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

вычислительной техники

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_ Д. В. Куприянова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ПРОЕКТИРОВАНИЕ И ЛОГИЧЕСКИЙ СИНТЕЗ СУММАТОРА-УМНОЖИТЕЛЯ ДВОИЧНО-ЧЕТВЕРИЧНЫХ ЧИСЕЛ

БГУИР КР 1-40 02 01 324 ПЗ

Студент Полховский А. Ф.

Руководитель Куприянова Д. В.

Минск 2020

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

вычислительной техники

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_ Б. В. Никульшин

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2020 г.

ЗАДАНИЕ

по курсовой работе студента  
 Полховского Артема Федоровича

1. Тема работы: Проектирование и логический синтез сумматора-умножителя двоично-десятичных чисел
2. Срок сдачи студентом законченной работы: 20 мая 2020 г.
3. Исходные данные к работе:
   1. Исходные сомножители: Мн = 29,71; Мт = 56,49.
   2. Алгоритм умножения: Б.
   3. Метод умножения: умножение закодированного двоично-четверичного множимого на два разряда двоичного множителя одновременно в прямых кодах.
   4. Коды четверичных цифр множимого для перехода к двоично-четверичной системе кодирования; 04 – 10, 14 – 11, 24 – 01, 34 – 00.
   5. Тип синтезируемого сумматора - умножителя: 1.
   6. Логический базис для реализации ОЧС: А5 – ИЛИ НЕ; метод минимизации – алгоритм Рота.
   7. Логический базис для реализации ОЧУ: А1 – И ИЛИ НЕ; метод минимизации – карты Карно-Вейча.
4. Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Введение. 1 Разработка алгоритма умножения. 2 Разработка структурной схемы сумматора-умножителя. 3 Разработка функциональных схем основным узлов сумматора-умножителя. 4 Синтез комбинационных схем на основе мультиплексоров. 5 Логический синтез преобразователя множителя. 6 Оценка результатов разработки. Заключение. Список использованных источников.

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

КАЛЕНДАРНЫЙ ПЛАН

Разделы 1, 2 к 1 марта 2020 г. – 20 %;

Раздел 3, 4 к 1 апреля 2020 г. – 50 %;

Разделы 5, 6 к 1 мая 2020 г. – 80 %;

Оформление пояснительной записки и графического материала

к 20 мая 2020 г. – 15 %

Защита курсового проекта с 25 мая 2020 г. по 10 июня 2020 г.

Дата выдачи задания: 14 февраля 2019 г.

Руководитель Д. В. Куприянова

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Ф. Полховский

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc41406956)

[1. РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ 6](#_Toc41406957)

[1.1 Перевод сомножителей из десятичной системы счисления в 6](#_Toc41406958)

[четверичную. 6](#_Toc41406959)

[1.2 Запись сомножителей в форме с плавающей запятой в прямом коде: 6](#_Toc41406960)

[1.3 Умножение двух чисел с плавающей запятой на два разряда множителя одновременно в прямых кодах. 6](#_Toc41406961)

[2. РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА-УМНОЖИТЕЛЯ 9](#_Toc41406962)

[3. РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА-УМНОЖИТЕЛЯ 12](#_Toc41406963)

[3.1 Логический синтез одноразрядного четверичного умножителя 12](#_Toc41406964)

[3.2 Логический синтез одноразрядного четверичного сумматора 16](#_Toc41406965)

[4. ЛОГИЧЕСКИЙ СИНТЕЗ ОДНОРАЗРЯДНОГО ЧЕТВЕРИЧНОГО СУММАТОРА НА ОСНОВЕ МУЛЬТИПЛЕКСОРА 27](#_Toc41406966)

[5. ЛОГИЧЕСКИЙ СИНТЕЗ ПРЕОБРАЗОВАТЕЛЯ МНОЖИТЕЛЯ 29](#_Toc41406967)

[6. ВРЕМЕННЫЕ ЗАТРАТЫ НА УМНОЖЕНИЕ 31](#_Toc41406968)

[ЗАКЛЮЧЕНИЕ 32](#_Toc41406969)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc41406970)

[ПРИЛОЖЕНИЕ А 34](#_Toc41406971)

[ПРИЛОЖЕНИЕ Б 35](#_Toc41406972)

[ПРИЛОЖЕНИЕ В 36](#_Toc41406973)

[ПРИЛОЖЕНИЕ Г 37](#_Toc41406974)

[ПРИЛОЖЕНИЕ Д 38](#_Toc41406975)

[ПРИЛОЖЕНИЕ Е 39](#_Toc41406976)

[ПРИЛОЖЕНИЕ Ж 40](#_Toc41406977)

# ВВЕДЕНИЕ

Целью данной курсовой работы является проектирование цифрового устройства двоично-четверичного сумматора-умножителя (CУ). Сумматор является одним из центральных узлов арифметико-логического устройства (АЛУ) вычислительной машины.

Для того чтобы спроектировать данное устройство, необходимо пройти несколько последовательных этапов разработки: разработка алгоритма умножения чисел, по которому работает СУ; разработка структурной схемы; разработка функциональной схемы основных узлов структурной схемы СУ (одноразрядного четверичного умножителя (ОЧУ), одноразрядного четверичного сумматора (ОЧС), а также переключательные функции ОЧС на мультиплексорах); оценка результатов проделанной работы (эффективность минимизации и время выполнения операций). Минимизация перечисленных ранее устройств осуществляется с помощью карт Карно-Вейча и алгоритма извлечения (Рота).

# 1. РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ

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

## **четверичную.**

Множимое

29 | 4 0.71

28 **\_**7| 4 \* 4

1 4 1 2.84

3 \* 4

3.36

\* 4

1.44

Мн4 = 131,231.

В соответствии с заданной кодировкой множимого:

Мн2/4 = 110011,010011.

Множитель

**\_**56 | 4 0.49

56 **\_**14 | 4 \* 4

0 12 ­­ 3 1.96

2 \* 4

3.84

\* 4

3.36

Мт4 = 320,133.

В соответствии с обычной весомозначной кодировкой множителя:

Мт2/4 = 111000,011111.

## **1.2 Запись сомножителей в форме с плавающей запятой в прямом коде:**

Мн = 0,110011010011 РМн = 0.1000 + 0310  – закодировано по заданию,

Мт = 0,111000011111 РМт = 0.0011 + 0310  – закодировано традиционно.

## **1.3 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 = 0.1000 034

РМт = 0.0011 034

РМн∙Мт = 0.1101 124

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 прямых кодах диада 11(34) заменяется на триаду 1. Преобразованный множитель имеет вид: Мтп4 = 12020 или Мтп2= 1100100. Перемножение мантисс по алгоритму “Б” приведено в таблице 1.1.

Таблица 1.1 - Перемножение мантисс

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Четверичная с/c** | | | **Двоично-четверичная с/с** | | | **Комментарии** |
| **1** | | | **2** | | | **3** |
| 0. | 000000 | 000000 | 0. | 10 10 10 10 10 10 | 10 10 10 10 10 10 | **∑0 ч** |
| 3. | 333333 | 202103 | 1. | 00 00 00 00 00 00 | 01 10 01 11 10 00 | П1Ч = [-Мн]д |
| 3. | 333333 | 202103 | 1. | 00 00 00 00 00 00 | 01 10 01 11 10 00 | **∑1 ч = ∑2 ч** |
| 0. | 000032 | 312200 | 0. | 10 10 10 10 00 01 | 00 11 01 01 10 10 | П3Ч= [2Мн] ∙ 22 |
| 0. | 000001 | 120303 | 0. | 10 10 10 10 10 11 | 01 10 00 10 00 10 | **∑3 ч = ∑4 ч** |
| 0. | 003231 | 220000 | 0. | 10 10 00 01 00 11 | 01 01 10 10 10 10 | П5Ч= [2Мн] ∙ 24 |
| 0. | 003233 | 000303 | 0. | 10 10 00 01 00 00 | 10 10 10 00 10 00 | **∑5 ч** |
| 3. | 320210 | 300000 | 1. | 00 01 10 01 11 10 | 00 10 10 10 10 10 | П6Ч= [-Мн]д ∙ 25 |
| 3. | 330103 | 300303 | 1. | 00 00 10 11 10 00 | 00 10 10 00 10 00 | **∑6 ч** |
| 0. | 131231 | 000000 | 0. | 11 00 11 01 00 11 | 10 10 10 10 10 10 | П7Ч= [Мн]∙ 26 |
| 0. | 121330 | 300303 | 0. | 11 01 11 00 00 10 | 00 10 10 00 10 00 | **∑7ч** |

После окончания умножения необходимо оценить погрешность вычислений. Для этого полученное произведение (Мн∙Мт4 = 0, 2210323003030, РМн∙Мт10 = 6) приводится к нулевому порядку, а затем переводится в десятичную систему счисления:

Мн ∙ Мт4 = 121330.300303; РМн ∙ Мт = 0;

Мн ∙ Мт10 = 1660.7624

Результат прямого перемножения операндов дает следующее значение:

Мн10 · Мт10 = 29,71 · 56,49 = 1678,3179

Абсолютная погрешность:

Δ = 1678,3179 – 1660,7624= 17,5555.

Относительная погрешность рассчитывается по формуле:

Относительная погрешность равна:

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

# 2. РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА-УМНОЖИТЕЛЯ

Структура первого типа строится на базе заданных узлов ОЧУ, ОЧС и аккумулятора (накапливающего сумматора). Управление режимами работы схемы осуществляется внешним сигналом *mul/sum*, который определяет вид текущей арифметической операции (умножение или суммирование).

Структурная схема сумматора-умножителя первого типа для алгоритма «Б» приведена в приложении А.

*Если* *устройство работает как сумматор* (на входе *Mul/sum* – “1”), то оба слагаемых последовательно (за два такта) заносятся в регистр множимого, а на управляющий вход формирователя дополнительного кода (ФДК) *F2*

поступает “1”.

Следует учесть, что числа представлены в форме с плавающей запятой, поэтому, прежде чем складывать мантиссы, необходимо выровнять порядки.

В блоке порядков необходимо обеспечить сравнение порядков, используя сумматор порядков, и в зависимости от знака результата сдвигать первое или второе слагаемое.

Реализация сдвига мантиссы числа с меньшим порядком будет производится в регистре множимого.

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).

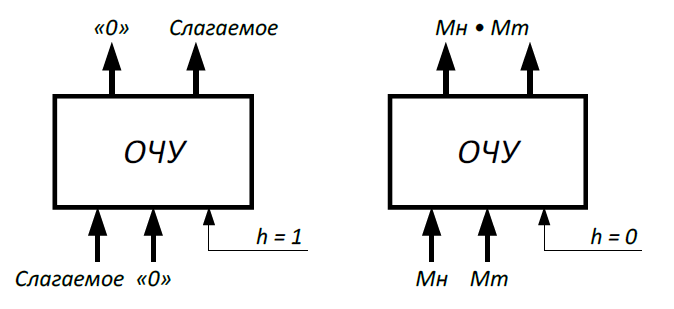


Рисунок 2.1 – Режимы работы ОЧУ

При необходимости выравнивания порядков в регистре-аккумуляторе может выполняться сдвиг мантиссы первого слагаемого. Если на вход 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).

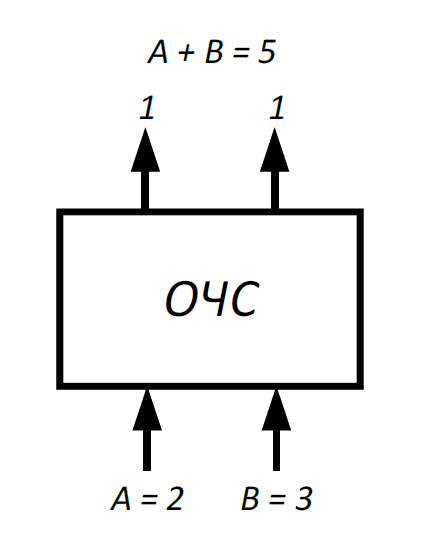


Рисунок 2.2 – Одноразрядный четверичный сумматор

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 аккумулятор (накапливающий сумматор) складывает операнды и хранит результат.

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

Диада множителя поступает на входы преобразователя множителя (ПМ). Задачей ПМ является преобразование диады множителя в соответствии с алгоритмом преобразования. При этом в случае образования единицы переноса в старшую диаду множителя она должна быть учтена при преобразовании следующей старшей диады (выход 1 ПМ), т.е. сохраняться до следующего такта на триггер.

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

Выход 2 ПМ переходит в единичное состояние, если текущая диада содержит отрицание (). В этом случае инициализируется управляющий вход *F1* формирователя дополнительного кода (ФДК) 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 |

На выходах 3 и 4 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.

Для суммирования результата умножения текущей диады Мн∙Мт с переносом из предыдущей диады предназначены ОЧС. 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 в ОЧС) и нулевой частичной суммы (хранящейся в аккумуляторе).

В аккумуляторе происходит сложение *i*-й с (*i*+1)-м частичным произведением, результат сложения сохраняется.

На четырёх входах 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”:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | \* | 2 | = | 12 |
| 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.

Частичные суммы хранятся только в аккумуляторе, т.к. алгоритм умножения «Б» не предполагает возможность сдвига сумм в регистр множителя.

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 – 10; 1 – 11; 2 – 01; 3 – 00.

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

Таблица 3.1.1 – Таблица истинности ОЧУ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Мн** | | **Мт** | | **Упр.** | **Старшие**  **разряды** | | **Младшие**  **разряды** | | **Пример операции в четверичной с/с** |
| ***x1*** | ***x2*** | ***y1*** | ***y2*** | ***h*** | ***P1*** | ***P2*** | ***P3*** | ***P4*** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 3·0=00 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Выход – код «03» |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 3·1=03 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | Выход – код «03» |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 3·2=12 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | Выход – код «03» |
| 0 | 0 | 1 | 1 | 0 | x | x | x | x | 3·3=21 |
| 0 | 0 | 1 | 1 | 1 | x | x | x | x | Выход – код «03» |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 2·0=00 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | Выход – код «02» |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2·1=02 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | Выход – код «02» |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 2·2=10 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | Выход – код «02» |
| 0 | 1 | 1 | 1 | 0 | x | x | x | x | 2·3=12 |
| 0 | 1 | 1 | 1 | 1 | x | x | x | x | Выход – код «02» |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0·0=00 |

*Продолжение таблицы 3.1.1*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | Выход – код «00» |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0·1=00 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | Выход – код «00» |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0·2=00 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | Выход – код «00» |
| 1 | 0 | 1 | 1 | 0 | x | x | x | x | 0·3=00 |
| 1 | 0 | 1 | 1 | 1 | x | x | x | x | Выход – код «00» |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1·0=00 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | Выход – код «01» |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1·1=01 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | Выход – код «01» |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1·2=02 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | Выход – код «01» |
| 1 | 1 | 1 | 1 | 0 | x | x | x | x | 1·3=03 |
| 1 | 1 | 1 | 1 | 1 | x | x | x | x | Выход – код «01» |

Минимизацию переключательный функций проведём с помощью карт Вейча.

Для функции *P1* заполненная карта приведена на рисунке 3.1.1, где символом «х» отмечены наборы, на которых функция может принимать произвольное значение.

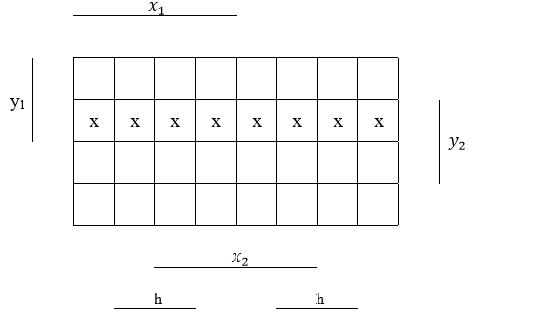


Рисунок 3.1.1 Минимизация функции *P1* при помощи карты Вейча

*P1*МКНФ *= 0*

Для функции *P2* заполненная карта приведена на рисунке 3.1.2

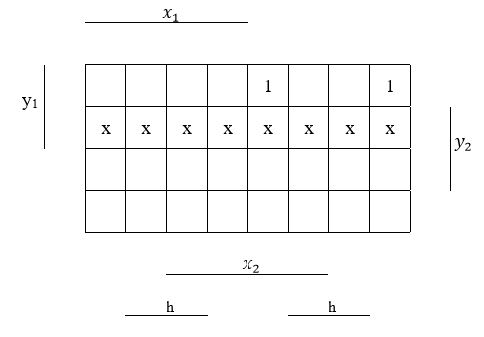
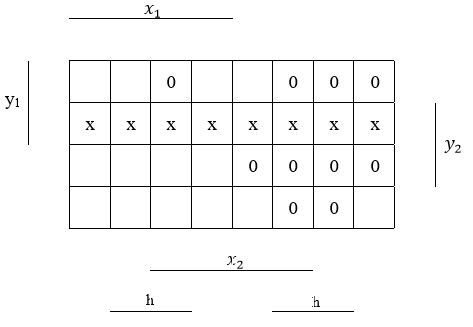


Рисунок 3.1.2 Минимизация функции P2 при помощи карты Вейча

*P2*МДНФ*=*

Для функции *P3* заполненная карта приведена на рисунке 3.1.3



0

Рисунок 3.1.3 Минимизация функции *P3* при помощи карты Вейча

*P3*МКНФ *=*

Для функции P4 заполненная карта приведена на рисунке 3.1.4

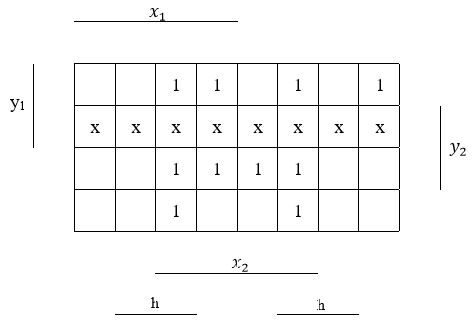


Рисунок 3.1.4 Минимизация функции *P4* при помощи карты Вейча

*P4*МДНФ *= x2y2 .*

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

Функциональная схема для ОЧУ приведена в приложении В. Функции для реализации в заданном базисе (А1) будут иметь следующий вид:

*P1*МКНФ *=* 0

*P2*МДНФ*=*

*P3*МКНФ *=*

*P4*МДНФ *= x2y2*

## **3.2 Логический синтез одноразрядного четверичного сумматора**

Одноразрядный четверичный сумматор – это комбинационное устройство, имеющее 5 входов (2 разряда одного слагаемого, 2 разряда второго слагаемого и вход переноса) и 3 выхода (старший и младший разряды и перенос). Принцип работы ОЧС представлен с помощью таблицы истинности (таблица 3.2.1).

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

В таблице 3.2.1 ОЧС синтезируется для схемы первого типа. В таблице истинности необходимо выделить 16 безразличных наборов, т. к. со старших выходов ОЧУ не могут прийти коды «2» и «3».

Таблица 3.2.1 – Таблица истинности ОЧС

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

*Продолжение таблицы 3.2.1*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0+1+1=02 |
| 1 | 1 | 0 | 0 | 0 | x | x | x | 1+3+0=10 |
| 1 | 1 | 0 | 0 | 1 | x | x | x | 1+3+1=11 |
| 1 | 1 | 0 | 1 | 0 | x | x | x | 1+2+0=03 |
| 1 | 1 | 0 | 1 | 1 | x | x | x | 1+2+1=10 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1+0+0=01 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1+0+1=02 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1+1+0=02 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1+1+1=03 |

Минимизацию переключательных функций проведём с помощью карт Карно.

Для функции *S1* заполненная карта Карно приведена на рисунке 3.2.1, где символом «х» отмечены наборы, на которых функция может принимать произвольное значение.

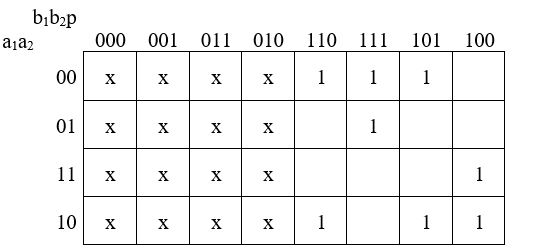


Рисунок 3.2.1 – Минимизация функции *S1*при помощи карты Карно

Результат минимизации примет вид:

*S1*МДНФ *=*

Для минимизации функции *S1* также дополнительно воспользуемся алгоритмом Рота.

Определим множество единичных кубов:

Далее определим множество безразличных кубов:

В целях упрощения минимизации произведём минимизацию безразличных кубов. Минимизацию безразличных кубов приведём с помощью карты Карно. Для безразличных кубов заполненная карта приведена на рисунке 3.2.2.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | y1y2h | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |  |
| x1x2 |
|  | 00 | x | x | x | x |  |  |  |  |  |
|  | 01 | x | x | x | x |  |  |  |  |  |
|  | 11 | x | x | x | x |  |  |  |  |  |
|  | 10 | x | x | x | x |  |  |  |  |  |

Рисунок 3.2.2 – Минимизация безразличных кубов функции *S*2 при помощи карты Карно.

Множество безразличных кубов после минимизации*:*

Сформируем множество:

Первым этапом алгоритма Рота является нахождение множества простых импликант. Для реализации этого этапа будем использовать операцию умножения (\*) над множествами C0, C1 и т.д., пока в результате операции будут образовываться новые кубы большей размерности.

Первый шаг умножения *(С0 \* С0)* приведен в таблице 3.2.2.

Таблица 3.2.2 – Первый шаг умножения (С0 \* С0)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *C0*\**C0* | 00101 | 00110 | 00111 | 01111 | 10100 | 10101 | 10110 | 11100 | xx0xx |
| 00101 | - |  |  |  |  |  |  |  |  |
| 00110 |  | - |  |  |  |  |  |  |  |
| 00111 | 001y1 | 0011y | - |  |  |  |  |  |  |
| 01111 |  |  | 0y111 | - |  |  |  |  |  |
| 10100 |  |  |  |  | - |  |  |  |  |
| 10101 | y0101 |  |  |  | 1010y | - |  |  |  |
| 10110 |  | y0110 |  |  | 101y0 |  | - |  |  |
| 11100 |  |  |  |  | 1y100 |  |  | - |  |
| xx0xx | 00y01 | 00y10 | 00y11 | 01y11 | 10y00 | 10y01 | 10y10 | 11y00 | - |
| *A1* | 001x1 x0101 00x01 | 0011x x0110 00x10 | 0x111 00x11 | 01x11 | 1010x 101x0 1x100 10x00 | 10x01 | 10x10 | 11x00 | Ø |

Из таблицы 3.2.2 – Поиск простых импликант (*С0* \* *С0*)следует:

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

Сформируем множество :

После этой операции сформируется новое множество кубов*.*

В таблице 3.2.3 приведён следующий шаг поиска простых импликант с помощью операции C1\* C1. Таблица 3.2.3 приведена в приложении Б.

Из таблицы 3.2.2 – Поиск простых импликант () следует:

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

Сформируем множество :

После этой операции сформируется новое множество кубов .

В таблице 3.2.4 приведён следующий шаг поиска простых импликант с помощью операции C2\* C2.

Из таблицы 3.2.4 – Поиск простых импликант () следует:

Таблица 3.2.4 – Третий шаг умножения (C2\* C2)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C2\*C2 | 00xx1 | x0x01 | 00x1x | x0x10 | 0xx11 | 10x0x | 10xx0 | 1xx00 |
| 00xx1 | - |  |  |  |  |  |  |  |
| x0x01 |  | - |  |  |  |  |  |  |
| 00x1x |  |  | - |  |  |  |  |  |
| x0x10 |  |  |  | - |  |  |  |  |
| 0xx11 |  |  |  |  | - |  |  |  |
| 10x0x |  |  |  |  |  | - |  |  |
| 10xx0 |  |  |  |  |  |  | - |  |
| 1xx00 |  |  |  |  |  |  |  | - |
| xx0xx |  |  |  |  |  |  |  |  |
| A3 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

На этом этап поиска простых импликант заканчивается т.к. .

Конечное множество простых импликант:

Следующий этап – поиск *L*-экстремалей на множестве простых импликант приведен в таблице 3.2.5. Для этого используется операция # (решетчатое вычитание).

Таблица 3.2.5 - Поиск *L*-экстремалей

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Z#(Z\z) | 00xx1 | x0x01 | 00x1x | x0x10 | 0xx11 | 10x0x | 10xx0 | 1xx00 | xx0xx |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 00xx1 | - | 10x01 | 00x10 | x0x10 | 01x11 | 10x0x | 10xx0 | 1xx00 | 1x0xx, x10xx, xx0x0 |
| x0x01 | 00x11 | - | 00x10 | x0x10 | 01x11 | 10x00 | 10xx0 | 1xx00 | 110xx, 1x01x, 1x0x0, x10xx, xx0x0 |
| 00x1x | Ø | 10x01 | - | 10x10 | 01x11 | 10x00 | 10xx0 | 1xx00 | 110xx, 1x01x, 1x0x0, x10xx, 1x0x0, x10x0,  xx000 |
| x0x10 | Ø | 10x01 | Ø | - | 01x11 | 10x00 | 10x00 | 1xx00 | 110xx, 1101x, 1x011, 110x0, 1x000, x10xx, 110x0, 1x000, x10x0, xx000 |

*Продолжение таблицы 3.2.5*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0xx11 | Ø | 10x01 | Ø | 10x10 | - | 10x00 | 10x00 | 1xx00 | 110xx, 1101x, 1x011, 110x0, 1x000, 110xx, x100x, x10x0, 110x0, 1x000, x10x0, xx000 |
| 10x0x | Ø | Ø | Ø | 10x10 | 01x11 | - | Ø | 11x00 | 110xx, 1101x, 1x011, 110x0, 11000, 110xx, x100x, x10x0, 110x0, 11000, x10x0, 0x000, x1000 |
| 10xx0 | Ø | Ø | Ø | Ø | 01x11 | Ø | - | 11x00 | 110xx, 1101x, 1x011, 110x0, 11000, 110xx, x100x, x10x0, 110x0, 11000, x10x0, 0x000, x1000 |
| 1xx00 | Ø | Ø | Ø | Ø | 01x11 | Ø | Ø | - | 1101x, 110x1,  1101x, 1x011, 11010, 1101x, 110x1, 0100x, x1001, 010x0,  x1010, 11010, 010x0, x1010, 0x000, 01000 |
| xx0xx | Ø | Ø | Ø | Ø | 01111 | Ø | Ø | 11100 | - |
| Остаток | Ø | Ø | Ø | Ø | 01111 | Ø | Ø | 11100 | 1101x, 110x1, 1101x, 1x011, 11010, 1101x, 110x1, 0100x, x1001, 010x0, x1010, 11010, 010x0, x1010, 0x000, 01000 |

В таблице 3.2.5 из каждой простой импликанты поочередно вычитаются все остальные простые импликанты Z#(Z\z), результат операции (последняя строка таблицы) указывает на то, что *L*-экстремалями стали следующие простые импликанты:

Необходимо проверить, нет ли среди полученных *L*-экстремалей таких, которые стали *L*-экстремалями за счет безразличных кубов. Для этого в таблице 3.2.6 из кубов множества *L* вычитаются остатки простых импликант, полученные в таблице 3.2.5 (результат выполнения операции Z#(Z\z)).

Таблица 3.2.6 – Проверка *L*-экстремалей

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Z#(Z\z) n L | 00101 | 00110 | 00111 | 01111 | 10100 | 10101 | 10110 | 11100 |
| 01111 | Ø | Ø | Ø | 01111 | Ø | Ø | Ø | Ø |
| 11100 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | 11100 |
| 1101x | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 110x1 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 1101x | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 1x011 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 11010 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 1101x | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 110x1 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 0100x | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| x1001 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 010x0 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| x1010 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 11010 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 010x0 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| x1010 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 0x000 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 01000 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

Множество L-экстремалей E:

Далее необходимо проанализировать, какие из исходных единичных кубов (множество *L*) не покрыты найденными *L*-экстремалями. Этот анализ осуществляется с помощью таблицы 3.2.7.

Таблица 3.2.7 – Поиск непокрытых исходных наборов

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L#E | 00101 | 00110 | 00111 | 01111 | 10100 | 10101 | 10110 | 11100 |
| 0xx11 | 00101 | 00110 | Ø | Ø | 10100 | 10101 | 10110 | 11100 |
| 1xx00 | 00101 | 00110 | Ø | Ø | Ø | 10101 | 10110 | Ø |
| Остаток | 00101 | 00110 | Ø | Ø | Ø | 10101 | 10110 | Ø |

Из таблицы 3.2.7 видно, что *L*-экстремалями не покрыто четыре куба из множества L - {00101, 00110, 10101, 10110}. Чтобы их покрыть, воспользуемся множеством простых импликант, не являющихся *L*-экстремалями (таблица 3.2.8).

Таблица 3.2.8 – Покрытие оставшихся кубов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Ẑ∩L* | 00101 | 00110 | 10101 | 10110 |
| 00xx1 | 00101 | Ø | Ø | Ø |
| x0x01 | 00101 | Ø | 10101 | Ø |
| 00x1x | Ø | 00110 | Ø | Ø |
| x0x10 | Ø | 00110 | Ø | 10110 |
| 10x0x | Ø | Ø | 10101 | Ø |
| 10xx0 | Ø | Ø | Ø | 10110 |
| xx0xx | Ø | Ø | Ø | Ø |

Из таблицы 3.2.8 видно, что непокрытый единичный куб может быть покрыт одним минимальным способом.

Следовательно, существует одна тупиковая (минимальная) форма:

*S1*МДНФ *=*

Для функции *S2* проведём минимизацию картой Карно. Заполненная для функции *S2* карта Карно приведена на рисунке 3.2.3.

Результат минимизации примет вид:

*S2*МКНФ *=*

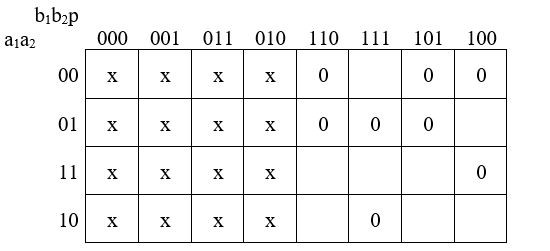


Рисунок 3.2.3 – Минимизация функции *S2*при помощи карты Карно

Для функции П проведём минимизацию картой Карно. Заполненная карта Карно для функции П приведена на рисунке 3.2.4.

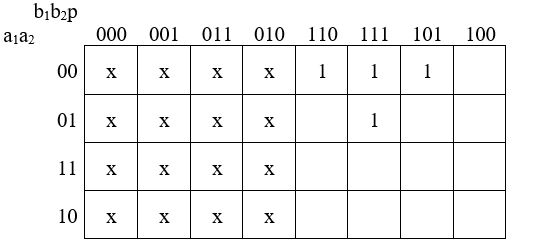


Рисунок 3.2.4 – Минимизация функции Ппри помощи карты Карно

Результат минимизации примет вид:

П *=*

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

Функциональная схема для ОЧС приведена в приложении Г. Функции для реализации в заданном базисе (А5) будут иметь следующий вид:

*S1*МДНФ *= =*

*S2*МКНФ *=*

П *= =*  =

# 4. ЛОГИЧЕСКИЙ СИНТЕЗ ОДНОРАЗРЯДНОГО ЧЕТВЕРИЧНОГО СУММАТОРА НА ОСНОВЕ МУЛЬТИПЛЕКСОРА

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 адресными входами, это позволит упростить одну нашу большую функцию от пяти аргументов до восьми функций от двух переменных. Функции от двух переменных достаточно просты для того, чтобы самостоятельно заметить их минимальную форму.

Синтез дополнительных логических схем для ПФ ОЧC приведен в таблице 4.1.

Таблица 4.1 – Таблица истинности для ОЧС на мультиплексорах

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1a2b1 | b2 | p | П | П | S1 | S1 | S2 | S2 |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| 000 | 0 | 0 | \* | \* | \* | \* | \* | \* |
| 000 | 0 | 1 | \* | \* | \* |
| 000 | 1 | 0 | \* | \* | \* |
| 000 | 1 | 1 | \* | \* | \* |
| 001 | 0 | 0 | 0 |  | 0 |  | 0 |  |
| 001 | 0 | 1 | 1 | 1 | 0 |
| 001 | 1 | 0 | 1 | 1 | 0 |
| 001 | 1 | 1 | 1 | 1 | 1 |
| 010 | 0 | 0 | \* | \* | \* | \* | \* | \* |
| 010 | 0 | 1 | \* | \* | \* |
| 010 | 1 | 0 | \* | \* | \* |
| 010 | 1 | 1 | \* | \* | \* |
| 011 | 0 | 0 | 0 |  | 0 |  | 1 |  |
| 011 | 0 | 1 | 0 | 0 | 0 |
| 011 | 1 | 0 | 0 | 0 | 0 |
| 011 | 1 | 1 | 1 | 1 | 0 |
| 100 | 0 | 0 | \* | \* | \* | \* | \* | \* |
| 100 | 0 | 1 | \* | \* | \* |
| 100 | 1 | 0 | \* | \* | \* |

*Продолжение таблицы 4.1*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| 100 | 1 | 1 | \* |  | \* |  | \* |  |
| 101 | 0 | 0 | 0 | “0” | 1 |  | 0 |  |
| 101 | 0 | 1 | 0 | 1 | 1 |
| 101 | 1 | 0 | 0 | 1 | 1 |
| 101 | 1 | 1 | 0 | 0 | 1 |
| 110 | 0 | 0 | \* | \* | \* | \* | \* | \* |
| 110 | 0 | 1 | \* | \* | \* |
| 110 | 1 | 0 | \* | \* | \* |
| 110 | 1 | 1 | \* | \* | \* |
| 111 | 0 | 0 | 0 | “0” | 1 |  | 1 |  |
| 111 | 0 | 1 | 0 | 0 | 1 |
| 111 | 1 | 0 | 0 | 0 | 1 |
| 111 | 1 | 1 | 0 | 0 | 0 |

Функциональная схема реализации ОЧС на мультиплексорах приведена в приложении Д.

# 5. ЛОГИЧЕСКИЙ СИНТЕЗ ПРЕОБРАЗОВАТЕЛЯ МНОЖИТЕЛЯ

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

Таблица 5.1 - Таблица истинности ПМ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Вх. диада** | | **Мл. бит** | **Зн.** | **Вых. 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 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

Минимизируем выходные функции картами Карно. Заполненные карты для переменных соответственно, в табл. 5.2, табл. 5.3, табл. 5.4.

Таблица 5.2 – Минимизация функции *P*

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

Таблица 5.3 – Минимизация функции

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

Таблица 5.4 – Минимизация функции

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

Функциональная схема ПМ приведена в приложении E.

# 6. ВРЕМЕННЫЕ ЗАТРАТЫ НА УМНОЖЕНИЕ

Формула расчёта временных затрат на умножение:

, где

– время преобразования множителя;

– время формирования дополнительного кода множимого;

– время умножения на ОЧУ;

– время формирования единицы переноса в ОЧС;

– время сдвига частичной суммы;

*n* – количество разрядов множителя;

m – количество разрядов на множимого.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы были разработаны алгоритмы выполнения операций умножения и сложения, структурная схема сумматора-умножителя первого типа, а также функциональные схемы основных узлов данного устройства (ОЧС и ОЧУ). 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 выполнения и минимальные затраты времени. Однако применение данного способа для функций с большим количеством переменных будет затруднительно, также он не является формализованным.

Для минимизации подобных функций удобно использовать алгоритм Рота. Его преимуществом является полная формализация алгоритмов минимизации, что даёт возможность проводить минимизацию в автоматическом режиме.

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.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

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

[3] Лысиков, Б. Г. Арифметические и логические основы цифровых автоматов / Б. Г. Лысиков. – Минск : Выш. шк., 1980. – 342 с.

[4] Лысиков, Б. Г. Цифровая вычислительная техника / Б. Г. Лысиков. –

Минск : Выш. шк., 2003. – 242 с.

[5] Савельев, А. Я. Прикладная теория цифровых автоматов / А. Я. Са-

. – М. : Высш. шк., 1987. – 272 с.

# ПРИЛОЖЕНИЕ А

*(обязательное)*

Сумматор умножитель первого типа. Схема электрическая структурная

# ПРИЛОЖЕНИЕ Б

*(обязательное)*

Таблица 3.2.3 - Поиск простых импликант ()

# ПРИЛОЖЕНИЕ В

*(обязательное)*

Одноразрядный четверичный умножитель. Схема электрическая функциональная

# ПРИЛОЖЕНИЕ Г

*(обязательное)*

Одноразрядный четверичный сумматор. Схема электрическая функциональная

# ПРИЛОЖЕНИЕ Д

*(обязательное)*

Одноразрядный четверичный сумматор.

Реализация на мультиплексорах.

Схема электрическая функциональная

# ПРИЛОЖЕНИЕ Е

*(обязательное)*

Преобразователь множителя. Схема электрическая функциональная

# ПРИЛОЖЕНИЕ Ж

*(обязательное)*

Ведомость документов