**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc166161271)

[**1. РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ** 6](#_Toc166161272)

[**1.1** **Перевод сомножителей из десятичной системы счисления в четверичную** 6](#_Toc166161273)

[**1.2** **Нахождение произведения множимого и множителя** 7](#_Toc166161274)

[**2 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫСУММАТОРА-УМНОЖИТЕЛЯ** 10](#_Toc166161275)

[**3 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ** **СУММАТОРА-УМНОЖИТЕЛЯ**……………………………………………….13](#_Toc166161276)

[**3.1 Логический синтез одноразрядного четверичного умножителя-сумматора** 13](#_Toc166161277)

[**3.2 Логический синтез одноразрядного четверичного сумматора** 21](#_Toc166161278)

[**3.3 Логический синтез преобразователя множителя** 24](#_Toc166161279)

[**4 СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ УСТРОЙСТВ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ** 26](#_Toc166161280)

[**5 ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ** 28](#_Toc166161281)

[**ЗАКЛЮЧЕНИЕ** 29](#_Toc166161282)

[**ПРИЛОЖЕНИЕ А** 30](#_Toc166161283)

[**ПРИЛОЖЕНИЕ Б** 31](#_Toc166161284)

[**ПРИЛОЖЕНИЕ В** 32](#_Toc166161285)

[**ПРИЛОЖЕНИЕ Г** 33](#_Toc166161286)

[**ПРИЛОЖЕНИЕ Д** 34](#_Toc166161287)

[**ПРИЛОЖЕНИЕ E** 35](#_Toc166161288)

[**ПРИЛОЖЕНИЕ Ж** 36](#_Toc166161289)

[**ПРИЛОЖЕНИЕ З** 37](#_Toc166161290)

[**ПРИЛОЖЕНИЕ И** 38](#_Toc166161291)

[**ПРИЛОЖЕНИЕ К** 39](#_Toc166161292)

[**ПРИЛОЖЕНИЕ Л** 40](#_Toc166161293)

[**ПРИЛОЖЕНИЕ М** 41](#_Toc166161294)

[**ПРИЛОЖЕНИЕ Н** 41](#_Toc166161294)

# **ВВЕДЕНИЕ**

Курсовое проектирование – неотъемлемая часть образовательного процесса высшего уровня, а также ключевой этап оценки знаний студентов по соответствующей дисциплине. Для многих студентов это первый опыт выполнения работы такого масштаба и значимости. В рамках этого проекта проводятся как теоретические, так и практические исследования по дисциплине "Арифметические и логические основы цифровых устройств", включающие аналитические, расчетные и экспериментальные задания, а также проектирование и разработку графической документации.

Цель данного курсового проекта – создание цифрового устройства, а именно двоичного-четверичного сумматора-умножителя (СУ). Сумматор играет ключевую роль в арифметико-логическом устройстве (АЛУ) компьютера, и глубокое понимание его принципов работы критически важно для инженеров. Для достижения этой цели требуется несколько этапов разработки:

− Разработка алгоритма умножения чисел, по которому работает СУ;

− Разработка структурной схемы СУ;

− Разработка функциональной схемы основных узлов структурной схемы СУ;

− Оценка результатов проделанной работы;

− Оформление документации по 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**

Исходные данные:

− исходные сомножители: Мн = 34,71; Мт = 61,23;

− алгоритм умножения: В;

− метод умножения: умножение закодированного двоично-четверичного множимого на два разряда двоичного множителя одновременно в дополнительных кодах;

− кодирование четверичных цифр множимого для перехода к двоично-четверичной системе кодирования: 04 – 01, 14 – 11, 24 – 00, 34 – 10;

− тип синтезируемого умножителя: умножитель 2-го типа на базе ОЧУС, ОЧС и регистра результат;

## **1.1** **Перевод сомножителей из десятичной системы счисления в четверичную**

**Множимое**: 34,71

Целая часть:

Дробная часть:

|  |  |
| --- | --- |
| \* | 0,71  4 |
| \* | 2,84  4 |
| \* | 3,36  4 |
|  | 1,44 |

В соответствии с заданной кодировкой множимого:

= 000100,001011

**Множитель**: 49,56

Целая часть:

|  |  |
| --- | --- |
| \* | 0,23  4 |
| \* | 0,92  4 |
| \* | 3,68  4 |
|  | 2,72 |

Дробная часть:

= 331,032

В соответствии с заданной кодировкой множителя:

= 111101,001110

## **1.2** **Нахождение произведения множимого и множителя**

Запишем сомножители в форме с плавающей запятой в прямом коде:

= 0,000100001011 = 0.0010 + - закодировано по заданию.

= 1,100011000111 = 0.0011 + - закодировано традиционно.

Порядок произведения:

|  |  |  |
| --- | --- | --- |
|  | = | 0.0010 |
|  | = | 0.0011 |
|  | = | 0.1110 |

Знак определяется операцией сложения по модулю два знаков сомножителей:

зн Мн ⊕ зн Мт = 0 ⊕ 0 = 0.

При умножении чисел в дополнительных кодах диады 10() и 11() преобразуются в диады и 0 соответственно. После преобразования диады мы проверяем старший разряд младшей диады на единицу, при ее наличии суммируем ее с преобразованной диадой. Преобразованный множитель имеет вид: =

Перемножение мантисс по алгоритму “В” представлено в таблице 1.1.

Таблица 1.1 — Перемножение мантисс

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Четверичная сс | | Двоично-четверичная сс | | Комментарии |
| 0, | 000000000000 | 0, | 000000000000000000000000 |  |
| 0, | 000000202231 | 0, | 101010101010100100001011 | =Мн\*(1) |
| 0, | 000000202231 | 0, | 101010101010100100001011 |  |
| 0, | 000002022310 | 0, | 010101010101001000010110 | \* 4 = |
| 0, | 000020223100 | 0, | 101010101010010000101100 | \* 4 |
| 3, | 333333131103 | 1, | 101010101010111011110110 | =Мн\*(0) |
| 0, | 000020020203 | 0, | 101010100001010001000110 |  |
| 0, | 000200202030 | 0, | 010101000010100010001100 | \* 4 |
| 0, | 000000202231 | 0, | 101010101010100100001011 | =Мн\*(2) |
| 0, | 000201010321 | 0, | 010101010011011101100011 |  |
| 0, | 002010103210 | 0, | 101010100110111011000110 | \* 4 |
| 0, | 000000202231 | 0, | 101010101010100100001011 | =Мн\*(-2) |
| 0, | 002010312101 | 0, | 010100011101101100110111 |  |
| 0, | 020103121010 | 0, | 011001110110110011011101 | \* 4 = |
| 0, | 201031210100 | 0, | 000101101011001101110101 | \* 4 |
| 3, | 333332322212 | 1, | 101010101000100000001100 | =Мн\*(-1) |
| 0, | 201030132312 | 0, | 000111011001111000101100 |  |

После окончания умножения необходимо оценить погрешность вычислений. Для этого полученное произведение (\* = 0,201030132312, РМн ∙ Мт = 6) приводится к нулевому порядку, а затем переводится в десятичную систему счисления:

\* = 0,201030132312 = 0;

\* = 2124,48.

Результат прямого перемножения:

\* = 34,71 \* 61,23 = 2125,29.

Абсолютная погрешность:

= 2125,29 - 2124,48= 0,81.

Относительная погрешность:

= = = 0,00038. ( = 0,038%).

Эта погрешность получена за счёт приближённого перевода (до заданного количества знаком после запятой) из десятичной системы счисления в четверичную обоих сомножителей, а также за счёт округления полученного результата произведения (количества разрядов).

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **2 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫСУММАТОРА-УМНОЖИТЕЛЯ**

Если устройство работает как сумматор, то оба слагаемых последовательно (за два такта) заносятся в регистр множимого, а на управляющий вход формирователя дополнительного кода поступает «1». Необходимо обеспечить выполнение алгоритма сложения чисел, представленных в форме с плавающей запятой, базируясь на схеме умножителя, реализующего заданный алгоритм умножения.

Первое слагаемое переписывается в регистр результата под действием управляющих сигналов, поступающих на входы h всех ОЧУС (рисунок 2.1).

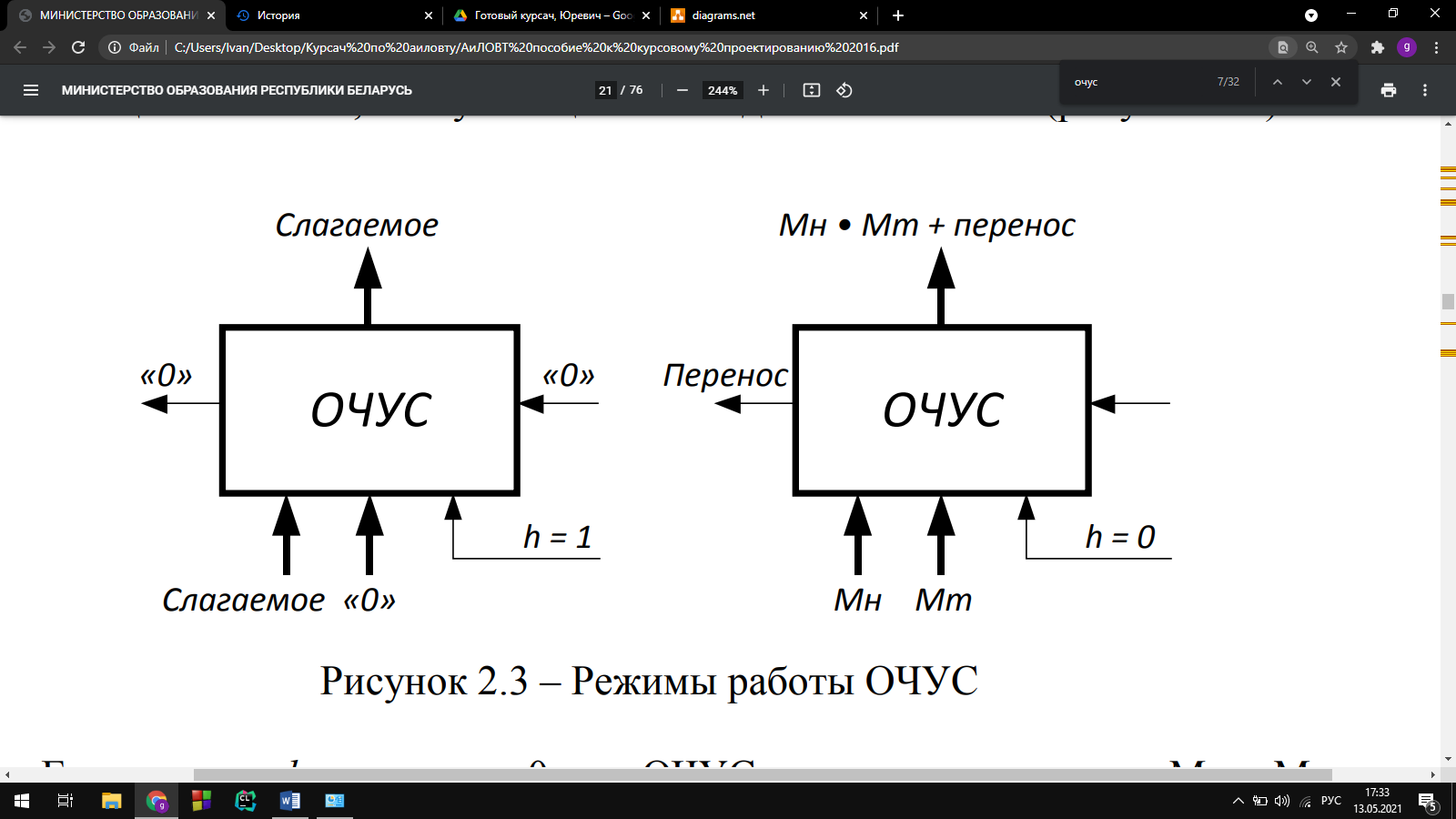


Рисунок 2.1 – Режимы работы ОЧУС

Если на вход h поступает «0», то ОЧУС перемножает разряды Мн и Мт и добавляет к полученному результату перенос из предыдущего ОЧУС.

На втором такте второе слагаемое из регистра множимого через цепочку ОЧУС попадает на входы ОЧС и складывается с первым слагаемым, хранящимся в регистре результата.

Сумма хранится в регистре результата. Разрядность регистра результата должна быть на единицу больше, чем разрядность исходных слагаемых, чтобы предусмотреть возможность возникновения при суммировании переноса.

Если устройство работает как умножитель, то множимое и множитель помещаются в соответствующие регистры, а на управляющий вход ФДК F2 поступает «0».

Диада множителя поступает на входы преобразователя множителя. Единица переноса в следующую диаду, если она возникает, должна быть добавлена к следующей диаде множителя (выход 1 ПМ) в следующем такте, т. е. должна храниться на триггере до следующего такта.

В регистре множителя после каждого такта умножения содержимое сдвигается на два двоичных разряда, и в конце умножения регистр обнуляется. Это позволяет использовать регистр множителя для хранения младших разрядов произведения при умножении по алгоритму «Б».

Выход 2 ПМ переходит в единичное состояние, если текущая диада содержит отрицание (). 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 отражён в таблице 2.1.

На выходах 3 и 4 ПМ формируются диады преобразованного множителя, которые поступают на входы ОЧУС вместе с диадами множимого. На трёх выходах ОЧУС формируется результат умножения диад Мн·Мт плюс перенос из предыдущего ОЧУС. Максимальной цифрой в диаде преобразованного множителя является двойка, поэтому перенос, формируемый ОЧУС, может быть только двоичным («0» или «1»):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | ∙ | 2 | = | 12 | (+1 в случае переноса из предыдущего ОЧУС) | |
| max |  | max |  | max | |  | |
| Мн |  | Мт |  | Перенос | |  | |

Так как на входы ОЧУС из регистра Мт не могут поступить коды «3», в таблице истинности работы ОЧУС будут содержаться 16 безразличных входных наборов.

Частичные произведения, получаемые на выходах ОЧУС, складываются с накапливаемой частичной суммой из регистра результата с помощью цепочки ОЧС (на первом такте выполняется сложение с нулём).

Частичные суммы хранятся в регистре результата и регистре множителя, т. к. алгоритм умножения «А» предполагает возможность синхронного сдвига этих регистров. Количество тактов умножения определяется разрядностью Мт.

Формирователь дополнительного кода при поступлении на F2 единицы формирует на выходе дополнительный код слагаемого. Если же на F2 подается ноль, то ФДК имеет два режима работы, которые зависят от значения, которое подается на F1. Если на вход подается единица, то есть диада множителя преобразовалась в 0, то формирователь дополнительного кода на выходе формирует дополнительный код множимого с измененным знаком. В противном случает на выходе формируется дополнительный код множимого без изменения знака.

Таблица 2.1 – Режим работы формирователя дополнительного кода

|  |  |  |
| --- | --- | --- |
| Сигналы на входах ФДК | | Результат на выходах ФДК |
|  |  |
| 0 | 0 | Дополнительный код множимого |
| 0 | 1 | Дополнительный код слагаемого |
| 1 | 0 | Меняется знак Мн |

Структурная электрическая схема преобразователя множителя представлена в приложении А.

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

## 

## **3.1 Логический синтез одноразрядного четверичного умножителя-сумматора**

ОЧУС – это комбинационное устройство, имеющее шесть входов (два разряда из регистра множимого, два разряда из регистра множителя, вход переноса и управляющий вход h) и три выхода.

Принцип работы ОЧУС представлен с помощью таблицы истинности (таблица 3.1.1).

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

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

Управляющий вход *h* определяет тип операции:

«0» – вывод результата умножения закодированных цифр с добавлением переноса из предыдущего ОЧУС, перенос в следующий ОЧУС.

«1» – вывод без изменения значения разрядов, поступивших из регистра множимого, перенос *из* и *в* ОЧУС равны нулю.

В таблице 3.1.1 выделены безразличные наборы, так как на входы ОЧУС из разрядов множителя не может поступить код 11, при работе ОЧУС как сумматора на вход переноса не может поступить единица, а при умножении на нуль или единицу на вход переноса также не может поступить единица.

Таблица 3.1.1 — Таблица истинности ОЧУС

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Пер | Мн | | Мт | | Упр | Перенос | Результат | | Результат в четверичной |
|  |  |  |  |  | *h* | *П* |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2\*0+0=00 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Выход - код "02" |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2\*1+0=02 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Выход - код "02" |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 2\*2+0=10 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | Выход - код "02" |
| 0 | 0 | 0 | 1 | 1 | 0 | x | x | x | 2\*3+0=12 |
| 0 | 0 | 0 | 1 | 1 | 1 | x | x | x | Выход - код "02" |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0\*0+0=00 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | Выход - код "00" |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0\*1+0=00 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | Выход - код "00" |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0\*2+0=00 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | Выход - код "00" |
| 0 | 0 | 1 | 1 | 1 | 0 | x | x | x | 0\*3+0=00 |
| 0 | 0 | 1 | 1 | 1 | 1 | x | x | x | Выход - код "00" |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3\*0+0=00 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Выход - код "03" |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 3\*1+0=03 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | Выход - код "03" |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 3\*2+0=12 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | Выход - код "03" |
| 0 | 1 | 0 | 1 | 1 | 0 | x | x | x | 3\*3+0=21 |
| 0 | 1 | 0 | 1 | 1 | 1 | x | x | x | Выход - код "03" |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1\*0+0=00 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | Выход - код "01" |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1\*1+0=01 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | Выход - код "01" |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1\*2+0=02 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | Выход - код "01" |
| 0 | 1 | 1 | 1 | 1 | 0 | x | x | x | 1\*3+0=03 |
| 0 | 1 | 1 | 1 | 1 | 1 | x | x | x | Выход - код "01" |
| 1 | 0 | 0 | 0 | 0 | 0 | x | x | x | 2\*0+1=01 |
| 1 | 0 | 0 | 0 | 0 | 1 | x | x | x | Выход - код "02" |
| 1 | 0 | 0 | 0 | 1 | 0 | x | x | x | 2\*1+1=03 |
| 1 | 0 | 0 | 0 | 1 | 1 | x | x | x | Выход - код "02" |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 2\*2+1=11 |
| 1 | 0 | 0 | 1 | 0 | 1 | x | x | x | Выход - код "02" |
| 1 | 0 | 0 | 1 | 1 | 0 | x | x | x | 2\*3+1=13 |
| 1 | 0 | 0 | 1 | 1 | 1 | x | x | x | Выход - код "02" |
| 1 | 0 | 1 | 0 | 0 | 0 | x | x | x | 0\*0+1=01 |
| 1 | 0 | 1 | 0 | 0 | 1 | x | x | x | Выход - код "00" |
| 1 | 0 | 1 | 0 | 1 | 0 | x | x | x | 0\*1+1=01 |
| 1 | 0 | 1 | 0 | 1 | 1 | x | x | x | Выход - код "00" |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0\*2+1=01 |
| 1 | 0 | 1 | 1 | 0 | 1 | x | x | x | Выход - код "00" |
| 1 | 0 | 1 | 1 | 1 | 0 | x | x | x | 0\*3+1=01 |
| 1 | 0 | 1 | 1 | 1 | 1 | x | x | x | Выход - код "00" |
| 1 | 1 | 0 | 0 | 0 | 0 | x | x | x | 3\*0+1=01 |
| 1 | 1 | 0 | 0 | 0 | 1 | x | x | x | Выход - код "03" |
| 1 | 1 | 0 | 0 | 1 | 0 | x | x | x | 3\*1+1=10 |
| 1 | 1 | 0 | 0 | 1 | 1 | x | x | x | Выход - код "03" |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 3\*2+1=13 |
| 1 | 1 | 0 | 1 | 0 | 1 | x | x | x | Выход - код "03" |
| 1 | 1 | 0 | 1 | 1 | 0 | x | x | x | 3\*3+1=22 |
| 1 | 1 | 0 | 1 | 1 | 1 | x | x | x | Выход - код "03" |
| 1 | 1 | 1 | 0 | 0 | 0 | x | x | x | 1\*0+1=01 |
| 1 | 1 | 1 | 0 | 0 | 1 | x | x | x | Выход - код "01" |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1\*1+1=02 |
| 1 | 1 | 1 | 1 | 0 | 1 | x | x | x | Выход - код "01" |
| 1 | 1 | 1 | 1 | 1 | 0 | x | x | x | 1\*2+1=03 |
| 1 | 1 | 1 | 1 | 1 | 1 | x | x | x | Выход - код "01" |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1\*3+1=10 |
| 1 | 1 | 1 | 0 | 1 | 1 | x | x | x | Выход - код "01" |

Минимизация функции *,* ипроведена при помощи минимизирующих карт Карно на рисунке 3.1.1 – 3.1.3. В рисунках 3.1.1 – 3.1.3 символом «х» отмечены наборы, на которых функция может принимать произвольное значение (безразличные наборы).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 1 |  |  |  |  |  |
| 1 | x | x |  | 1 |  |  | 1 |
| x | x | x | x | x | x | x | x |
| x | x | x | x | x | x | x | x |
| 1 | 1 | 1 | 1 |  |  |  |  |
| x | x | x | x | x | x | x | x |
| x | x | x | x | x | x | x | x |
|  | 1 | 1 |  |  |  |  |  |

Рисунок 3.1.1 — Минимизация функции *Q1* картой Карно-Вейче

Функция минимизирована по алгоритму Рота.

Для этого определить множество единичных и безразличных кубов:

*L* = {010001, 010010, 010011, 010101, 011001, 011010, 011011, 011101, 100100, 101100, 110100};

*N =* {000110, 000111, 001110, 001111, 010110, 010111, 011110, 011111, 100000, 100001, 100010, 100011, 100101, 100110, 100111, 101000, 101001, 101010, 101011, 101101, 101110, 101111, 110000, 110001, 110010, 110011, 110101, 110110, 110111, 111000, 111001, 111101, 111110, 111111, 111011};

Склеиваем набор безразличных кубов N: *N* = {111011, 11100x, 1xx110, 1100xx, 0xx11x, 10x0xx, 1xx1x1};

Сформируем множество = ;

= {010001, 010010, 010011, 010101, 011001, 0110101, 011011, 011101, 100100, 101100, 110100, 111011, 11100x, 1xx110, 1100xx, 0xx11x, 10x0xx, 1xx1x1};

Первым этапом алгоритма Рота является нахождение множества простых импликант *Z*. Для его реализации используется операция умножения «\*» над множествами , и так до . Операция умножения прекращается, когда мощность очередного множества становится меньше двух. При этом множества = , где - множество результатов операции умножения «\*» над , а = .

Первый шаг умножения (С0\*С0) представлен в приложении Б.

В результате умножения следующие множества:  
 A1 = {0100x1; 010x01; 01x001; x10001; 01001x; 01x010; x10010; 010x10; 01x011; x10011; 010x11; 01x101; 0101x1; x10101; 0110x1; 011x01; x11001; 01101x; 011x10; x11011; 011x11; 0111x1; x11101; 10x100; 1x0100; 1001x0; 100x00; 10010x; 1011x0; 101x00; 10110x; 1101x0; 110x00; 11010x; 1110x1; 11x011; 1x1011; 111x11; 11x00x; 1x100x; 111x01; 110x10; xxx110; 10xx10; 1xx11x; 1x00xx; 110xx1; xxx111; 10xxx1}  
 Z0 = {Ø}  
 B1 = {010001; 010010; 010011; 010101; 011001; 011010; 011011; 011101; 100100; 101100; 110100; 111011; 11100x; 1xx110; 1100xx; 0xx11x; 10x0xx; 1xx1x1}

= , где мы применяем операцию склеивания, после чего получится множество:  
 C1 = {0100x1; 010x01; 01x001; x10001; 01001x; 01x010; x10010; 010x10; 01x011; x10011; 010x11; 01x101; 0101x1; x10101; 0110x1; 011x01; x11001; 01101x; 011x10; x11011; 011x11; 0111x1; x11101; 10x100; 1x0100; 1001x0; 100x00; 10010x; 1011x0; 101x00; 10110x; 1101x0; 110x00; 11010x; 1110x1; 11x011; 1x1011; 111x11; 11x00x; 1x100x; 111x01; 110x10; xxx110; 10xx10; 1xx11x; 1x00xx; 110xx1; xxx111; 10xxx1; 0xx11x; 10x0xx; 1xx1x1}

Перед выполнением \* разберем операцию:

\* = = .

Результат выполнения \* приведен в приложении В.  
 A2 = {010xx1; 01x0x1; x100x1; 01xx01; x10x01; x1x001; 01x01x; x1001x; 010x1x; 01xx10; x10x10; x1x011; 01xx11; x10x11; 01x1x1; x1x101; x101x1; 011xx1; x110x1; x11x01; 011x1x; x11x11; x111x1; 10x1x0; 10x10x; 10xx00; 1x01x0; 1x0x00; 1x010x; 100xx0; 1001xx; 100x0x; 1011xx; 101xx0; 101x0x; 110xx0; 1101xx; 110x0x; 11x0x1; 1x10x1; 111xx1; 11xx11; 1xx011; 1x1x11; 1xx00x; 11xx01; 1x1x01; xxx11x; 10xx1x; 1x0x1x; 1x0xx1}

##### Z1 = {Ø} B2 = {0100x1; 010x01; 01x001; x10001; 01001x; 01x010; x10010; 010x10; 01x011; x10011; 010x11; 01x101; 0101x1; x10101; 0110x1; 011x01; x11001; 01101x; 011x10; x11011; 011x11; 0111x1; x11101; 10x100; 1x0100; 1001x0; 100x00; 10010x; 1011x0; 101x00; 10110x; 1101x0; 110x00; 11010x; 1110x1; 11x011; 1x1011; 111x11; 11x00x; 1x100x; 111x01; 110x10; xxx110; 10xx10; 1xx11x; 1x00xx; 110xx1; xxx111; 10xxx1; 0xx11x; 10x0xx; 1xx1x1} = , где проводим операцию склеивания и получаем: C2 = {010xx1; 01x0x1; x100x1; 01xx01; x10x01; x1x001; 01x01x; x1001x; 010x1x; 01xx10; x10x10; x1x011; 01xx11; x10x11; 01x1x1; x1x101; x101x1; 011xx1; x110x1; x11x01; 011x1x; x11x11; x111x1; 10x1x0; 10x10x; 10xx00; 1x01x0; 1x0x00; 1x010x; 100xx0; 1001xx; 100x0x; 1011xx; 101xx0; 101x0x; 110xx0; 1101xx; 110x0x; 11x0x1; 1x10x1; 111xx1; 11xx11; 1xx011; 1x1x11; 1xx00x; 11xx01; 1x1x01; xxx11x; 10xx1x; 1x0x1x; 1x0xx1; 1x00xx; 10xxx1; 10x0xx; 1xx1x1}

Результат выполнения \* приведен в приложении Г.  
 A3 = {01xxx1; x10xx1; x1x0x1; x1xx01; 01xx1x; x10x1x; x1xx11; x1x1x1; x11xx1; 10x1xx; 10xxx0; 10xx0x; 1x01xx; 1x0xx0; 1x0x0x; 100xxx; 101xxx; 110xxx; 1xx0x1; 11xxx1; 1x1xx1; 1xxx11; 10x0xx; 1x00xx; 1xxx01; 1x0x1x; 10xx1x}  
 Z2 = {Ø}  
 B3 = {010xx1; 01x0x1; x100x1; 01xx01; x10x01; x1x001; 01x01x; x1001x; 010x1x; 01xx10; x10x10; x1x011; 01xx11; x10x11; 01x1x1; x1x101; x101x1; 011xx1; x110x1; x11x01; 011x1x; x11x11; x111x1; 10x1x0; 10x10x; 10xx00; 1x01x0; 1x0x00; 1x010x; 100xx0; 1001xx; 100x0x; 1011xx; 101xx0; 101x0x; 110xx0; 1101xx; 110x0x; 11x0x1; 1x10x1; 111xx1; 11xx11; 1xx011; 1x1x11; 1xx00x; 11xx01; 1x1x01; xxx11x; 10xx1x; 1x0x1x; 1x0xx1; 1x00xx; 10xxx1; 10x0xx; 1xx1x1}  
 = , после склеивания:  
 C3 = {01xxx1; x10xx1; x1x0x1; x1xx01; 01xx1x; x10x1x; x1xx11; x1x1x1; x11xx1; 10x1xx; 10xxx0; 10xx0x; 1x01xx; 1x0xx0; 1x0x0x; 100xxx; 101xxx; 110xxx; 1xx0x1; 11xxx1; 1x1xx1; 1xxx11; 10x0xx; 1x00xx; 1xxx01; 1x0x1x; 10xx1x; 1xx00x; xxx11x; 1x0xx1; 10xxx1; 1xx1x1}

Результат выполнения \* приведен в приложении Д.  
 A4 = {x1xxx1; 10xxxx; 1x0xxx; 1xxxx1}  
 Z3 = {01xx1x; x10x1x; 1xx00x; xxx11x}  
 B4 = {01xxx1; x10xx1; x1x0x1; x1xx01; x1xx11; x1x1x1; x11xx1; 10x1xx; 10xxx0; 10xx0x; 1x01xx; 1x0xx0; 1x0x0x; 100xxx; 101xxx; 110xxx; 1xx0x1; 11xxx1; 1x1xx1; 1xxx11; 10x0xx; 1x00xx; 1xxx01; 1x0x1x; 10xx1x; 1x0xx1; 10xxx1; 1xx1x1}

= , после склеивания:  
 C4 = {x1xxx1; 10xxxx; 1x0xxx; 1xxxx1}

Результат выполнения \* приведен в приложении Е.

##### A5 = {Ø} Z = {01xx1x; x10x1x; 1xx00x; xxx11x; x1xxx1; 10xxxx; 1x0xxx; 1xxxx1} B5 = {Ø} C5 = {Ø}

Cледующим этапом алгоритма Рота является поиск *L*-экстремалей на множестве простых импликант. Для этого с помощью операции вычитания «#» находим кубы, которые могут являться *L*-экстремалями. Операция *z*#(*Z*-*z*) для поиска таких кубов представлена в приложении Ж.

Получено множество кубов подозрительных на *L*-экстремаль:  
 E = {01xx1x; x1xxx1; 10xxxx; 1x0xxx}  
 Z´ = Z - E = {x10x1x; 1xx00x; xxx11x; 1xxxx1}

Теперь с помощью вычитания *L*#*E* проверяем покрывает ли множество *E* все кубы множества *L*. Операция *L*#*E* для поиска таких кубов представлена в приложении 3.  
 Минимальное покрытие - множество L-экстремалей E = {01xx1x; x1xxx1; 10xxxx; 1x0xxx}

Минимальная функция *f*: .

Приводим функцию к нужному базису:

*.*

Оценим эффективность минимизации функции:

= 1,2.

Функциональная электрическая схема ОЧУС представлена в приложении И.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 |  |  | 1 |  |  |
|  | x | x |  | 1 |  |  | 1 |
| x | x | x | x | x | x | x | x |
| x | x | x | x | x | x | x | x |
|  |  | 1 | 1 |  | 1 |  |  |
| x | x | x | x | x | x | x | x |
| x | x | x | x | x | x | x | x |
|  |  | 1 |  |  | 1 |  |  |

Рисунок 3.1.2 — Минимизация функции картой Карно

Функция имеет вид: .

Приводим функцию к нужному базису:

.

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

*.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  | 1 |
| 1 | x | x | 1 |  |  |  | 1 |
| x | x | x | x | x | x | x | x |
| x | x | x | x | x | x | x | x |
|  |  |  |  |  |  |  |  |
| x | x | x | x | x | x | x | x |
| x | x | x | x | x | x | x | x |
|  |  |  |  |  |  |  |  |

Рисунок 3.1.3 — Склеивание наборов *П* картой Карно-Вейча

Функция имеет вид: .

Приводим функцию к нужному базису:

.

Эффективность минимизации 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. Кодировка слагаемых обоих разрядов: 04 – 01, 14 – 11, 24 – 00, 34 – 10.

Принцип работы ОЧС представлен с помощью таблицы истинности (таблица 3.2.1).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | Операция в четверичной |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2+2+0=10 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2+2+1=11 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2+0+0=02 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 2+0+1=03 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 2+3+0=11 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2+3+1=12 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 2+1+0=03 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 2+1+1=10 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0+2+0=02 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0+2+1=03 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0+0+0=00 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0+0+1=01 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0+3+0=03 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0+3+1=10 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0+1+0=01 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0+1+1=02 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 3+2+0=11 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 3+2+1=12 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 3+0+0=03 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 3+0+1=10 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 3+3+0=12 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 3+3+1=13 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 3+2+0=11 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 3+2+1=12 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1+2+0=03 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1+2+1=10 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1+0+0=01 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1+0+1=02 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1+3+0=10 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1+3+1=11 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1+1+0=02 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1+1+1=03 |

Минимизацию функции*П*, , проведем с помощью карт Вейча (рисунок 3.2.2 - 3.2.4).

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

Рисунок 3.2.2 - минимизирующая карта Вейча для функции *П*

Функция Пимеет вид:

.

Функцию П запишем в нужном базисе И-ИСКЛ.ИЛИ-1:

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

.

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

Рисунок 3.2.3 - минимизирующая карта Вейча для функции

Функция имеет вид:

*.*

Функцию запишем в нужном базисе И-ИСКЛ.ИЛИ-1:

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

*.*

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

p

p

Рисунок 3.2.4 - минимизирующая карта Вейча для функции

Функция имеет вид:

Функцию запишем в нужном базисе И-ИСКЛ.ИЛИ-1:

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

.

Функциональная электрическая схема ОЧС представлена в приложении К.

## **3.3 Логический синтез преобразователя множителя**

Преобразователь множителя(ПМ) служит для исключения из множителя диад 11 и 10, заменяя их на диады 0 и и преобразования возможных диад, с помощью анализа старшего разряда младшей диады, по средствам которого к текущей диаде добавляется 1 или 0. Принцип работы ПМ описывается с помощью таблицы истинности 3.3.1.

Таблица 3.3.1 – таблица истинности ПМ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | *r* | *S* |  |  |
| 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 | 0 | 0 | 0 |

Функции *S*, , минимизируем при помощи минимизирующих карт Карно (рисунки 3.3.2 - 3.3.4).

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

D2r

D1

Рисунок 3.3.2 – минимизирующая карта Карно для *S.*

Минимальная форма для функции *S*:

*S* = .

D2r

D1

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

Рисунок 3.3.3 – минимизирующая карта Карно для *.*

Минимальная форма для функции :

= .

D2r

D1

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

Рисунок 3.3.4 – минимизирующая карта Карно для *.*

Минимальная форма для функции :

= .

Функциональная электрическая схема преобразователя множителя представлена в приложении Л.

# **4 СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ УСТРОЙСТВ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ**

*Мультиплексор* - это логическое устройство, которое принимает *n* информационных входов, *m* управляющих входов и имеет один выход. Условие *m =*  должно быть выполнено.

На выход мультиплексора можно передать любой логический сигнал, поступающий на один из информационных входов, без изменений. Порядковый номер информационного входа, значение которого должно быть передано на выход, определяется двоичным кодом, подаваемым на управляющие входы.

Переключательные функции(ПФ) от пяти переменных, например, ОЧС, могут быть реализованы на мультиплексоре "один из восьми". Управляющее поле такого мультиплексора определяется тремя переменными, что означает, что количество групп с одинаковыми значениями этих переменных составляет восемь. Реализация нескольких ПФ требует отдельного мультиплексора для каждой функции.

Для определения управляющего поля мультиплексора используем переменные , и .

Таблица истинности для синтеза ПФ ОЧС приведена в таблице 4.1.

Таблица 4.1 - таблица истинности для синтеза ОЧС на мультиплексорах.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0 | 0 | 0 | 0 | 0 | 1 |  | 0 | *p* | 0 |  |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |  | 1 |  | 1 |  |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | *0* | 0 | *p* | 0 |  |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |  | 1 |  | 0 |  |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |  | 1 |  | 1 |  |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | 1 | *1* | 0 |  | 0 |  |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |  | 1 |  | 0 |  |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |  | 0 | *p* | 0 |  |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

Функциональная электрическая схема ОЧС на мультиплексорах представлена в приложении М.

# **5 ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ**

Формула расчёта временных затрат на умножение:

= , где

– время преобразования множителя;

– время формирования дополнительного кода множимого;

– время умножения на ОЧУС;

– время формирования единицы переноса в ОЧС;

– время сдвига в регистрах;

n – количество разрядов множителя без знакового разряда;

m – количество разрядов множимого без знакового разряда;

+1 – знаковый разряд.

Таблица 5.1 содержат в себе значения коэффициента минимизации *k* для ОЧС и ОЧУС.

Таблица 5.1 коэффициент минимизации *k* для ОЧС и ОЧУС.

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | *k* для ОЧС | Функция | *k* для ОЧУС |
| *П* | *k* = 1,08 | *П* | *k* = 1,62 |
|  | *k* = 1,09 |  | *k* = 1,2 |
|  | *k* = 1,50 |  | *k* = 1,25 |

Минимизация функций позволила в несколько раз удешевить схему умножителя-сумматора и уменьшить затраты времени на выполнение за счет уменьшения количества элементов.

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсовой работы была разработана структурная схема сумматора-умножителя второго типа, а также функциональные схемы основных узлов данного устройства. Для уменьшения стоимости логических схем были выполнены минимизации переключательных функций 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.

Функциональные схемы были построены в различных логических базисах. Это позволило закрепить теоретические знания основных законов булевой алгебры, например, правило де Моргана.

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

# **ПРИЛОЖЕНИЕ А**

**(**обязательное**)**

Сумматор-умножитель второго типа. Схема электрическая структурная

# **ПРИЛОЖЕНИЕ Б**

**(**обязательное**)**

Алгоритм Рота

# **ПРИЛОЖЕНИЕ В**

**(**обязательное**)**

Алгоритм Рота

# **ПРИЛОЖЕНИЕ Г**

**(**обязательное**)**

Алгоритм Рота

# **ПРИЛОЖЕНИЕ Д**

**(**обязательное**)**

Алгоритм Рота

# **ПРИЛОЖЕНИЕ E**

**(**обязательное**)**

Алгоритм Рота

# **ПРИЛОЖЕНИЕ Ж**

**(**обязательное**)**

Алгоритм Рота

# **ПРИЛОЖЕНИЕ З**

**(**обязательное**)**

Алгоритм Рота

# **ПРИЛОЖЕНИЕ И**

**(**обязательное**)**

Одноразрядный четверичный умножитель-сумматор. Схема электрическая функциональная

# **ПРИЛОЖЕНИЕ К**

**(**обязательное**)**

Одноразрядный четверичный сумматор. Схема электрическая функциональная

# **ПРИЛОЖЕНИЕ Л**

**(**обязательное**)**

Преобразователь множителя. Схема электрическая функциональная

# **ПРИЛОЖЕНИЕ М**

**(**обязательное**)**

Однозарядный четверичный сумматор. Схема электрическая функциональная на основе мультиплексоров

# **ПРИЛОЖЕНИЕ Н**

**(**обязательное**)**

Ведомость документов