Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Арифметические и логические основы цифровых устройств

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_\_И. В. Лукьянова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе  
на тему

ПРОЕКТИРОВАНИЕ И ЛОГИЧЕСКИЙ СИНТЕЗ СУММАТОРА-УМНОЖИТЕЛЯ ДВОИЧНО-ЧЕТВЕРИЧНЫХ ЧИСЕЛ

БГУИР КР 1-40 02 01 530 ПЗ

Студент

Руководитель

М. А. Ходосевич

И. В. Лукьянова

МИНСК 2022

Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Арифметические и логические основы цифровых устройств

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_ Б. В. Никульшин

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2022 г.

ЗАДАНИЕ

по курсовой работе студента  
Матвей Александрович Ходосевич

1. Тема работы: “Проектирование и логический синтез сумматора-умножителя двоично-четверичных чисел”
2. Срок сдачи студентом законченной работы: 20 мая 2022 г.
3. Исходные данные к работе:
   1. исходные сомножители: Мн = 94,35; Мт = 02,68.
   2. алгоритм умножения: В.
   3. метод умножения: умножение закодированного двоично-четверичного множимого на два разряда двоичного множителя одновременно в дополнительном коде.
   4. коды четверичных цифр множимого для перехода к двоично-четверичной системе кодирования; 04 – 01, 14 – 00, 24 – 10, 34 – 11.
   5. тип синтезируемого умножителя: 1.
   6. тип синтезируемого умножителя: 1.
   7. логический базис для реализации ОЧС: И, НЕ; метод минимизации – карты Карно-Вейча.
   8. логический базис для реализации ОЧУ: ИЛИ-НЕ; метод минимизации – алгоритм Рота.
4. Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Введение. 1. Разработка алгоритма умножения. 2. Разработка структурной схемы сумматора-умножителя. 3. Разработка функциональных схем основных узлов сумматора-умножителя. 4. Синтез комбинационных схем устройств на основе мультиплексоров. 5. Оценка результатов разработки. Заключение. Список литературы.

1. Перечень графического материала:
   1. Сумматор-умножитель первого типа. Схема электрическая структурная.
   2. Одноразрядный четвертичный сумматор. Схема электрическая функциональная.
   3. Одноразрядный четверичный умножитель. Схема электрическая функциональная.
   4. Одноразрядный четвертичных сумматор. Реализация на мультиплексорах. Схема электрическая функциональная.
   5. Преобразователь множителя. Схема электрическая функциональная.

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов курсовой работы | Объём этапа, % | Срок выполнения этапа | Примечания |
| Разработка алгоритма умножения | 10 | 10.02-20.02 |  |
| Разработка структурной схемы сумматора-умножителя | 10 | 21.02-09.03 | С выполнением чертежа |
| Разработка функциональных схем основных узлов сумматора-умножителя | 50 | 10.03-30.04 | С выполнением чертежей |
| Синтез комбинационных схем устройств на основе мультиплексоров | 10 | 01.05-15.05 | С выполнением чертежа |
| Завершение оформления пояснительной записки | 20 | 15.05-20.05 |  |

Дата выдачи задания: 10 февраля 2022 г.

Руководитель И. В. Лукьянова

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_\_\_ /Ходосевич М.A. /

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc71831794)

[1 РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ 6](#_Toc71831795)

[2 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА-УМНОЖИТЕЛЯ 8](#_Toc71831796)

[3 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА-УМНОЖИТЕЛЯ 11](#_Toc71831797)

[3.1 Логический синтез одноразрядного четверичного умножителя 11](#_Toc71831798)

[3.2 Логический синтез одноразрядного четверичного сумматора 22](#_Toc71831799)

[4 СИНТЕЗ СХЕМЫ ОЧС НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ 26](#_Toc71831800)

[5 ЛОГИЧЕСКИЙ СИНТЕЗ ПРЕОБРАЗОВАТЕЛЯ МНОЖИТЕЛЯ (ПМ) 28](#_Toc71831801)

[6 ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ ВРЕМЕННЫЕ ЗАТРАТЫ НА УМНОЖЕНИЕ 29](#_Toc71831802)

[ЗАКЛЮЧЕНИЕ 30](#_Toc71831803)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc71831804)

[ПРИЛОЖЕНИЕ А 32](#_Toc71831805)

[ПРИЛОЖЕНИЕ Б 33](#_Toc71831806)

[ПРИЛОЖЕНИЕ В 34](#_Toc71831807)

[ПРИЛОЖЕНИЕ Г 35](#_Toc71831808)

[ПРИЛОЖЕНИЕ Д 36](#_Toc71831809)

[ПРИЛОЖЕНИЕ Е 37](#_Toc71831810)

# **ВВЕДЕНИЕ**

Курсовое проектирование является обязательным элементом подготовки специалиста с высшим образованием и является одной из форм текущей аттестации студента по учебной дисциплине.

Целью данной курсовой работы является проектирование двоично-четверичного сумматора-умножителя (СУ). Сумматор является одним из центральных узлов арифметико-логического устройства (АЛУ) вычислительной машины, поэтому глубокое понимание принципов его работы критически важно для современного инженера. Для того чтобы спроектировать данное устройство, необходимо пройти несколько последовательных этапов разработки:

* разработать алгоритм умножения и оценить погрешности вычислений;
* разработать структурную схему сумматора-умножителя первого типа;
* разработать функциональные схемы основных узлов сумматора-умножителя в заданных логических базисах;
* разработать схему одноразрядного четвертичного сумматора на основе мультиплексора;
* разработать преобразователь множителя;
* рассчитать время умножения;
* оформить документацию о проделанной работе.

В ходе выполнения курсовой работы автором были пройдены все эти этапы. В настоящей пояснительной записке изложено краткое описание процесса проектирования и приведена разработанная автором графическая документация по структурной схеме и функциональным схемам основных её узлов.

# 

# **1 РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ**

Множимое

94 | 4 0.35 Мн4 = 1132,11

8 23| 4 4 в соответствии с заданной

14 20 5| 4 1,04 кодировкой множимого

12 3 4 | 1 4 Мн2/4 = 00001110,0000

2 1 1,06

Множитель

2 | 4 0,13 Мт4 = 2,22320

0 0 4 В соответствии с обычной

2 0,52 весомозначной кодировкой

4 множителя:

2,08 Мт2/4 = 10,1010111000

4

0,32

Запишем сомножители в форме с плавающей запятой в прямом коде:

Мн = 0,000011100000 Рмн = 0,0100 + 1010 закодировано по заданию Мт = 0,101010111000 Рмт = 0,0001 + 0110 закодировано традиционно

Умножение двух чисел с плавающей запятой на два разряда множителя одновременно в прямых кодах. Это сводится к сложению порядков, формированию знака произведения, преобразованию разрядов множителя согласно алгоритму, и перемножению мантисс сомножителей.

Порядок произведения будет равен:

РМн = 0.0100 +104

РМт = 0.0001 +014

РМн∙Мт = 0.0101 +114

Результат закодирован в соответствии с заданием на кодировку множимого.

Знак произведения определяется суммой по модулю “два” знаков сомножителей:

зн Мн ⊕ зн Мт = 0 ⊕ 0 = 0

Для умножения мантисс необходимо предварительно преобразовать множитель. При умножении чисел в дополнительных кодах диада 11(34) заменяется на триаду , а диада 10(24) заменяется на триаду

Преобразованный множитель имеет вид: Мтп4 = . Перемножение мантисс по алгоритму “В” приведено в таблице 1.1

Таблица 1.1 - Перемножение мантисс

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Четверичная с/c | | | Двоично-четверичная с/с | | | Комментарии |
| 0. | 000000 | 0000000 | 00. | 010101010101 | 01010101010101 | ∑0 ч∙ 22 |
| 0. | 000000 | 0113211 | 00. | 010101010101 | 01000011100000 | П1Ч=[Мн] |
| 0. | 000000 | 0113211 | 00. | 010101010101 | 01000011100000 | ∑1 ч |
| 0. | 000000 | 1132110 | 00. | 010101010101 | 00001110000001 | ∑1 ч∙ 22 |
| 3. | 333333 | 0220123 | 11. | 111111111111 | 01101000101011 | П2Ч =[-Мн] |
| 0. | 000000 | 1012233 | 00. | 010101010101 | 00010010101111 | ∑2 ч |
| 0. | 000001 | 0122330 | 00. | 010101010100 | 01001010111101 | ∑2 ч∙ 22 |
| 3. | 333333 | 3220123 | 11. | 111111111111 | 11101001001011 | П3Ч= [-Мн] |
| 0. | 000001 | 0003113 | 00. | 010101010100 | 01010111000011 | ∑3 ч |
| 0. | 000010 | 0031130 | 00. | 010101010001 | 01011100001101 | ∑3 ч∙ 22 |
| 3. | 333333 | 3220123 | 11. | 111111111111 | 11101001001011 | П4Ч= [-Мн] |
| 0. | 000003 | 3311313 | 00. | 010101010111 | 11110000110111 | ∑4 ч |
| 0. | 000033 | 3113130 | 00. | 010101011111 | 11000011001101 | ∑4 ч∙ 22 |
| 0. | 000333 | 1131300 | 00. | 010101111111 | 00001100110101 | П5Ч= 0; ∑5 ч∙ 22; |
| 3. | 333333 | 3100312 | 11. | 111111111111 | 11000101110010 | П6Ч =[-2Мн] |
| 0. | 000333 | 0232212 | 00. | 010101111111 | 01101110100010 | ∑6 ч |
| 0. | 003330 | 2322120 | 00. | 010111111101 | 10111010001001 | ∑6ч∙22 |
| 0. | 033302 | 3221200 | 00. | 011111110110 | 11101000100101 | П7Ч =0; ∑7ч∙22 |
| 0. | 033302 | 3221200 | 00. | 011111110110 | 11101000100101 | ∑7ч =Мн∙Mт |

После окончания умножения необходимо оценить погрешность вычислений. Для этого полученное произведение (Мн∙Мт4 = 0,03330232212, РМн∙Мт = 5) приводится к нулевому порядку, а затем переводится в десятичную систему счисления:

Мн∙Мт4 = 3330,232212 РМн∙Мт = 0;

Мн∙Мт10 = 252,72802734375

Результат прямого перемножения операндов дает следующее значение:

Мн10 · Мт10 = 94,35 · 2.68 = 252.858.

Абсолютная погрешность:

Δ = 252.858 – 252,72802734375 = 0,12997266.

Относительная погрешность:

Эта погрешность получена за счет приближенного перевода из десятичной системы счисления в четверичную обоих сомножителей, а также за счет округления полученного результата произведения.

# **2 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА-УМНОЖИТЕЛЯ**

Структурная схема сумматора-умножителя первого типа для алгоритма умножения «B» представлена на схеме *ГУИР 400201.409 Э1*.

Структурная схема сумматора-умножителя первого типа строится на базе заданных узлов ОЧУ, ОЧС, формирователя дополнительного кода, преобразователя множителя и аккумулятора. Управление режимами работы схемы осуществляется внешним сигналом *mul/sum,* который определяет вид текущей арифметической операции (умножение или суммирование).

*Когда устройство работает как сумматор* (на входе *mul/sum* –  «1»), оба слагаемых последовательно (за два такта) заносятся в регистр множимого, а на управляющий вход формирователя дополнительного кода (ФДК) *F2* поступает «1».

На выходах ФДК формируется дополнительный код одного из слагаемых с учётом знака. Это слагаемое может быть записано в регистр результата, при этом управляющие сигналы, поступающие на входы *h* всех ОЧУ, дают возможность переписать на выходы ОЧУ разряды слагаемого без изменений (рисунок 2.1).

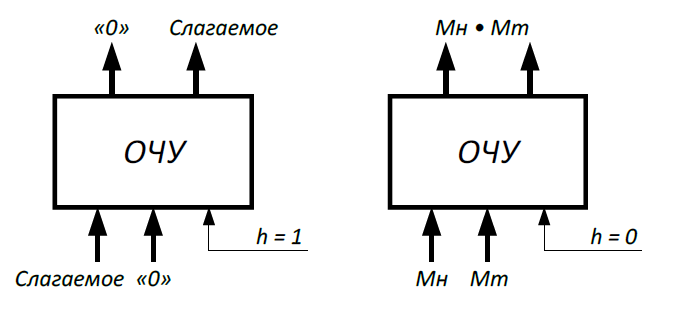
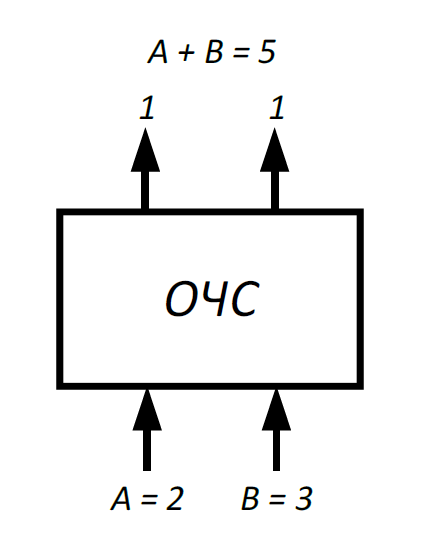


Рисунок 2.1 – Режимы работы ОЧУ

Если на вход *h* поступает «0», то ОЧУ перемножает разряды Мн и Мт.

Одноразрядный четверичный сумматор предназначен для сложения двух двоично-четверичных цифр, подаваемых на его входы (рисунок 2.2).

Рисунок 2.2 – Одноразрядный четверичный сумматор

В ОЧС первое слагаемое складывается с нулём, т.к. на старших выходах ОЧУ будут формироваться только коды нуля. Затем первое слагаемое попадает в регистр-аккумулятор, который изначально обнулён.

На втором такте второе слагаемое из регистра множимого через цепочку ОЧУ и ОЧС попадает в аккумулятор, где складывает с первым слагаемым. Таким образом, аккумулятор (накапливающий сумматор) складывает операнды и хранит результат.

*Когда устройство работает как умножитель* (на входе *Mul/sum* - «0»), множимое и множитель помещаются в соответствующие регистры, а на управляющий вход ФДК *F2* поступает «0».

В регистре множителя в конце каждого такта умножения содержимое сдвигается на два двоичных разряда и в последнем такте умножения регистр обнуляется.

Триада множителя поступает на входы преобразователя множителя.

В регистре множителя после каждого такта умножения содержимое сдвигается на два двоичных разряда и в конце умножения регистр обнуляется.

Выход 1 ПМ переходит в единичное состояние, если текущая диада содержит отрицание ( или ). В этом случае инициализируется управляющий вход F1 формирователя дополнительного кода и на выходах ФДК формируется дополнительный код множимого с обратным знаком (умножение на «– 1»).

Принцип работы ФДК, в зависимости от управляющих сигналов, приведён в таблице 2.1.

Таблица 2.1 – Режимы работы формирователя дополнительного кода

|  |  |  |
| --- | --- | --- |
| **Сигналы на входах ФДК** | | **Результат на выходах ФДК** |
| *F1* | *F2* |
| 0 | 0 | Дополнительный код множимого |

|  |  |  |
| --- | --- | --- |
| 0 | 1 | Дополнительный код слагаемого |
| 1 | 0 | Меняется знак Мн |
| 1 | 1 | Меняется знак слагаемого |

*Продолжение таблицы 2.1*

На выходах 2 и 3 ПМ формируются диады преобразованного множителя, которые поступают на входы ОЧУ вместе с диадами множимого.

ОЧУ предназначен лишь для умножения двух четверичных цифр. Если в процессе умножения возникает перенос в следующий разряд, необходимо предусмотреть возможность его прибавления.

ОЧС предназначен для суммирования результата умножения текущей диады Мн·Мт с переносом из предыдущей диады. Следовательно, чтобы полностью сформировать частичное произведение четверичных сомножителей, необходима комбинация цепочек ОЧУ и ОЧС.

Частичные суммы формируются в аккумуляторе. На первом этапе он обнулён и первая частичная сумма получается за счёт сложения первого частичного произведения (сформированного на выходах ОЧС) и нулевой частичной суммы (хранящейся в аккумуляторе).

В аккумуляторе происходит сложение *i*-й частичной суммы с (*i*+1)-м частичным произведением, результат сложения сохраняется. Содержимое аккумулятора сдвигается на один четверичный разряд влево в конце каждого такта умножения по алгоритму «В».

На четырёх выходах ОЧУ формируется результат умножения диад Мн·Мт. Максимальной цифрой в диаде преобразованного множителя является двойка, поэтому в старшем разряде произведения максимальной цифрой может оказаться только «1»:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 34 | · | 24 | = | 124 |
| max |  | max |  |  |
| Мн |  | Мт |  |  |

Это означает, что на младшие входы ОЧС никогда не поступят диады цифр, соответствующие кодам «2» и «3», следовательно, в таблице истинности работы ОЧС будут содержаться 16 безразличных входных наборов.

Количество тактов умножения определяется разрядностью Мт.

# **3 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА-УМНОЖИТЕЛЯ**

# **3.1 Логический синтез одноразрядного четверичного умножителя**

Одноразрядный четверичный умножитель – это комбинационное устройство, имеющее 5 двоичных входов (2 разряда из регистра Мн, 2 разряда из регистра Мт и управляющий вход *h*) и 4 двоичных выхода. Принцип работы ОЧУ представлен с помощью таблицы истинности (таблица 3.1).

Разряды множителя закодированы: 0 – 00; 1 – 01; 2- 10; 3 – 11.

Разряды множимого закодированы: 0 – 01; 1 – 00; 2 – 10; 3 – 11.

Управляющий вход *h* определяет тип операции:

- «0» - умножение закодированных цифр, поступивших на информационные входы;

- «1» - вывод на выходы без изменения значения разрядов, поступивших из регистра множимого.

В таблице 3.1 выделено восемь безразличных наборов, т.к. на входы ОЧУ из разрядов множителя не может поступить код «11».

Таблица 3.1 – Таблица истинности ОЧУ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Мн** | | **Мт** | | **Упр.** | **Старшие разряды** | | **Младшие разряды** | | **Пример операции в четверичной с/c** |
| ***x₁*** | ***x₂*** | ***y₁*** | ***y₂*** | ***h*** | ***P₁*** | ***P₂*** | ***P₃*** | ***P₄*** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1\*0=00 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | Выход – код «01» |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1\*1=01 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | Выход – код «01» |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1\*2=02 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | Выход – код «01» |
| 0 | 0 | 1 | 1 | 0 | x | x | x | x | 1\*3=03 |
| 0 | 0 | 1 | 1 | 0 | x | x | x | x | Выход – код «01» |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0\*0=00 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | Выход – код «00» |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0\*1=00 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | Выход – код «00» |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0\*2=00 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | Выход – код «00» |
| 0 | 1 | 1 | 1 | 0 | x | x | x | x | 0\*3=00 |
| 0 | 1 | 1 | 1 | 1 | x | x | x | x | Выход – код «00» |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 2\*0=00 |

*Продолжение таблицы 3.1*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | Выход – код «02» |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 2\*1=02 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | Выход – код «02» |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2\*2=10 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | Выход – код «02» |
| 1 | 0 | 1 | 1 | 0 | x | x | x | x | 2\*3=12 |
| 1 | 0 | 1 | 1 | 1 | x | x | x | x | Выход – код «02» |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 3\*0=00 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | Выход – код «03» |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 3\*1=03 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | Выход – код «03» |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 3\*2=12 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | Выход – код «03» |
| 1 | 1 | 1 | 1 | 0 | x | x | x | x | 3\*3=21 |
| 1 | 1 | 1 | 1 | 1 | x | x | x | x | Выход – код «03» |

Минимизацию функций P1, P2, P*3*  проведем с помощью карт Вейча.

Минимизацию функции P4 проведем с помощью алгоритма Рота. Для функции *P1* заполненная карта приведена на рисунке 3.1.1. В рисунках 3.1.1 – 3.1.3 символом «х» отмечены наборы, на которых функция может принимать произвольное значение (безразличные наборы).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| x | x | x | x | x | x | x | x | y2 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |

Рисунок 3.1.1 – Минимизация функции *P1* при помощи карты Вейча

Следовательно:

y1y2

Запишем результат минимизации в логическом базисе ИЛИ-НЕ:

Для функции *P2* заполненная карта приведена на рисунке 3.1.2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |  |
| x | x | x | x | x | x | x | x | y2 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |

Рисунок 3.1.2 – Минимизация функции *P2* при помощи карт Вейча

Следовательно:

Запишем результат минимизации в логическом базисе ИЛИ-НЕ:

Для функции *P3* заполненная карта приведена на рисунке 3.1.3.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |  |
| x | x | x | x | x | x | x | x | y2 |
|  | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |  |

Рисунок 3.1.3 - Минимизация функции *P3* при помощи карт Вейча

Следовательно:

Запишем результат минимизации в логическом базисе ИЛИ-НЕ:

Для минимизации функции *P4* воспользуемся алгоритмом Рота:

Определим множество единичных кубов:

*L* =

и множество безразличных кубов:

*N* =

Минимизацию безразличных кубов проведём с помощью карты Карно. Для безразличных кубов заполненная карта приведена на рисунке 3.1.4, где символом «x» отмечены наборы, на которых функция не определена.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 |  |  |  |  | x | x |  |  |
| 01 |  |  |  |  | x | x |  |  |
| 11 |  |  |  |  | x | x |  |  |
| 10 |  |  |  |  | x | x |  |  |

Рисунок 3.1.4 – Минимизация безразличных кубов с помощью карты Карно

Множество безразличных наборов после минимизации:

.

Сформируем множество *С0* = *L* ⋃ *N*:

.

Первым этапом алгоритма Рота является нахождение множества простых импликант.

Для реализации этого этапа воспользуемся операцией умножения (\*) над множествами *С0, С1* и т. д., пока в результате операции будут образовываться новые кубы большей размерности.

Первый шаг умножения (*С0\*С0*)приведён в таблице 3.1.5

Таблица 3.1.5 – Поиск простых импликант (*С0\*С0*)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C0\*C0 | 11011 | 11101 | 0x000 | 010x1 | x1010 | 0110x | 10x00 | 1100x |
| 11011 | - |  |  |  |  |  |  |  |
| 11101 | 11yy1 | - |  |  |  |  |  |  |
| 0x000 | y10yy | y1y0y | - |  |  |  |  |  |
| 010x1 | y1011 | y1y01 | 0100y | - |  |  |  |  |
| x1010 | 1101y | 11yyy | 010y0 | 0101y | - |  |  |  |
| 0110x | y1yy1 | y1101 | 01y00 | 01y01 | 01yy0 | - |  |  |
| 10x00 | 1y0yy | 1y10y | y0000 | yy00y | 1y0y0 | yy100 | - |  |
| 1100x | 110y1 | 11y01 | y1000 | y1001 | 110y0 | y1y0x | 1y000 | - |
| xx11x | 11y11 | 111y1 | 0xyy0 | 01y11 | x1y10 | 011yx | 101y0 | 11yyx |
| A1 | x1011 1101x 110x1 11x11 | x1101 11x01 111x1 | 0100x 010x0 01x00 x0000 x1000 | 0101x 01x01 x1001 01x11 | 110x0 x1x10 | 011xx | 1x000 101x0 | Ø |

В результате этой операции образуется множество кубов:

##### ={x1011; 1101x; 110x1; 11x11; x1101; 11x01; 111x1; 0100x; 010x0; 01x00; x0000; x1000; 0101x; 01x01; x1001; 01x11; 110x0; x1x10; 011xx; 1x000; 101x0}

Множество *Z0* кубов, не участвовавших в образовании новых кубов, пустое.

Также формируется множество B1 = C0 — Z0.

##### { 11011; 11101; 0x000; 010x1; x1010; 0110x; 10x00; 1100x; xx11x }

Для следующего шага получения множества Z формируется множество

С1 =.

##### { x1011; 1101x; 110x1; 11x11; x1101; 11x01; 111x1; 0100x; 010x0; 01x00; x0000; x1000; 0101x; 01x01; x1001; 01x11; 110x0; x1x10; 011xx; 1x000; 101x0; 0x000; 010x1; 10x00; 1100x; xx11x }

В таблице 3.1.6 приведён следующий шаг поиска простых импликант с помощью операции *С1\*С1*.

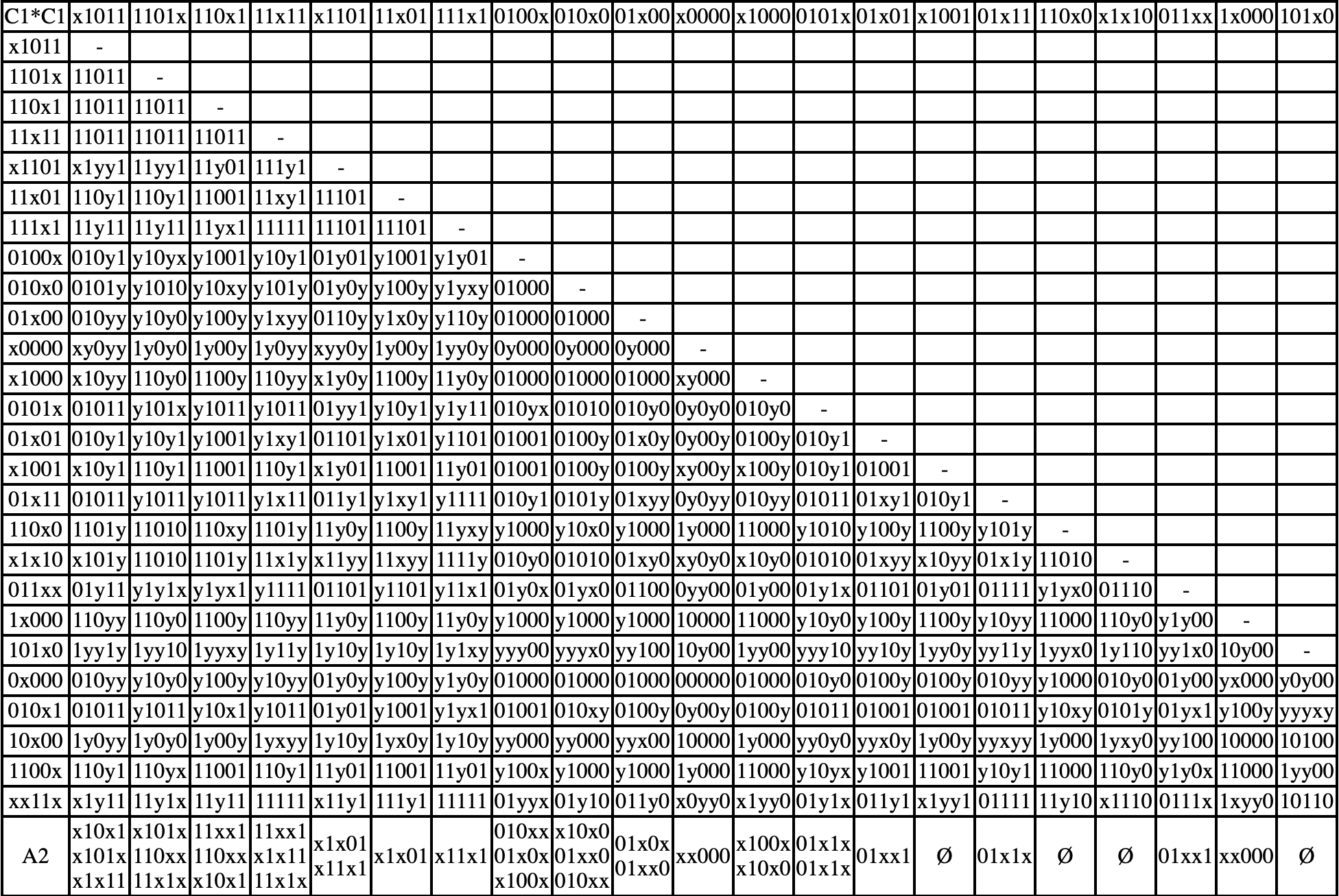


Таблица 3.1.6 – Поиск простых имплкант (*С1\*С1*)

В результате этой операции образуется множество кубов:

##### = { x10x1; x101x; x1x11; 110xx; 11x1x; 11xx1; x1x01; x11x1; 010xx; 01x0x; x100x; x10x0; 01xx0; xx000; 01x1x; 01xx1 }

В ходе это операции появилось множество Z.

##### = { 101x0; 10x00 }

Также формируется множество B2 = C1 — Z1.

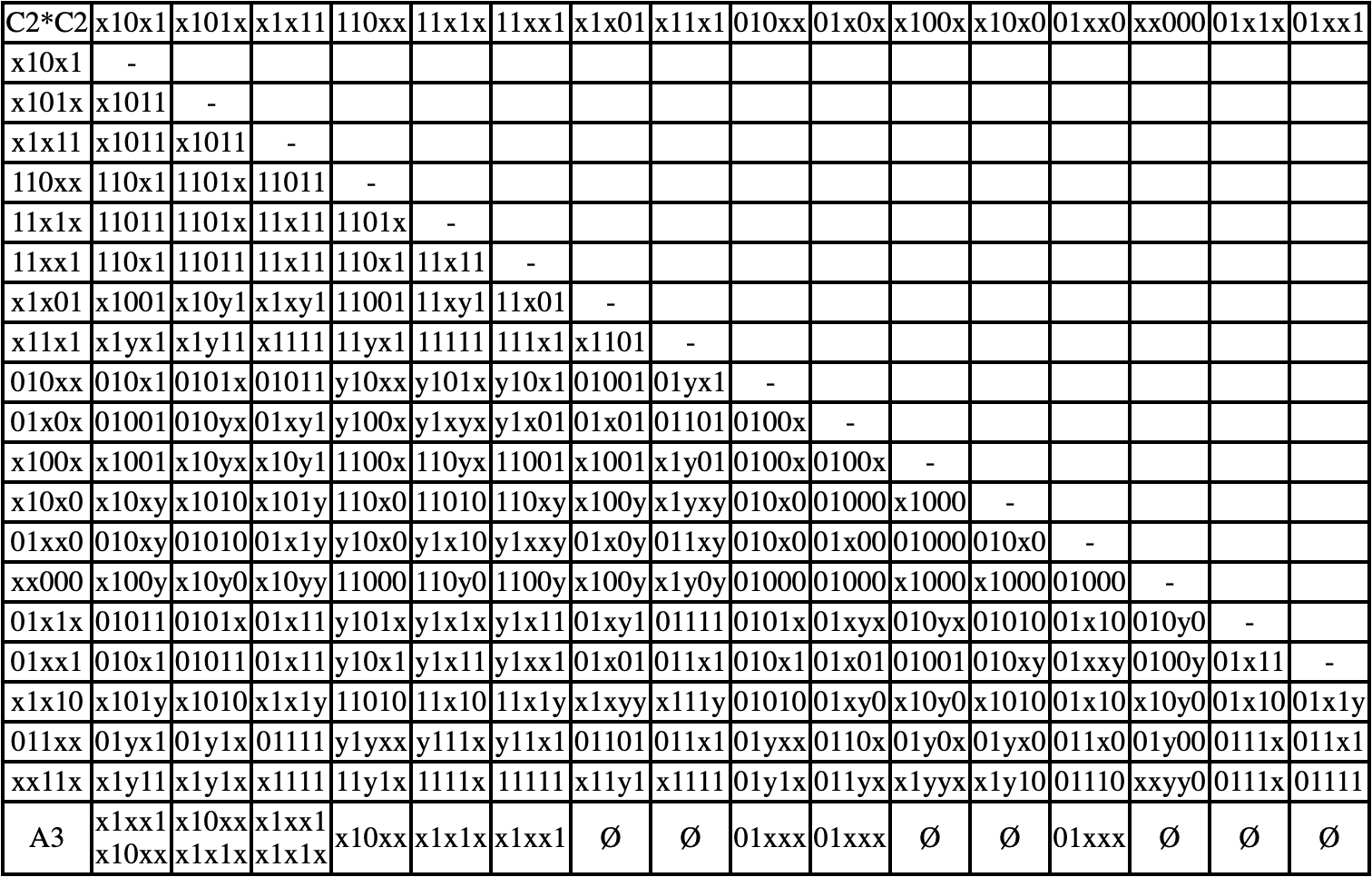
##### = { x1011; 1101x; 110x1; 11x11; x1101; 11x01; 111x1; 0100x; 010x0; 01x00; x0000; x1000; 0101x; 01x01; x1001; 01x11; 110x0; x1x10; 011xx; 1x000; 0x000; 010x1; 1100x; xx11x }

Для следующего шага получения множества Z формируется множество С2 =.

##### С2 = { x10x1; x101x; x1x11; 110xx; 11x1x; 11xx1; x1x01; x11x1; 010xx; 01x0x; x100x; x10x0; 01xx0; xx000; 01x1x; 01xx1; x1x10; 011xx; xx11x }

В таблице 3.1.7 приведён следующий шаг поиска простых импликант – операция *С2\*С2*.

Таблица 3.1.7 – Поиск простых импликант *С2\*С2*



В результате этой операции образуется множество кубов:

= { x1xx1; x10xx; x1x1x; 01xxx }

Также получено множество :

= { 101x0; 10x00; xx000 }

Также формируется множество B3 = C2 — Z2.

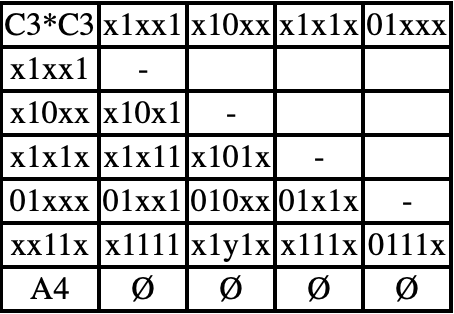
= { x10x1; x101x; x1x11; 110xx; 11x1x; 11xx1; x1x01; x11x1; 010xx; 01x0x; x100x; x10x0; 01xx0; 01x1x; 01xx1; x1x10; 011xx; xx11x }

Для следующего шага получения множества Z формируется множество С2 =.

= { x1xx1; x10xx; x1x1x; 01xxx; xx11x }

В таблице 3.1.8 приведён следующий шаг поиска простых импликант – операция *С3\*С3*.

Таблица 3.1.8 – Поиск простых импликант *С3\*С3*



Получено множество

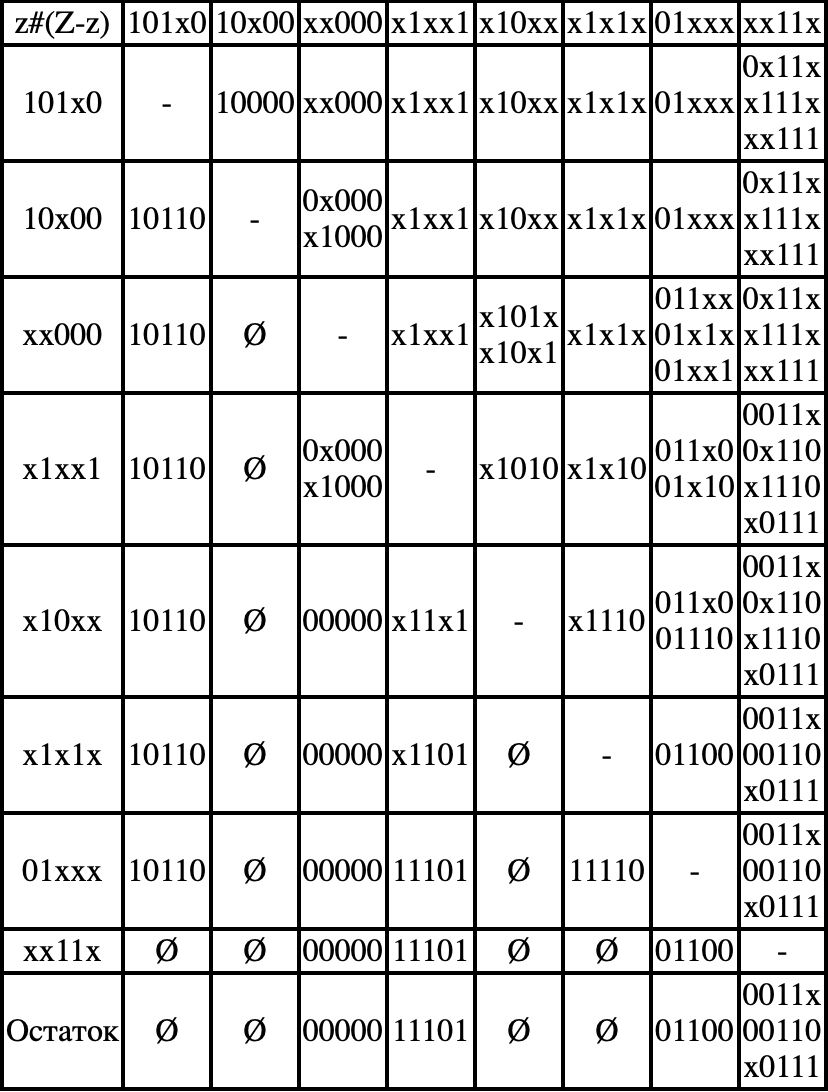
*=* {x1xx1; x10xx; x1x1x; 01xxx; xx11x}

Поскольку |*C4*|≤1, поиск простых импликант заканчивается. Множество простых импликант:

##### *Z* =*Z0* ⋃ *Z1* ⋃ *Z2* = {101x0; 10x00; xx000; x1xx1; x10xx; x1x1x; 01xxx; xx11x}

Следующий этап – поиск *L-*экстремалей на множестве простых импликант (таблица 3.1.9). Для этого используется операция # (решётчатое вычитание).

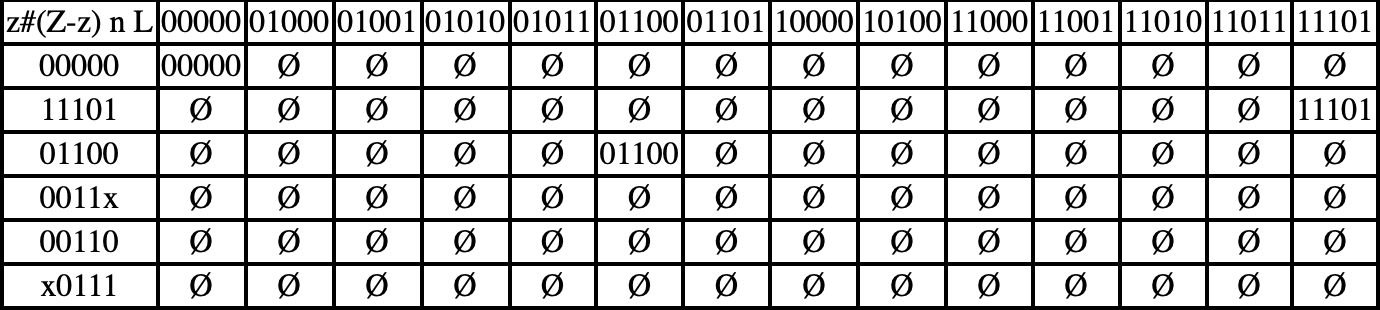
Таблица 3.1.9 – Поиск *L*-экстремалей



В таблице 3.1.9 из каждой простой импликанты поочерёдно вычитаются все остальные простые импликанты *Z#(Z-z)*.

Получили кубы, “подозрительные” на *L*-экстремальность. Проверяем в таблице 3.1.10.

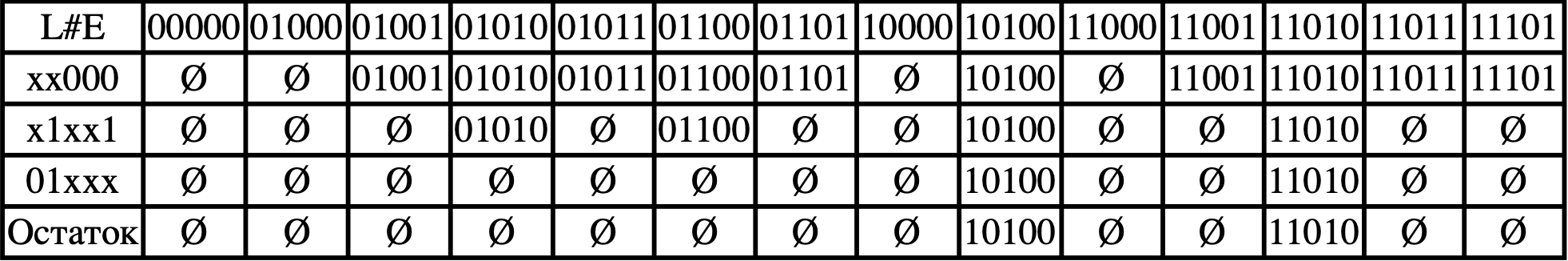
Таблица 3.1.10 – Проверка на *L-*экстремальность



По результатам таблицы 3.1.10, L-экстремалями стали кубы E. Эти кубы обязательно должны войти в минимальное покрытие.

*E* = .

Далее необходимо проанализировать, какие из исходных единичных кубов не покрыты найденной *L*-экстремалью. Анализ осуществляется с помощью таблицы 3.1.11.

Таблица 3.1.11 – Поиск непокрытых наборов

Из таблицы 3.1.11 видно, что L-экстремалями не покрыты два единичных куба (10100, 11010). Чтобы покрыть их, воспользуемся множеством простых импликант, не являющихся L-экстремалями.(Таблица 3.1.12)

Таблица 3.1.12 – Покрытие оставшихся кубов

|  |  |  |
| --- | --- | --- |
| Ž ∩ L | 10100 | 11010 |
| 101x0 | 10100 |  |
| 10x00 | 10100 |  |
| x10xx |  | 11010 |
| x1x1x |  | 11010 |
| xx11x |  |  |

Из таблицы 3.1.12 видно, что кубы 101x0, 10x00, x10xx, x1x1x, xx11x максимально пересекаются с кубами из множества L| и, следовательно, их покрывают (реализуют).

Эти пять кубов добавляются в минимальное покрытие.

Следовательно:

x2

Запишем результат минимизации в логическом базисе ИЛИ-НЕ:

Эффективность минимизаций можно оценить отношением числа входов схем, реализующих переключательную функцию до и после минимизации:

*=*

*=*

*=*

*=*

# **3.2 Логический синтез одноразрядного четверичного сумматора**

Одноразрядный четверичный сумматор – это комбинационное устройство, имеющее 5 двоичных входов (2 разряда одного слагаемого, 2 разряда второго слагаемого и вход переноса) и 3 двоичных выхода.

Принцип работы ОЧС представлен с помощью таблицы истинности (таблица 3.2). Разряды обоих слагаемых закодированы: 0 – 01; 1 – 00; 2 – 10; 3 – 11. В таблице 3.2 выделено 16 безразличных наборов, т.к. со старших выходов ОЧУ не могут прийти коды «2» и «3».

Таблица 3.2 – Таблица истинности ОЧС

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***a₁*** | ***a₂*** | ***b₁*** | ***b₂*** | **p** | **П** | ***S₁*** | ***S₂*** | **Пример операции в четверичной с/c** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1+1+0=02 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1+1+1=03 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1+0+0=01 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1+0+1=02 |
| 0 | 0 | 1 | 0 | 0 | x | x | x | 1+2+0=03 |
| 0 | 0 | 1 | 0 | 1 | x | x | x | 1+2+1=10 |
| 0 | 0 | 1 | 1 | 0 | x | x | x | 1+3+0=10 |
| 0 | 0 | 1 | 1 | 1 | x | x | x | 1+3+1=11 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0+1+0=01 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0+1+1=02 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0+0+0=00 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0+0+1=01 |
| 0 | 1 | 1 | 0 | 0 | x | x | x | 0+2+0=02 |
| 0 | 1 | 1 | 0 | 1 | x | x | x | 0+2+1=03 |
| 0 | 1 | 1 | 1 | 0 | x | x | x | 0+3+0=03 |
| 0 | 1 | 1 | 1 | 1 | x | x | x | 0+3+1=10 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2+1+0=03 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 2+1+1=10 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2+0+0=02 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 2+0+1=03 |
| 1 | 0 | 1 | 0 | 0 | x | x | x | 2+2+0=10 |
| 1 | 0 | 1 | 0 | 1 | x | x | x | 2+2+1=11 |
| 1 | 0 | 1 | 1 | 0 | x | x | x | 2+3+0=11 |
| 1 | 0 | 1 | 1 | 1 | x | x | x | 2+3+1=12 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 3+1+0=10 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 3+1+1=11 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 3+0+0=03 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 3+0+1=10 |
| 1 | 1 | 1 | 0 | 0 | x | x | x | 3+2+0=11 |
| 1 | 1 | 1 | 0 | 1 | x | x | x | 3+2+1=12 |
| 1 | 1 | 1 | 1 | 0 | x | x | x | 3+3+0=12 |
| 1 | 1 | 1 | 1 | 1 | x | x | x | 3+3+1=13 |

Минимизацию функций П, S1, S2 проведём с помощью карт Карно. Для функции *П*заполненная карта приведена на рисунке 3.2.1. В рисунках 3.2.1 - 3.2.3 символом «х» отмечены наборы, на которых функция может принимать произвольное значение (безразличные наборы).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 | 0 | 0 | 0 | 0 | x | x | x | x |
| 01 | 0 | 0 | 0 | 0 | x | x | x | x |
| 11 | 1 | 1 | 1 | 0 | x | x | x | x |
| 10 | 0 | 1 | 0 | 0 | x | x | x | x |

Рисунок 3.2.1 – Минимизация функции *П*при помощи карты Карно

Следовательно:

Запишем результат минимизации в логическом базисе И, НЕ:

***Минимизация функции S1***

Для функции S1заполненная карта приведена на рисунке 3.2.2.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 | 1 | 1 | 1 | 0 | x | x | x | x |
| 01 | 0 | 1 | 0 | 0 | x | x | x | x |
| 11 | 0 | 0 | 0 | 1 | x | x | x | x |
| 10 | 1 | 0 | 1 | 1 | x | x | x | x |

Рисунок 3.2.2 – Минимизация функциипри помощи карты Карно

Следовательно:

Запишем результат минимизации в логическом базисе И, НЕ:

**Минимизация функции S2**

Для функции S2заполненная карта приведена на рисунке 3.2.3.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 | 0 | 1 | 0 | 0 | x | x | x | x |
| 01 | 0 | 0 | 0 | 1 | x | x | x | x |
| 11 | 1 | 0 | 1 | 1 | x | x | x | x |
| 10 | 1 | 1 | 1 | 0 | x | x | x | x |

Рисунок 3.2.3 – Минимизация функциипри помощи карты Карно

Следовательно:

Запишем результат минимизации в логическом базисе И, НЕ:

Эффективность минимизаций можно оценить отношением числа входов схем, реализующих переключательную функцию до и после минимизации:

*=*

*=*

*=*

# **СИНТЕЗ СХЕМЫ ОЧС НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ**

Мультиплексор – это логическая схема, имеющая n информационных входов, m управляющих входов и один выход. При этом должно выполняться условие n = .

Принцип работы мультиплексора состоит в следующем:

На выход мультиплексора может быть пропущен без изменений любой (один) логический сигнал, поступающий на один из информационных входов. Порядковый номер информационного входа, значение которого в данный момент должно быть передано на выход, определяется двоичным кодом, поданным на управляющие входы.

Функции ОЧС зависят от пяти переменных. Удобно взять мультиплексор с тремя управляющими входами, это позволит упростить одну нашу большую функцию от пяти аргументов до восьми функций от одной переменной. Функциональная схема ОЧС на базе мультиплексоров приведена на чертеже ГУИР.400201.530 Э2.3

Таблица 4.1 – таблица истинности ОЧС на базе мультиплексора

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***a₁*** | ***a₂*** | ***b₁*** | ***b₂*** | **p** | **П** |  | ***S₁*** |  | ***S₂*** |  |
| **1** | **2** | **3** | **4** | **5** | **6** |  | **7** |  | **8** |  |
| 0 | 0 | 0 | 0 | 0 | 0 |  | 1 |  | 0 |  |
| 0 | 0 | 0 | 0 | 1 | 0 | *0* | 1 |  | 1 |  |
| 0 | 0 | 0 | 1 | 0 | 0 |  | 0 |  | 0 |  |
| 0 | 0 | 0 | 1 | 1 | 0 |  | 1 |  | 0 |  |
| 0 | 0 | 1 | 0 | 0 | x |  | x |  | x |  |
| 0 | 0 | 1 | 0 | 1 | x |  | x |  | x |  |
| 0 | 0 | 1 | 1 | 0 | x |  | x |  | x |  |
| 0 | 0 | 1 | 1 | 1 | x |  | x |  | x |  |
| 0 | 1 | 0 | 0 | 0 | 0 |  | 0 |  | 0 |  |
| 0 | 1 | 0 | 0 | 1 | 0 | *0* | 1 |  | 0 |  |
| 0 | 1 | 0 | 1 | 0 | 0 |  | 0 |  | 1 |  |
| 0 | 1 | 0 | 1 | 1 | 0 |  | 0 |  | 0 |  |
| 0 | 1 | 1 | 0 | 0 | x |  | x |  | x |  |
| 0 | 1 | 1 | 0 | 1 | x |  | x |  | x |  |
| 0 | 1 | 1 | 1 | 0 | x |  | x |  | x |  |
| 0 | 1 | 1 | 1 | 1 | x |  | x |  | x |  |
| 1 | 0 | 0 | 0 | 0 | 0 |  | 1 |  | 1 |  |
| 1 | 0 | 0 | 0 | 1 | 1 |  | 0 |  | 1 |  |
| 1 | 0 | 0 | 1 | 0 | 0 |  | 1 |  | 0 |  |
| 1 | 0 | 0 | 1 | 1 | 0 |  | 0 |  | 1 |  |
| 1 | 0 | 1 | 0 | 0 | x |  | x |  | x |  |
| 1 | 0 | 1 | 0 | 1 | x |  | x |  | x |  |
| 1 | 0 | 1 | 1 | 0 | x |  | x |  | x |  |
| 1 | 0 | 1 | 1 | 1 | x |  | x |  | x |  |
| 1 | 1 | 0 | 0 | 0 | 1 |  | 0 |  | 1 |  |
| 1 | 1 | 0 | 0 | 1 | 1 |  | 0 |  | 0 |  |
| 1 | 1 | 0 | 1 | 0 | 0 |  | 1 |  | 1 |  |
| 1 | 1 | 0 | 1 | 1 | 1 |  | 0 |  | 1 |  |
| 1 | 1 | 1 | 0 | 0 | x |  | x |  | x |  |
| 1 | 1 | 1 | 0 | 1 | x |  | x |  | x |  |
| 1 | 1 | 1 | 1 | 0 | x |  | x |  | x |  |
| 1 | 1 | 1 | 1 | 1 | x |  | x |  | x |  |

# **5 ЛОГИЧЕСКИЙ СИНТЕЗ ПРЕОБРАЗОВАТЕЛЯ МНОЖИТЕЛЯ (ПМ)**

Преобразователь множителя (ПМ) служит для исключения из множителя диад 11 и 10, заменяя их на триады и , соответственно.

Функциональная схема ПМ приведена на чертеже ГУИР.400201.530 Э2.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Входная диада*** | | ***Младший бит*** | ***Знак*** | ***Выходная диада*** | |
| Qn | Qn-1 | Qn-2 | P | S1 | S2 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

Таблица 5.1 Таблица истинности ПМ.

Проведём минимизацию P при помощи карты Карно:

Qn-1Qn-2

Qn

1

0

10

11

01

00

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 1 | 1 | 1 |

Рисунок 5.1 – Минимизация функции *P*при помощи карты Карно

P = Qn

Видно, что S1 не минимизируется, поэтому Проведём минимизацию S2при помощи карты Карно:

Qn-1Qn-2

Qn

10

11

01

00

0

|  |  |  |  |
| --- | --- | --- | --- |
| 00  01  11  10  0  1 | 1 |  | 1 |
|  | 1 |  | 1 |

Рисунок 5.2 – Минимизация функции *S2* при помощи карты Карно

# **6 ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ ВРЕМЕННЫЕ ЗАТРАТЫ НА УМНОЖЕНИЕ**

Процесс умножения можно разбить на несколько этапов:

* преобразование разряда множителя;
* формирование дополнительного кода множимого в ФДК, если это необходимо;
* умножение дополнительного кода множимого в ОЧУ;
* добавление полученного произведения в аккумулятор посредством ОЧС;
* сдвиг в аккумуляторе и регистре множителя.

Процессы умножения в ОЧУ происходят параллельно. Все ОЧУ отрабатывают одновременно. После этого результат передаётся в блок ОЧС для обработки. В этот момент начинает работу первый ОЧС. После того, как отработает первый ОЧС, он может незамедлительно передать значение переноса во второй ОЧС, который сможет приступить к сложению. Таким образом, все ОЧС отработают последовательно. Формула для расчёта временных затрат на умножение имеет следующий вид:

, где

*n*– количество разрядов множителя;

*m* – количество разрядов множимого;

– время сдвига частичной суммы;

– время умножения на ОЧУ;

– время формирования единицы переноса в ОЧС;

– время преобразования множителя;

– время формирования дополнительного кода множимого.

Подсчитать конкретное время выполнения умножения можно зная характеристики основных узлов конкретного сумматора-умножителя.

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсовой работы были выполнены первоначально заданные цели, а именно разработана структурная схема сумматора-умножителя первого типа и функциональные схемы основных узлов данного устройства. Для уменьшения стоимости логических схем, переключательные функции были минимизированы различными способами. Такой подход позволил выявить достоинства и недостатки этих алгоритмов.

В качестве главного достоинства минимизации картами Карно-Вейча можно выделить простоту и минимальные затраты времени. Однако применение данного способа для функций многих переменных будет затруднительно. Для минимизации функций многих переменных удобно использовать алгоритм Рота, который полностью формализует алгоритмы минимизации и делает минимизацию доступной для выполнения компьютерной программой. В то же время, минимизация алгоритмом Рота вручную может быть очень времязатратной, если функция принимает большое количество единичных и безразличных наборов.

Функциональные схемы были построены в различных логических базисах. Это позволило закрепить теоретические знания основных законов булевой алгебры, например, правило де Моргана. Также можно отметить, что необходимо сократить количество уровней в логической схеме для уменьшения времени работы данного устройства.

Реализация переключательных функций на основе мультиплексоров позволила облегчить процесс минимизации этих функций и упростить функциональную схему одноразрядного четверичного сумматора.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Луцик Ю.А., Лукьянова И.В. – Учебное пособие по курсу "Арифметические и логические основы вычислительной техники". – Минск: БГУИР, 2014 г. – 178 c.

Луцик Ю.А., Лукьянова И.В. – Методические указания к курсовому проекту по курсу “Арифметические и логические основы вычислительной техники”. – Мн.: БГУИР, 2004 г.

Искра, Н. А. Арифметические и логические основы вычислительной техники: пособие / Н. А. Искра, И. В. Лукьянова, Ю. А. Луцик. – Минск: БГУИР, 2016. – 75 с.

# **ПРИЛОЖЕНИЕ А**

*(обязательное)*

Сумматор-умножитель первого типа. Схема электрическая

структурная

# **ПРИЛОЖЕНИЕ Б**

(обязательное)

Одноразрядный четверичный сумматор. Схема электрическая функциональная

# **ПРИЛОЖЕНИЕ В**

(обязательное)

Одноразрядный четверичный умножитель.

Схема электрическая функциональная

# **ПРИЛОЖЕНИЕ Г**

(обязательное)

Одноразрядный четверичный сумматор.

Реализация на мультиплексорах.

Схема электрическая функциональная

# **ПРИЛОЖЕНИЕ Д**

(обязательное)

Преобразователь множителя.

Схема электрическая функциональная

# **ПРИЛОЖЕНИЕ Е**

(обязательное)

Ведомость документов