bool x3)  
{  
 return AndNot(ANAnd(x1, x2), x3);  
}  
  
bool SchemeAndNot(bool \*X)  
{  
 bool U[9];  
 U[1] = DNFImp3(AndNot(X[1], X[1]), X[3], X[5]);  
 U[2] = DNFImp4(AndNot(X[2], X[2]), X[3], X[4], AndNot(X[5], X[5]));  
 U[3] = DNFImp3(AndNot(X[1], X[1]), X[2], X[3]);  
 U[4] = DNFImp3(X[1], X[3], AndNot(X[4], X[4]));  
 U[5] = DNFImp4(X[1], AndNot(X[3], X[3]), X[4], X[5]);  
 U[6] = DNFImp4(X[1], X[2], AndNot(X[3], X[3]), X[4]);  
 U[7] = DNFImp3(X[1],AndNot(X[5],X[5]),X[6]);  
 U[8] = DNFImp4(X[1], AndNot(X[6],X[6]),X[7]))  
 int f = 1;  
 for (int i = 1; i <= 6; i++)  
 f = ANAnd(f, U[i]);  
 return AndNot(f, U[6]);  
}  
  
bool KNFImp4(bool x1, bool x2, bool x3, bool x4)  
{  
 return OrNot(ONOr(x1, x2), ONOr(x3, x4));  
}  
  
bool KNFImp3(bool x1, bool x2, bool x3)  
{  
return OrNot(ONOr(x1, x2), x3);  
}  
  
bool SchemeOrNot(bool \*X)  
{  
 bool U[7];  
 U[1] = KNFImp4 (X[3], OrNot(X[4],X[4]), OrNot(OrNot(x[0],x[0]),OrNot(x[2],x[2]));  
 U[2] = KNFImp4 (X[1], OrNot(X[4],X[4]), OrNot(OrNot(x[0],x[0]),OrNot(x[2],x[2]));  
 U[3] = KNFImp3(OrNot(X[1],X[1]), OrNot(X[4],X[4]), OrNot(x[0],OrNot(x[2],x[2]));  
 U[4] = KNFImp4(OrNot(X[1], X[1]), OrNot(X[3], X[3]), OrNot(X[4], X[4]), OrNot(X[5], X[5]));  
 U[5] = KNFImp4(OrNot(X[1], X[1]), OrNot(X[2], X[2]), OrNot(X[3], X[3]), OrNot(X[4], X[4]));  
 U[6] = KNFImp4(X[1], OrNot(X[3], X[3]), OrNot(x[1],x[1]),x[2]);

U[7] = KNFImp3(OrNot(X[0], X[1]), OrNot(X[3], X[3]), OrNot(x[4],x[4]));  
 U[8] = KNFImp3(OrNot(x[0],x[3]), OrNot(x[2],x[2]),x[4]);

U[9] = KNFImp3(OrNot(x[0],x[3]), x[2],OrNot(x[4],x[4]));

U[10] = KNFImp3(OrNot(x[0],x[1]), OrNot(x[2],x[2]),x[4]);

U[11] = KNFImp3 (x[1],OrNot(x[4],x[4]), OrNot(x[0],x[2]);  
 U[12] = KNFImp3 (X[2], OrNot(x[0],x[0]),OrNot(x[1],x[1]));

U[13] = KNFImp3(X[3], OrNot(x[0],x[0]),OrNot(x[1],x[1]));

U[14] = KNFImp3(OrNot(x[2],x[2]), OrNot(x[1],x[1]),x[3]);

U[15] = KNFImp3(OrNot(x[3],x[3]), OrNot(x[0],x[0]),x[4]);

U[16] = KNFImp3(OrNot(x[4],x[4]), OrNot(x[1],x[1]),x[3]);

U[17] = KNFImp3(OrNot(x[1],x[1]), X[0],X[2]);

U[18] = KNFImp3(OrNot(x[3],x[3]), X[0],X[2]);

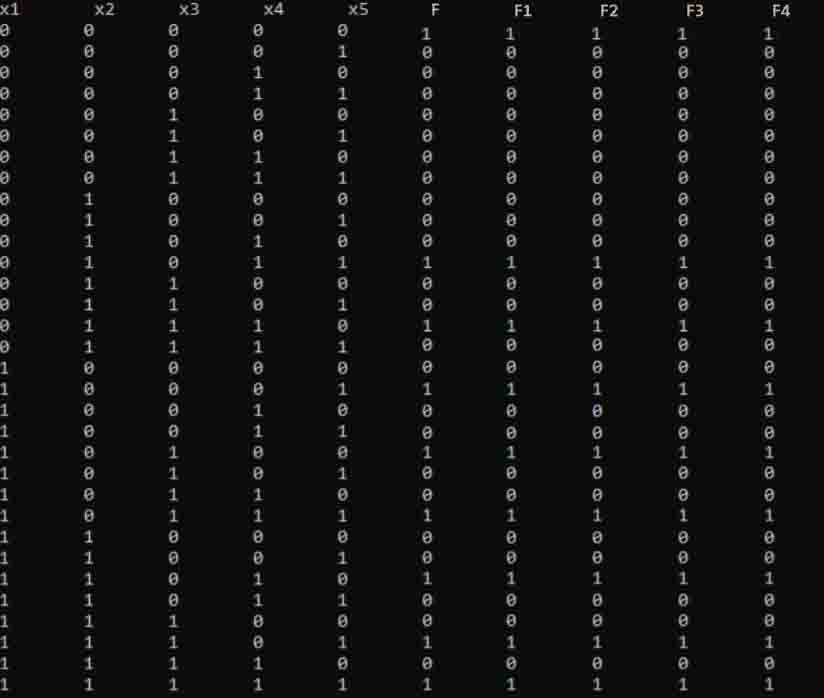
U[19] = KNFImp3(OrNot(x[2],x[2]), X[1],OrNot(X[3],X[3]));

U[20] = KNFImp3(OrNot(x[0],x[0]), X[1],OrNot(X[3],X[3]));

U[21] = KNFImp3(OrNot(x[3],x[3]), X[1],OrNot(X[4],X[4]));

U[22]= KNFImp3(X[2]), OrNot(X[0],X[0]),X[4]);

int f = 0;  
 for (int i = 1; i <= 6; i++)  
 f = ONOr(f, U[i]);  
 return OrNot(f, U[6]);  
}  
  
int main(int argc, const char \* argv[]) {  
 int X[6];  
 for (int i = 1; i < 6; i++)  
 printf(" x%d|", i);  
 printf(" F F1 F2 F3 F4 \n");  
  
 for (X[1] = 0; X[1] < 2; X[1]++)  
 for (X[2] = 0; X[2] < 2; X[2]++)  
 for (X[3] = 0; X[3] < 2; X[3]++)  
 for (X[4] = 0; X[4] < 2; X[4]++)  
 for (X[5] = 0; X[5] < 2; X[5]++)  
 {  
 for (int i = 1; i < 6; i++)  
 printf(" %2d|", X[i]);  
 printf(" %2d ", TableValue(X));  
 printf(" %2d ", AndNotDNF(X));  
 printf(" %2d ", OrNotKNF(X));  
 printf(" %2d ", SchemeAndNot(X));  
 printf(" %2d ", SchemeOrNot(X));  
  
 printf("\n");  
 }  
 return 0;  
}



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

ДНФ:  
Сложность по Квайну = 63  
Сложность по быстродействию =7

КНФ:  
Сложность по Квайну = 129  
Сложность по быстродействию =8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Лаб 1 | | Лаб 2 | |
|  | Сложность по Квайну | Сложность по быстродействию | Сложность по Квайну | Сложность по быстродействию |
| ДНФ: | 63 | 7 | 180 | 25 |
| КНФ: | 129 | 8 | 380 | 49 |
| 3 задние лаб 2 |  |  | 104 | 11 |
| 6 задание лаб 2 |  |  | 184 | 14 |

Ответы на вопросы

1. Функция ANAnd предназначена для ДНФ, входа в z
2. ANOr предназначен для окончательного ввода функции f для днф
3. Функция ONOr предназначена для КНФ, входа в z
4. ONAnd предназначен для окончательного ввода функции f для днф
5. Переменных использую меньше чем связей потому что использую циклы где можно