Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра вычислительных машин, систем и сетей

Дисциплина: Арифметические и логические основы

цифровых устройств

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_\_ Ю. А. Луцик

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ПРОЕКТИРОВАНИЕ И ЛОГИЧЕСКИЙ СИНТЕЗ СУММАТОРА-

УМНОЖИТЕЛЯ ДВОИЧНО-ЧЕТВЕРИЧНЫХ ЧИСЕЛ

БГУИР КР 1-40 02 01 204 ПЗ

Студент А. С. Бригадир

Руководитель Ю. А. Луцик

МИНСК 2023

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

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_г.

ЗАДАНИЕ

по курсовой работе студента

Бригадир Анны Сергеевны

1. Тема работы: «Проектирование и логический синтез сумматора- умножителя двоично-десятичных чисел»
2. Срок сдачи студентом законченной работы: до 20 мая 2023г.
3. Исходные данные к работе:
   1. Исходные сомножители: Мн = 57,49; Мт = 19,54;
   2. Алгоритм умножения: Г;
   3. Метод умножения: умножение закодированного двоично-четверичного множимого на два разряда двоичного множителя одновременно в дополнительных кодах;
   4. Коды четверичных цифр множимого для перехода к двоично- четверичной системе кодирования: 04 – 00, 14 – 10, 24 – 11, 34 – 01;
   5. Тип синтезируемого умножителя: 2;
   6. Логический базис для реализации ОЧС: И, НЕ; метод минимизации – карты Карно-Вейча.
   7. Логический базис для реализации ОЧУС: 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. Умножитель-сумматор 2 типа. Схема электрическая структурная.
   2. Однозарядный четверичный сумматор. Схема электричская

функциональная.

* 1. Однозарядный четверичный умножитель. Схема электрическая функциональная. Однозарядный четверичный умножитель. Схема электрическая функциональная.
  2. Однозарядный четверичный сумматор. Реализация на мультиплексорах. Схема электрическая функциональная.
  3. Преобразователь множителя. Схема электрическая функциональная.

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов курсовой работы | Объём этапа, % | Срок выполнения этапа | Примечания |
| Разработка алгоритма умножения | 10 | 10.02 - 24.02 |  |
| Разработка структурной схемы сумматора-умножителя | 10 | 25.02 - 24.03 | С выполнением чертежа |
| Разработка функциональных схем основных узлов сумматора-умножителя | 50 | 25.03 - 28.04 | С выполнением чертежей |
| Синтез комбинационных схем устройств на основе мультиплексоров | 10 | 29.04 – 14.05 | С выполнением чертежа |
| Завершение оформления пояснительной записки | 20 | 15.05 – 20.05 |  |

Дата выдачи задания: 10 февраля 2023г.

Руководитель \_\_\_\_\_\_\_\_\_ Ю. А. Луцик

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_\_ А. С. Бригадир

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ……………………………………………………………………….5

1 РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ………………………………6

2 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА-УМНОЖИТЕЛЯ………………………………………………………………….9

3 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА-УМНОЖИТЕЛЯ……………………………………………….13

3.1 Логический синтез одноразрядного четверичного сумматора-умножителя ……………………………………………………………….13

3. 2 Логический синтез одноразрядного четверичного сумматора…....18

3.3. Логический синтез преобразователя множителя ………………….32

4 СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ УСТРОЙСТВ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ ………………………………………………………...34

5 ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ…………………………………..36

6 ЗАКЛЮЧЕНИЕ………………………………………………………………..37

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………………………38

ПРИЛОЖЕНИЕ А……………………………………………………………….39

ПРИЛОЖЕНИЕ Б………………………………………………………………..40

ПРИЛОЖЕНИЕ В………………………………………………………………..41

ПРИЛОЖЕНИЕ Г………………………………………………………………..42

ПРИЛОЖЕНИЕ Д………………………………………………………………..43

ПРИЛОЖЕНИЕ Е………………………………………………………………..44

**ВВЕДЕНИЕ**

Курсовое проектирование является обязательным элементом подготовки специалиста с высшим образованием и одной из форм текущей аттестации студента по учебной дисциплине. Для студентов это первая работа такого рода и объёма. Она содержит результаты теоретических и экспериментальных исследований по дисциплине “Арифметические и логические основы вычислительной техники”, включает совокупность аналитических, расчётных, экспериментальных заданий и предполагает выполнение конструкторских работ и разработку графической документации.

Целью данной курсовой работы является проектирование такого цифрового устройства, как двоично-четверичный сумматор-умножитель (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:

* 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**

1. Перевод сомножителей из десятичной системы счисления в четверичную.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| \_ 57 | 4 |  |  |  |  |  | 0,49 |
| 56 | \_ 14 | 4 |  |  |  | \* | 4 |
| 1 | 12 | 3 |  |  |  |  | 1,96 |
|  | 2 |  |  |  |  | \* | 4 |
|  |  |  |  |  |  |  | 3,84 |
|  |  |  |  |  |  | \* | 4 |
|  |  |  |  |  |  |  | 3,36 |

Мн4 = 321,133.

В соответствии с заданной кодировкой множимого:

Мн2/4 = 011110,100101

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| \_ 19 | 4 |  |  |  |  |  | 0,54 |
| 16 | 4 | 4 |  |  |  | \* | 4 |
| 3 | 4 | 1 |  |  |  |  | 2,16 |
|  | 0 |  |  |  |  | \* | 4 |
|  |  |  |  |  |  |  | 0,64 |
|  |  |  |  |  |  | \* | 4 |
|  |  |  |  |  |  |  | 2,56 |

Мт4 = 103,202.

В соответствии с заданной кодировкой множителя:

Мт2/4 = 010011,100010

2. Запишем сомножители в форме с плавающей запятой в прямом коде:

Мн = 0, 00011110100101 РМн = 0.1000 +104

Мт = 0, 010011100010 РМт = 0.0011 +034

Порядок произведения:

|  |  |  |  |
| --- | --- | --- | --- |
| РМн | = | 0.1000 | 104 |
| РМт | = | 0.0011 | 034 |
| РМн∙Мт | = | 0.1001 | 134 |
|  |  |  |  |

Знак произведения определяется суммой по модулю два знаков

сомножителей:

зн Мн ⊕ зн Мт = 0 ⊕ 0 = 0.

При умножении чисел в прямых кодах диада 10(24) заменяется на триаду , 11(34) заменяется на триаду . Преобразованный множитель имеет вид М = .

Перемножение мантисс по алгоритму «Г» представлено в таблице 1.1

Таблица 1.1 — Перемножение мантисс

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Четверичная С/С** | | **Двоично-четверичная С/C** | | **Комментарии** |
| **1** | | **2** | | **3** |
| 0. | 0000000 000000 | 01. | 00000000000000 000000000000 | ∑0ч |
| 0. | 0032113 300000 | 01. | 00000111101001 010000000000 | П1ч = Мн \* 4-1 |
| 0. | 0032113 300000 | 01. | 00000111101001 010000000000 | ∑1ч |
| 0. | 0003211 330000 | 01. | 00000001111010 010100000000 | П2ч = Мн \* 4-2 |
| 0. | 0101331 230000 | 01. | 00100010010110 110100000000 | ∑2ч |
| 0. | 0000000 000000 | 01. | 00000000000000 000000000000 | П3ч = 0 |
| 0. | 0101331 230000 | 01. | 00100010010110 110100000000 | ∑3ч |
| 3. | 3333203 100200 | 01. | 01010101001011 110000100000 | П4ч = [-2Мн]д \* 4-4 |
| 0. | 0101200 330200 | 01. | 00100010001000 000100100000 | ∑4ч |
| 0. | 0000003 211330 | 10. | 00000000000001 111010010100 | П5ч = Мн\* 4-5 |
| 0. | 0101210 202130 | 01. | 00100010001001 010011000100 | ∑5ч |
| 3. | 3333332 031002 | 10. | 01010101010100 101111001000 | П6ч =[-2Мн]д \* 4-6 |
| 0. | 0101202 233132 | 01. | 00100010001000 000100100000 | ∑7ч |

После окончания умножения необходимо оценить погрешность вычислений. Для этого полученное произведение (Мн4 ∙ Мт4 =

0,0101202 233132, РМн ∙ Мт = 13) приводится к нулевому порядку, а затем переводится в десятичную систему счисления:

Мн4 ∙ Мт4 = 0101202,233132 РМн ∙ Мт = 0;

Мн10 ∙ Мт10 = 1122,7417.

Результат прямого перемножения операндов дает следующее:

Мн10 ∙ Мт10 = 57,49\*19,54 = 1123,3546.

Абсолютная погрешность:

Δ = 1123,3546 – 1122,7417 = 0,612.

Относительная погрешность:

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

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

*Если устройство работает как сумматор*, то оба слагаемых последовательно (за два такта) заносятся в регистр множимого, а на управляющий вход формирователя дополнительного кода F2 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).

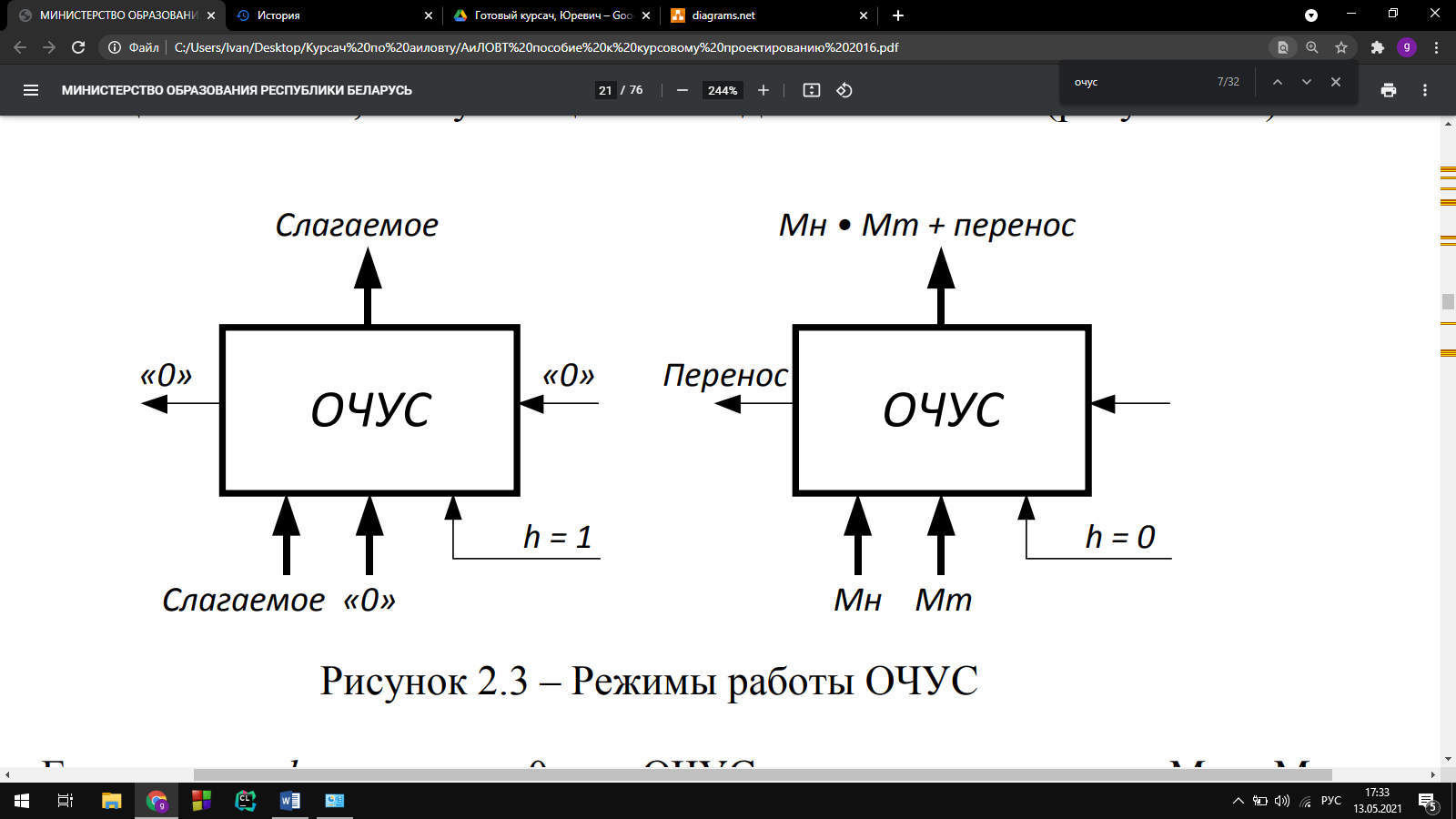


Рисунок 2.1 – Режимы работы ОЧУС

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

В ОЧС первое слагаемое складывается с нулём, записанным в регистре результата, и переписывается без изменений в регистр результата.

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 «А».

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.

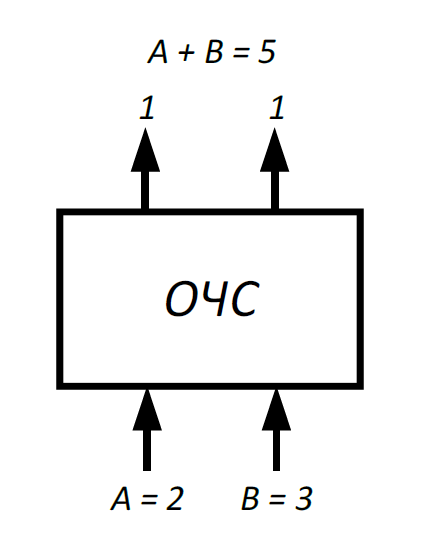
На выходах 3 и 4 ПМ формируются диады преобразованного множителя, которые поступают на входы ОЧУС вместе с диадами множимого. На трёх выходах ОЧУС формируется результат умножения диад Мн·Мт плюс перенос из предыдущего ОЧУС. Максимальной цифрой в диаде преобразованного множителя является двойка, поэтому 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 | |  | |

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.

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.

Разрядность аккумулятор должна быть на единицу больше, чем разрядность исходных слагаемых, чтобы предусмотреть возможность возникновения переноса при суммировании.

*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. Это позволяет использовать регистр множителя для хранения младших разрядов произведения при умножении по алгоритму «А» (регистр множителя служит как бы «продолжением» регистра результата).

Выход 2 ПМ переходит в единичное состояние, если текущая диада содержит отрицание (01). 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 предназначен 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 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 вправо в конце каждого такта умножения по алгоритму «А».

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»:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | · | 2 | = | 1 2 |
| 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 в аккумуляторе и регистре множителя, т.к. алгоритм умножения «А» предполагает возможность синхронного сдвига этих устройств. Количество тактов умножения определяется разрядностью Мт.

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

**3.****1 Логический синтез одноразрядного четверичного сумматора-умножителя**

ОЧУС – это комбинационное устройство, имеющее 6 двоичных входов (2 разряда из регистра Мн, 2 разряда из регистра Мт и управляющий вход *h*) и 3 двоичных выхода.

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

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

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

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, т.к. на входы ОЧУС из разрядов множителя не может поступить код «11».

*Таблица 3.1.1 — Таблица истинности ОЧУС*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Пер.** | **Мн.** | | **Мт.** | | **Упр.** | **Перенос** | **Результат** | | **Пример операции в четверичной с/с** |
| ***p*** | ***x1*** | ***x2*** | ***y1*** | ***y2*** | ***h*** | ***P*** | ***Q1*** | ***Q2*** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 \* 0 + 0 = 00 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Выход – код «00» |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 \* 1 + 0 = 00 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | Выход – код «00» |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 \* 2 + 0 = 00 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | Выход – код «00» |
| 0 | 0 | 0 | 1 | 1 | 0 | x | x | x | 0 \* 3 + 0 = 00 |
| 0 | 0 | 0 | 1 | 1 | 1 | x | x | x | Выход – код «00» |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 \* 0 + 0= 00 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | Выход – код «03» |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 3 \* 1 + 0 = 03 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | Выход – код «03» |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 3 \* 2 + 0 = 12 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | Выход – код «03» |
| 0 | 0 | 1 | 1 | 1 | 0 | x | x | x | 3 \* 3 + 0 = 21 |
| 0 | 0 | 1 | 1 | 1 | 1 | x | x | x | Выход – код «03» |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 \* 0 + 0 = 00 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Выход – код «01» |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 \* 1 + 0 = 01 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | Выход – код «01» |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 \* 2 + 0 = 02 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | Выход – код «01» |
| 0 | 1 | 0 | 1 | 1 | 0 | x | x | x | 1 \* 3 + 0 = 03 |
| 0 | 1 | 0 | 1 | 1 | 1 | x | x | x | Выход – код «01» |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 \* 0 + 0 = 00 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | Выход – код «02» |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 2 \* 1 + 0 = 02 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | Выход – код «02» |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 2 \* 2 + 0 = 10 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | Выход – код «02» |
| 0 | 1 | 1 | 1 | 1 | 0 | x | x | x | 2 \* 3 + 0 = 12 |
| 0 | 1 | 1 | 1 | 1 | 1 | x | x | x | Выход – код «02» |
| 1 | 0 | 0 | 0 | 0 | 0 | х | х | х | 0 \* 0 + 1 = 01 |
| 1 | 0 | 0 | 0 | 0 | 1 | х | х | х | Выход – код «00» |
| 1 | 0 | 0 | 0 | 1 | 0 | х | х | х | 0 \* 1 + 1 = 01 |
| 1 | 0 | 0 | 0 | 1 | 1 | х | х | х | Выход – код «00» |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 \* 2 + 1 = 01 |
| 1 | 0 | 0 | 1 | 0 | 1 | х | х | х | Выход – код «00» |
| 1 | 0 | 0 | 1 | 1 | 0 | x | x | x | 0 \* 3 + 1 = 01 |
| 1 | 0 | 0 | 1 | 1 | 1 | x | x | x | Выход – код «00» |
| 1 | 0 | 1 | 0 | 0 | 0 | х | х | х | 3 \* 0 + 1 = 01 |
| 1 | 0 | 1 | 0 | 0 | 1 | х | х | х | Выход – код «03» |
| 1 | 0 | 1 | 0 | 1 | 0 | х | х | х | 3 \* 1 + 1 = 10 |
| 1 | 0 | 1 | 0 | 1 | 1 | х | х | х | Выход – код «03» |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 3 \* 2 + 1 = 13 |
| 1 | 0 | 1 | 1 | 0 | 1 | х | х | х | Выход – код «03» |
| 1 | 0 | 1 | 1 | 1 | 0 | x | x | x | 3 \* 3 + 1 = 22 |
| 1 | 0 | 1 | 1 | 1 | 1 | x | x | x | Выход – код «03» |
| 1 | 1 | 0 | 0 | 0 | 0 | х | х | х | 1 \* 0 + 1 = 01 |
| 1 | 1 | 0 | 0 | 0 | 1 | х | х | х | Выход – код «01» |
| 1 | 1 | 0 | 0 | 1 | 0 | х | х | х | 1 \* 1 + 1 = 02 |
| 1 | 1 | 0 | 0 | 1 | 1 | х | х | х | Выход – код «01» |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 \* 2 + 1 = 03 |
| 1 | 1 | 0 | 1 | 0 | 1 | х | х | х | Выход – код «01» |
| 1 | 1 | 0 | 1 | 1 | 0 | x | x | x | 1 \* 3 + 1 = 10 |
| 1 | 1 | 0 | 1 | 1 | 1 | x | x | x | Выход – код «01» |
| 1 | 1 | 1 | 0 | 0 | 0 | х | х | х | 2 \* 0 + 1 = 01 |
| 1 | 1 | 1 | 0 | 0 | 1 | х | х | х | Выход – код «02» |
| 1 | 1 | 1 | 0 | 1 | 0 | х | х | х | 2 \* 1 + 1 = 03 |
| 1 | 1 | 1 | 0 | 1 | 1 | х | х | х | Выход – код «02» |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 2 \* 2 + 1 = 11 |
| 1 | 1 | 1 | 1 | 0 | 1 | х | х | х | Выход – код «02» |
| 1 | 1 | 1 | 1 | 1 | 0 | x | x | x | 2 \* 3 + 1 = 13 |
| 1 | 1 | 1 | 1 | 1 | 1 | x | x | x | Выход – код «02» |

**Минимизация функции P:**

Минимизацию функции P проведём с помощью карт Карно и Вейча. 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).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | X1 | | | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| p |  | x | x | x | x | x | x | x | x |  |  |  |
|  | x | x | x | x | x | x | x | x |  |  | H |
|  | x | x | x | x | x | x | x | x |  |  |  |
|  | 0 | x | x | 1 | 1 | x | x | 0 |  | Y1 |  |
|  |  | 0 | x | x | 1 | 1 | x | x | 0 |  |  |  |
|  |  | 0 | x | x | 0 | 0 | x | x | 0 |  |  | H |
|  |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |
|  |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | X2 |  |  |  |  |  |  | -1 |
|  |  |  | Y2 | |  |  | Y2 | |  |  |  |  |

Рисунок 3.1.1 — Минимизация функции Р картой Вейча

Следовательно:

P = x2y1

Запишем результат в базисе ИЛИ, НЕ:

P =

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

*K = =* 8,5

**Минимизация функции Q1:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |  |
| 000 |  |  |  |  | x | x |  |  |  |
|  |
| 001 |  |  |  |  | x | x |  | 1 |  |
|  |
| 011 |  | 1 | 1 | 1 | x | x | 1 |  |  |
|  |
| 010 |  | 1 | 1 | 1 | x | x | 1 | 1 |  |
|  |
|  |  |  |  |  |  |  |  |  |  |
| 110 | x | x | x | x | x | x | x |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 111 | x | x | x | x | x | x | x | 1 |  |
|  |  |  |  |  |  |  |  |  |  |
| 101 | x | x | x | x | x | x | x |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 100 | x | x | x | x | x | x | x | 1 |  |

Рисунок 3.1.2 – Минимизация функции *Q***1**при помощи карты Карно

Следовательно:

Q1=x1y2 + x1h + px1x2 + p12 + x12y1+ 1x2y1

Запишем результат в базисе ИЛИ, НЕ:

Q1 =

Эффективность минимизации:

*K = =* 3,21

##### **Минимизация функции Q2:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |  |
| 000 |  |  |  |  | x | x |  |  |  |
|  |
| 001 |  | 1 | 1 | 1 | x | x | 1 | 1 |  |
|  |
| 011 |  | 1 | 1 | 1 | x | x | 1 |  |  |
|  |
| 010 |  |  |  |  | x | x |  | 1 |  |
|  |
|  |  |  |  |  |  |  |  |  |  |
| 110 | x | x | x | x | x | x | x | 1 |  |
|  |  |  |  |  |  |  |  |  |  |
| 111 | x | x | x | x | x | x | x |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 101 | x | x | x | x | x | x | x | 1 |  |
|  |  |  |  |  |  |  |  |  |  |
| 100 | x | x | x | x | x | x | x |  |  |

Рисунок 3.1.3 – Минимизация функции *Q***2**при помощи карты Карно

Следовательно:

Q2 = x2y2 + x2h + 1x2y1 + x12y1

Запишем результат в базисе ИЛИ, НЕ:

Q2 =

Эффективность минимизации:

*K = =* 5

**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 3.2.1)

Кодировка слагаемых обоих разрядов: 0 – 00, 1 – 10, 2 – 11, 3 – 01;

*Таблица 3.2.1 — Таблица истинности ОЧС*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***а1*** | ***а2*** | ***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 | 0 | 0 | 0 | 0 + 0 + 0 = 00 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 + 0 + 1 = 01 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 + 3 + 0 = 03 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 + 3 + 1 = 10 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 + 1 + 0 = 01 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 + 1 + 1 = 02 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 + 2 + 0 = 02 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 + 2 + 1 = 03 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 3 + 0 + 0 = 03 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 3 + 0 + 1 = 10 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 3 + 3 + 0 = 12 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 3 + 3 + 1 = 13 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 3 + 1 + 0 = 10 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 3 + 1 + 1 = 11 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 3 + 2 + 0 = 11 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 + 2 + 1 = 12 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 + 0 + 0 = 01 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 + 0 + 1 = 02 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 + 3 + 0 = 10 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 + 3 + 1 = 11 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 + 1 + 0 = 02 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 + 1 + 1 = 03 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 + 2 + 0 = 03 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 + 2 + 1 = 10 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 2 + 0 + 0 = 02 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 2 + 0 + 1 = 03 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 2 + 3 + 0 = 11 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 2 + 3 + 1 = 12 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 2 + 1 + 0 = 03 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 2 + 1 + 1 = 10 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 2 + 2 + 0 = 10 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 2 + 2 + 1 =11 |

**Минимизация функции П**

Минимизацию функцииП проведем с помощью карт Карно. Для функции П заполненная карта приведена на рисунке 3.2.0.

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

Рисунок 3.2.0 — Минимизация функции П картой Вейча

Следовательно: П= 22 + 1b2p + 11b2 +121+21p + 12p + a1b2

Запишем результат в базисе И, НЕ:

П =

Эффективность минимизации:

*K = =* 3,48

**Минимизация функции S**1

Минимизацию функции **S1** проведем с помощью метода Рота.

Определим множество единичных кубов:

Множество безразличных кубов пустое.

Сформируем множество С0 = L ⋃ N:

C0 = {00010, 00011, 00111, 01000, 01001, 01010, 01011, 01101, 01110, 01111, 10011, 11001, 11010, 11011, 11110, 11111}

Первым этапом алгоритма Рота является нахождение множества простых импликант.

Для реализации этого этапа будем использовать операцию 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 3.2.2.

Таблица 3.2.2 – Поиск простых импликант (С0\*С0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C0\*C0 | 00001 | 00100 | 00101 | 01010 | 01101 | 01110 | 01111 | 10000 | 10001 | 10011 | 10100 | 11000 | 11010 | 11011 | 11111 |
| 00001 | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 00100 |  | - |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 00101 | 00y01 | 0010y | - |  |  |  |  |  |  |  |  |  |  |  |  |
| 01010 |  |  |  | - |  |  |  |  |  |  |  |  |  |  |  |
| 01101 |  |  | 0y101 |  | - |  |  |  |  |  |  |  |  |  |  |
| 01110 |  |  |  | 01y10 |  | - |  |  |  |  |  |  |  |  |  |
| 01111 |  |  |  |  | 011y1 | 0111y | - |  |  |  |  |  |  |  |  |
| 10000 |  |  |  |  |  |  |  | - |  |  |  |  |  |  |  |
| 10001 | y0001 |  |  |  |  |  |  | 1000y | - |  |  |  |  |  |  |
| 10011 |  |  |  |  |  |  |  |  | 100y1 | - |  |  |  |  |  |
| 10100 |  | y0100 |  |  |  |  |  | 10y00 |  |  | - |  |  |  |  |
| 11000 |  |  |  |  |  |  |  | 1y000 |  |  |  | - |  |  |  |
| 11010 |  |  |  | y1010 |  |  |  |  |  |  |  | 110y0 | - |  |  |
| 11011 |  |  |  |  |  |  |  |  |  | 1y011 |  |  | 1101y | - |  |
| 11111 |  |  |  |  |  |  | y1111 |  |  |  |  |  |  | 11y11 | - |
| A1 | 00x01  x0001 | 0010x  x0100 | 0x101 | 01x10  x1010 | 011x1 | 0111x | x1111 | 1000x  10x00  1x000 | 100x1 | 1x011 | Ø | 110x0 | 1101x | 11x11 | Ø |

В результате этой операции сформируется новое множество кубов:

С1 = {00x01, x0001, 0010x, x0100, 0x101, 01x10, x1010, 011x1, 0111x, x1111, 1000x, 10x00, 1x000, 100x1, 1x011, 110x0, 1101x, 11x11}

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

В таблице 3.2.3 приведён следующий шаг поиска простых импликант с помощью операции С1\*С1.

Таблица 3.2.3 – Поиск простых импликант С1\* С1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C1\*C1 | 00x01 | x0001 | 0010x | x0100 | 0x101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 1000x | 10x00 | 1x000 | 100x1 | 1x011 | 110x0 | 1101x | 11x11 |
| 00x01 | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| x0001 | 00001 | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0010x | 00101 | 00y01 | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| x0100 | 0010y | x0y0y | 00100 | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x101 | 00101 | 00y01 | 00101 | 0010y | - |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 01x10 | 0yxyy | 0y0yy | 0y1y0 | 0y1y0 | 011yy | - |  |  |  |  |  |  |  |  |  |  |  |  |
| x1010 | 0y0yy | xy0yy | 0yyy0 | xyyy0 | 01yyy | 01010 | - |  |  |  |  |  |  |  |  |  |  |  |
| 011x1 | 0y101 | 0yy01 | 0y101 | 0y10y | 01101 | 0111y | 01y1y | - |  |  |  |  |  |  |  |  |  |  |
| 0111x | 0y1y1 | 0yyy1 | 0y1yx | 0y1y0 | 011y1 | 01110 | 01y10 | 01111 | - |  |  |  |  |  |  |  |  |  |
| x1111 | 0y1y1 | xyyy1 | 0y1y1 | xy1yy | 011y1 | 0111y | x1y1y | 01111 | 01111 | - |  |  |  |  |  |  |  |  |
| 1000x | y0001 | 10001 | y0y0x | 10y00 | y0y01 | yy0y0 | 1y0y0 | yyy01 | yyyyx | 1yyy1 | - |  |  |  |  |  |  |  |
| 10x00 | y0x0y | 1000y | y0100 | 10100 | y010y | yyxy0 | 1y0y0 | yy10y | yy1y0 | 1y1yy | 10000 | - |  |  |  |  |  |  |
| 1x000 | y000y | 1000y | y0y00 | 10y00 | yxy0y | y10y0 | 110y0 | y1y0y | y1yy0 | 11yyy | 10000 | 10000 | - |  |  |  |  |  |
| 100x1 | y0001 | 10001 | y0y01 | 10y0y | y0y01 | yy01y | 1y01y | yyyx1 | yyy11 | 1yy11 | 10001 | 1000y | 1000y | - |  |  |  |  |
| 1x011 | y00y1 | 100y1 | y0yy1 | 10yyy | yxyy1 | y101y | 1101y | y1y11 | y1y11 | 11y11 | 100y1 | 100yy | 1x0yy | 10011 | - |  |  |  |
| 110x0 | yy00y | 1y00y | yyy00 | 1yy00 | y1y0y | y1010 | 11010 | y1yxy | y1y10 | 11y1y | 1y000 | 1y000 | 11000 | 1y0xy | 1101y | - |  |  |
| 1101x | yy0y1 | 1y0y1 | yyyyx | 1yyy0 | y1yy1 | y1010 | 11010 | y1y11 | y1y1x | 11y11 | 1y0yx | 1y0y0 | 110y0 | 1y011 | 11011 | 11010 | - |  |
| 11x11 | yyxy1 | 1y0y1 | yy1y1 | 1y1yy | y11y1 | y1x1y | 1101y | y1111 | y1111 | 11111 | 1y0y1 | 1yxyy | 110yy | 1y011 | 11011 | 1101y | 11011 | - |
| A2 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

Новых кубов не образовалось.

Получено множество Z1*=* {00x01, x0001, 0010x, x0100, 0x101, 01x10, x1010, 011x1, 0111x, x1111, 1000x, 10x00, 1x000, 100x1, 1x011, 110x0, 1101x, 11x11}

На этом заканчивается этап поиска простых импликант:

Z = Z1 = {00x01, x0001, 0010x, x0100, 0x101, 01x10, x1010, 011x1, 0111x, x1111, 1000x, 10x00, 1x000, 100x1, 1x011, 110x0, 1101x, 11x11}

Следующий этап – поиск L-экстремалей на множестве простых импликант (таблица 3.2.4). Для этого используется операция # (вычитание).

Таблица 3.2.4 – Поиск L-экстремалей

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| z#(Z-z) | 00x01 | x0001 | 0010x | x0100 | 0x101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 1000x | 10x00 | 1x000 | 100x1 | 1x011 | 110x0 | 1101x | 11x11 |
| 00x01 | - | 10001 | 00100 | x0100 | 01101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 1000x | 10x00 | 1x000 | 100x1 | 1x011 | 110x0 | 1101x | 11x11 |
| x0001 | 00101 | - | 00100 | x0100 | 01101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 10000 | 10x00 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| 0010x | Ø | 10001 | - | 10100 | 01101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 10000 | 10x00 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| x0100 |  | 10001 | Ø | - | 01101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| 0x101 |  | 10001 |  | 10100 | - | 01x10 | x1010 | 01111 | 0111x | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| 01x10 |  | 10001 |  | 10100 | 01101 | - | 11010 | 01111 | 01111 | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| x1010 |  | 10001 |  | 10100 | 01101 | 01110 | - | 01111 | 01111 | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11x11 |
| 011x1 |  | 10001 |  | 10100 | Ø | 01110 | 11010 | - | Ø | 11111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11x11 |
| 0111x |  | 10001 |  | 10100 |  | Ø | 11010 | Ø | - | 11111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11x11 |
| x1111 |  | 10001 |  | 10100 |  |  | 11010 |  |  | - | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11011 |
| 1000x |  | Ø |  | 10100 |  |  | 11010 |  |  | 11111 | - | Ø | 11000 | 10011 | 1x011 | 11000 | 11011 | 11011 |
| 10x00 |  |  |  | Ø |  |  | 11010 |  |  | 11111 | Ø | - | 11000 | 10011 | 1x011 | 11000 | 11011 | 11011 |
| 1x000 |  |  |  |  |  |  | 11010 |  |  | 11111 |  |  | - | 10011 | 1x011 | Ø | 11011 | 11011 |
| 100x1 |  |  |  |  |  |  | 11010 |  |  | 11111 |  |  | 11000 | - | 11011 |  | 11011 | 11011 |
| 1x011 |  |  |  |  |  |  | 11010 |  |  | 11111 |  |  | 11000 | Ø | - |  | Ø | Ø |
| 110x0 |  |  |  |  |  |  | Ø |  |  | 11111 |  |  | Ø |  | 11011 | - |  |  |
| 1101x |  |  |  |  |  |  |  |  |  | 11111 |  |  |  |  | Ø |  | - |  |
| 11x11 |  |  |  |  |  |  |  |  |  | Ø |  |  |  |  |  |  |  | - |
| Остаток | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

В таблице 3.2.4 из каждой простой импликанты поочерёдно вычитаются все остальные простые импликанты *Z#(Z-z)*.

E = { Ø }.

После операции *Z#(Z-z)* остатки не образованы, необходимо применить алгоритм ветвления. Так как все кубы имеют одинаковую ценность и все одинаково пересекаются с множеством L, то выбираем любой набор и выполняем операцию *Z#(Z-z)* без него.

Таблица 3.2.5 – Поиск L-экстремалей

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x0001 | 0010x | x0100 | 0x101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 1000x | 10x00 | 1x000 | 100x1 | 1x011 | 110x0 | 1101x | 11x11 |
| x0001 | - | 0010x | x0100 | 0x101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 10000 | 10x00 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| 0010x | x0001 | - | 10100 | 01101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 10000 | 10x00 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| x0100 | x0001 | 00101 | - | 01101 | 01x10 | x1010 | 011x1 | 0111x | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| 0x101 | x0001 | Ø | 10100 | - | 01x10 | x1010 | 01111 | 0111x | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| 01x10 | x0001 |  | 10100 | 01101 | - | 11010 | 01111 | 01111 | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 110x0 | 1101x | 11x11 |
| x1010 | x0001 |  | 10100 | 01101 | 01110 | - | 01111 | 01111 | x1111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11x11 |
| 011x1 | x0001 |  | 10100 | Ø | 01110 | 11010 | - | Ø | 11111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11x11 |
| 0111x | x0001 |  | 10100 |  | Ø | 11010 | Ø | - | 11111 | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11x11 |
| x1111 | x0001 |  | 10100 |  |  | 11010 |  |  | - | 10000 | 10000 | 1x000 | 10011 | 1x011 | 11000 | 11011 | 11011 |
| 1000x | 00001 |  | 10100 |  |  | 11010 |  |  | 11111 | - | Ø | 11000 | 10011 | 1x011 | 11000 | 11011 | 11011 |
| 10x00 | 00001 |  | Ø |  |  | 11010 |  |  | 11111 | Ø | - | 11000 | 10011 | 1x011 | 11000 | 11011 | 11011 |
| 1x000 | 00001 |  |  |  |  | 11010 |  |  | 11111 |  |  | - | 10011 | 1x011 | Ø | 11011 | 11011 |
| 100x1 | 00001 |  |  |  |  | 11010 |  |  | 11111 |  |  | 11000 | - | 11011 |  | 11011 | 11011 |
| 1x011 | 00001 |  |  |  |  | 11010 |  |  | 11111 |  |  | 11000 | Ø | - |  | Ø | Ø |
| 110x0 | 00001 |  |  |  |  | Ø |  |  | 11111 |  |  | Ø |  | 11011 | - |  |  |
| 1101x | 00001 |  |  |  |  |  |  |  | 11111 |  |  |  |  | Ø |  | - |  |
| 11x11 | 00001 |  |  |  |  |  |  |  | Ø |  |  |  |  |  |  |  | - |
| Остаток | 00001 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

Множество L-экстремалей = E = { x0001 }.

Выясним далее, какие из вершин комплекса *L* не покрываются *L*-экстремалями. Для этого из каждого куба комплекса *L* вычтем (#) элементы множества *Е* (табл. 3.2.6).

Таблица 3.2.6 – Поиск не покрытых L-экстремалями кубов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L # E | 00001 | 00100 | 00101 | 01010 | 01101 | 01110 | 01111 | 10000 | 10001 | 10011 | 10100 | 11000 | 11010 | 11011 | 11111 |
| x0001 | - | 00100 | 00101 | 01010 | 01101 | 01110 | 01111 | 10000 | - | 10011 | 10100 | 11000 | 11010 | 11011 | 11111 |

В результате вычитания получим L' = {00100, 00101, 01010, 01101, 01110, 01111, 10000, 10011, 10100, 11000, 11010, 11011, 11111}

Ẑ = Z/E = {00x01, 0010x, x0100, 0x101, 01x10, x1010, 011x1, 0111x, x1111, 1000x, 10x00, 1x000, 100x1, 1x011, 110x0, 1101x, 11x11}

Таблица 3.2.7 – Покрытие кубов множеством Z

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L' ⋂ Ẑ | 00100 | 00101 | 01010 | 01101 | 01110 | 01111 | 10000 | 10011 | 10100 | 11000 | 11010 | 11011 | 11111 |
| 0010x | 00100 | 00101 | - | - | - | - | - | - | - | - | - | - | - |
| x0100 | 00100 | - | - | - | - | - | - | - | 10100 | - | - | - | - |
| 0x101 | - | 00101 | - | 01101 | - | - | - | - | - | - | - | - | - |
| 01x10 | - | - | 01010 | - | 01110 | - | - | - | - | - | - | - | - |
| x1010 | - | - | 01010 | - | - | - | - | - | - | - | 11010 | - | - |
| 011x1 | - | - | - | 01101 | - | 01111 | - | - | - | - | - | - | - |
| 0111x | - | - | - | - | 01110 | 01111 | - | - | - | - | - | - | - |
| x1111 | - | - | - | - | - | 01111 | - | - | - | - | - | - | 11111 |
| 1000x | - | - | - | - | - | - | 10000 | - | - | - | - | - | - |
| 10x00 | - | - | - | - | - | - | 10000 | - | 10100 | - | - | - | - |
| 1x000 | - | - | - | - | - | - | 10000 | - | - | 11000 | - | - | - |
| 100x1 | - | - | - | - | - | - | - | 10011 | - | - | - | - | - |
| 1x011 | - | - | - | - | - | - | - | 10011 | - | - | - | 11011 | - |
| 110x0 | - | - | - | - | - | - | - | - | - | 11000 | 11010 | - | - |
| 1101x | - | - | - | - | - | - | - | - | - | - | 11010 | 11011 | - |
| 11x11 | - | - | - | - | - | - | - | - | - | - | - | 11011 | 11111 |

Поиск минимального покрытия завершён. Так как все простые импликанты имеют одинаковую ценность, методом последовательного исключения из таблицы покрытия кубов(табл. 3.2.7) получили одну из тупиковых форм { 00x01, x0100, 01x10, 011x1, x1111, 1000x, 1x011, 110x0 }.

Следовательно: S1= 122p + 2b12+ 1a2b2 + 1a2b1p + a2b1b2p + a1212 + a11b2p + a1a21

Запишем результат в базисе И, НЕ:

S1 =

Эффективность минимизации:

K = = 5,14

**Минимизация функции S2**

Минимизацию функцииS2 проведем с помощью карт Карно. Для функции S2 заполненная карта приведена на рисунке 3.2.2.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1a2 | 000  b1b2p | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 |  |  |  | 1 | 1 | 1 | 1 |  |
| 01 | 1 |  | 1 | 1 |  | 1 |  |  |
| 11 | 1 | 1 | 1 |  |  |  |  | 1 |
| 10 |  | 1 |  |  | 1 |  | 1 | 1 |

Рисунок 3.2.2 — Минимизация функции S2 картой Карно

Следовательно:

S2= 212 + a112p + a122p + a21b2p + 11b2 + 12b2 + 2b1b2 + 1b1b2p + 12b1p + a1b12

Запишем результат в базисе И, НЕ:

S2 =

Эффективность минимизации:

K = = 1,73

**3.3 Логический синтез преобразователя множителя**

Преобразователь множителя (ПМ) – это устройство, которое преобразовывает диады множителя в соответствии с методом умножения.

При умножении в дополнительных кодах ПМ заменяет диады 11 (34) и 10 (24) на триады и соответственно.

Принцип работы ПМ представлен с помощью таблицы истинности (таблица 3.3.1).

Таблица 3.3.1 – Таблица истинности ПМ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Входная диада** | | **Младший разряд** | **Знак** | **Выходная диада** | |
| ***a*1** | ***a*2** | ***p*** | ***Q*** | ***S*1** | ***S*2** |
| **1** | **2** | **3** | **4** | **5** | **6** |
| 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 |

Минимизацию переключательных функций произведём с помощью карт Вейча и реализуем их в базисе И, ИЛИ, НЕ, ИСКЛЮЧАЮЩЕЕ И.

**Функция Q**

Для функции Q заполненная карта приведена на рисунке 3.3.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | a2 | |  |  |
|  |  |  |  |  |  |
| a1 |  | 1 |  | 1 | 1 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  | p | |  |

Рисунок 3.3.1 – Минимизация функции Q при помощи карты Вейча

Следовательно:

**Функция S1**

Для функции S1 Заполненная карта приведена на рисунке 3.3.2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | *a*2 | |  |  |
|  |  |  |  |  |  |
| *a*1 |  |  |  |  | 1 |
|  |  |  | 1 |  |  |
|  |  |  |  |  |  |
|  |  |  | *p* | |  |

Рисунок 3.3.2 – Минимизация функции S1 при помощи карты Вейча

Следовательно:

**Функция S2**

Для функции S2 заполненная карта приведена на рисунке 3.3.3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | *a*2 | |  |  |
|  |  |  |  |  |  |
| *a*1 |  | 1 |  | 1 |  |
|  |  | 1 |  | 1 |  |
|  |  |  |  |  |  |
|  |  |  | *p* | |  |

Рисунок 3.3.3 – Минимизация функции S2 при помощи карты Вейча

Следовательно:

Функциональная схема ПМ приведена в приложении Д.

# **4 Синтез комбинационных схем устройств**

# **на основе мультиплексоров**

*Мультиплексор* – это логическая схема, которая имеет *n* информационных входов, *m* управляющих входов и один выход. При этом должно выполнятся условие .

На выход мультиплексора может быть пропущен без изменений один любой 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, подаваемым на управляющие входы.

Переключательные функции (ПФ) от пяти переменных (как, например, ОЧС) можно реализовать на мультиплексоре «один из восьми». Управляющее поле такого мультиплексора будет определяться тремя переменными, следовательно, число групп с одинаковыми значениями этих переменных будет равно восьми. Также, реализация нескольких ПФ требует для каждой ПФ отдельного мультиплексора.

Для определения управляющего поля мультиплексора возьмём переменные *a*1, *a*2 и *b*1.

Таблица истинности для синтеза ПФ ОЧС приведена в таблице 4.1.

Таблица 4.1 – Таблица истинности для синтеза ПФ ОЧС

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a1** | **a2** | **b1** | **b2** | **p** | **П** | **Функция** | **S1** | **Функция** | **S2** | **Функция** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| 0 | 0 | 0 | 0 | 0 | 0 |  | 0 |  | 0 |  |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | “0” | 1 |  | 0 |  |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |  | 0 |  | 1 |  |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | “1” | 0 |  | 0 |  |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 |  | 1 |  | 0 |  |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |  | 1 |  | 1 |  |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |  | 1 |  | 1 |  |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |  | 0 |  | 1 |  |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

При синтезе ОЧС на основе мультиплексоров получается более эффективная схема (если считать, что количество входов мультиплексора = n + 2n, не включая входы элементов, из которых он состоит).

Функциональная схема ОЧС на основе мультиплексоров представлена в приложении Г.

**5 ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ**

Формула расчёта временных затрат на умножение:

Т=𝑛∗𝑇ПМ + m∗𝑇ФДК + m∗𝑇 ОЧУC + (m+2)\*𝑇ОЧС + 𝑛∗𝑇сдвига, где

𝑇ПМ – время преобразования множителя;

𝑇ФДК – время формирования дополнительного кода множимого;

𝑇ОЧУC – время умножения на ОЧУC;

𝑇ОЧС – время формирования единицы переноса в ОЧС;

𝑇сдвига – время сдвига в регистрах;

n – количество разрядов множителя;

m – количество разрядов множимого.

Минимизация функций позволила в несколько раз удешевить схему сумматора-умножителя 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.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Луцик, Ю. А. Учебное пособие по курсу «Арифметические и логические основы вычислительной техники» / Ю. А. Луцик, И. В. Лукьянова. - Минск : БГУиР, 2014. – 76с.

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

Единая система конструкторской документации (ЕСКД) : справ. пособие / С. С. Борушек [и др.]. – М. : Изд-во стандартов, 1989. – 352 с.

Лысиков, Б. Г. Цифровая вычислительная техника / Б. Г. Лысиков. – Минск : Выш. шк., 2003. – 242 с.

Савельев, А. Я. Прикладная теория цифровых автоматов / А. Я. Савельев. – М. : Высш. шк., 1987. – 272 с.

**ПРИЛОЖЕНИЕ А**

**(**обязательное**)**

Сумматор-умножитель второго типа. Схема электрическая структурная

**ПРИЛОЖЕНИЕ Б**

**(**обязательное**)**

Одноразрядный четверичный сумматор. Схема электрическая функциональная

**ПРИЛОЖЕНИЕ В**

**(**обязательное**)**

Одноразрядный четверичный сумматор-умножитель. Схема электрическая функциональная

**ПРИЛОЖЕНИЕ Г**

**(**обязательное**)**

Однозарядный четверичный сумматор. Схема электрическая функциональная на основе мультиплексоров

**ПРИЛОЖЕНИЕ Д**

**(**обязательное**)**

Преобразователь множителя. Схема электрическая функциональная

**ПРИЛОЖЕНИЕ Е**

**(**обязательное**)**

Ведомость документов