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 01 312 ПЗ

Студент В. И. Лазакович
Руководитель И. В. Лукьянова

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
Лазаковича Владислава Игоревича

1. Тема работы: «Проектирование и логический синтез сумматора-умножителя двоично-десятичных чисел»
2. Срок сдачи студентом законченной работы: до 20 мая 2022г.
3. Исходные данные к работе:
   - **3.1.** Исходные сомножители: Мн = 23,48; Мт = 29,83;
   - **3.2.** Алгоритм умножения: Г;
   - **3.3.** Метод умножения: умножение закодированного двоично-четверичного множимого на два разряда двоичного множителя одновременно в прямых кодах;
   - **3.4.** Коды четверичных цифр множимого для перехода к двоично-четверичной системе кодирования: 0_4 – 00, 1_4 – 01, 2_4 – 11, 3_4 – 10;
   - **3.5.** Тип синтезируемого умножителя: 2;
   - **3.6.** Логический базис для реализации ОЧС: И, Константная единица, Сумма по модулю; метод минимизации – алгоритм Рота.
   - **3.7.** Логический базис для реализации ОЧУС: 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 основных узлов сумматора-умножителя. 4. Синтез комбинационных схем устройств на основе мультиплексоров. 5. Оценка результатов разработки. Заключение. Список литературы.
5. Перечень графического материала:
   - **5.1.** Умножитель-сумматор 2 типа. Схема электрическая структурная.

5.2.Однозарядный четверичный сумматор. Схема электричская функциональная.

5.3. Однозарядный четверичный умножитель. Схема электрическая функциональная. Однозарядный четверичный умножитель. Схема электрическая функциональная.

5.4. Однозарядный четверичный сумматор. Реализация на мультиплексорах. Схема электрическая функциональная.

5.5.Преобразователь множителя. Схема электрическая функциональная.

КАЛЕНДАРНЫЙ ПЛАН

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

Дата выдачи задания: февраль 2023 г.

Руководитель _____И. В. Лукьянова

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ _____В. И. Лазакович

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **5**

1 РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ **6**

2. РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА – **8**

УМНОЖИТЕЛЯ **8**

3 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА-УМНОЖИТЕЛЯ **11**

3.1 Логический синтез одноразрядного четверичного сумматора-умножителя 11

3.2 Логический синтез одноразрядного четверичного сумматора 23

3.3. Логический синтез преобразователя множителя 27

4.СИНТЕЗ ОЧС НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ **29**

5. ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ **31**

ЗАКЛЮЧЕНИЕ **32**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **33**

ПРИЛОЖЕНИЕ А **34**

ПРИЛОЖЕНИЕ Б **1**

ПРИЛОЖЕНИЕ В **2**

ПРИЛОЖЕНИЕ Г **3**

ПРИЛОЖЕНИЕ Д **4**

ПРИЛОЖЕНИЕ Е **5**

ПРИЛОЖЕНИЕ Ж **6**

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.

```
_ 23 | 4                    0,48
  20   _  5 | 4          *    4
   3       4   1 |          1,92
           1              *    4
                             3,68
                          *    4
                             2,72
```

$Мн_4$ = 113,132.

В соответствии с заданной кодировкой множимого:
$Мн_{2/4}$ =010110,011011

```
_ 29 | 4                    0,83
  28   _  7 | 4          *    4
   1       4   1            3,32
           3              *    4
                             1,28
                          *    4
                             1,12
```

$Мт_4$ =131,311.

В соответствии с заданной кодировкой множителя:
$Мт_{2/4}$ =011101,110101

2. Запишем сомножители в форме с плавающей запятой в прямом коде:
Мн = 0,010110011011 $Р_{Мн}$ = 0.0010 +03_{10}
Мт = 0,011101110101 $Р_{Мт}$ = 0.0011 +03_{10}

Порядок произведения:

$Р_{Мн}$	=	0.0010	03_4
$Р_{Мт}$	=	0.0011	03_4
$Р_{Мн \cdot Мт}$	=	0.0101	12_4

Знак произведения определяется суммой по модулю два знаков сомножителей:

зн Мн $\oplus$ зн Мт = 0 $\oplus$ 0 = 0.

При умножении чисел в прямых кодах диада $11(3_4)$ заменяется на триаду $10\overline{1}$. Преобразованный множитель имеет вид $Мт_4^{п} = 1012\overline{12}$. Перемножение мантисс по алгоритму «Г» представлено в таблице 1.1

Таблица 1.1 — Перемножение мантисс

Четверичная С/С		**Двоично-четверичная С/С**		**Комментарии**
1		**2**		**3**
0. 0.	000000 000000 023233 000000	01. 01.	00000000000000 0000000000 00111011101000 0000000000	$\sum_0^{ч}$ $П_1^{ч} = Мн * 4^{-1}$
0. 3.	023233 000000 332202 020000	01. 01.	00111011101000 0000000000 10101111001100 1100000000	$\sum_1^{ч}$ $П_2^{ч} = 0$
0. 0.	022101 020000 000232 330000	01. 01.	00111101000100 1100000000 00000011101110 1000000000	$\sum_2^{ч}$ $П_3^{ч} = Мн * 4^{-3}$
0. 3.	023000 010000 333322 020200	01. 01.	00111000000000 0100000000 10101010111100 1100110000	$\sum_3^{ч}$ $П_4^{ч} = 2Мн* 4^{-4}$
0. 0.	022322 030200 000001 131320	01. 10.	00111110111100 1000110000 00000000000101 1001101100	$\sum_4^{ч}$ $П_5^{ч} = [-Мн]_д*4^{-5}$
0. 3.	022323 222120 000000 113132	01. 10.	00111110111011 1111011100 00000000000001 0110011011	$\sum_5^{ч}$ $П_6^{ч} =[-2Мн]_д* 4^{-6}$
0.	022330 001312	01.	00111110100000 0001100111	$\sum_6^{ч}$

После окончания умножения необходимо оценить погрешность вычислений. Для этого полученное произведение ($Мн_4 \cdot Мт_4$ = 0,022330001312, $Р_{Мн \cdot Мт} = 12$) приводится к нулевому порядку, а затем переводится в десятичную систему счисления:

$Мн_4 \cdot Мт_4$ = 22330,00131133123 $Р_{Мн \cdot Мт} = 0$;
$Мн_{10} \cdot Мт_{10}$ =700,0288.

Результат прямого перемножения операндов дает следующее:
$Мн_{10} \cdot Мт_{10}$ = 23,48*29,83 =700,4084.

Абсолютная погрешность:
Δ =700,4084 –700,0288 = 0,3796.

Относительная погрешность:

$$\delta = \frac{\Delta}{Мн \cdot Мт} = \frac{0{,}3796}{700{,}0288} = 0{,}0005422 \ (\delta = 0{,}05422\%)$$

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 (см. описание структуры сумматора-умножителя первого типа).

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

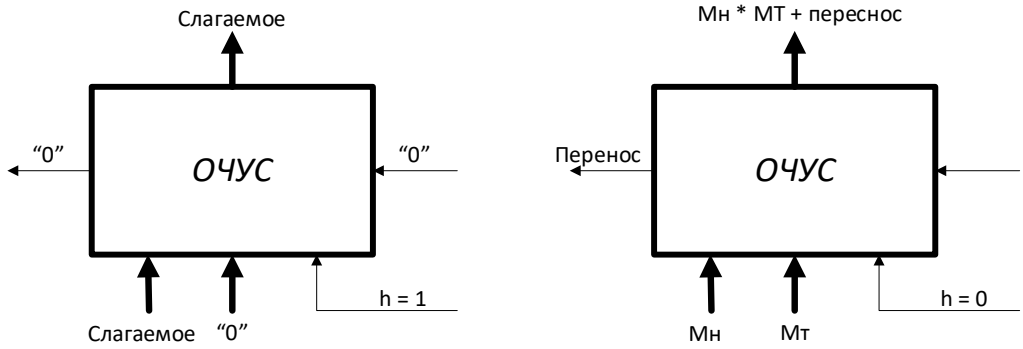


Рисунок 2.1 – Режимы работы ОЧУС

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.

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

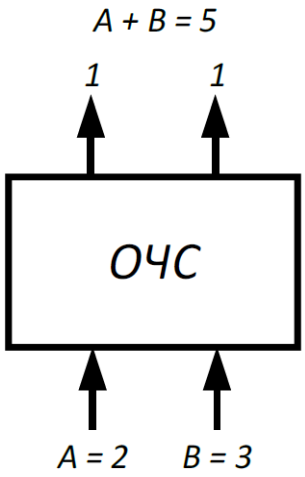


Рисунок 2.2 – Одноразрядный четверичный сумматор

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, 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.

Выход 2 ПМ переходит в единичное состояние, если текущая диада содержит отрицание ($0\bar{1}$). В этом случае инициализируется управляющий вход F1 формирователя дополнительного кода и на выходах ФДК формируется дополнительный код множимого с обратным знаком (умножение на «– 1»).

Принцип работы ФДК в зависимости от управляющих сигналов приведён в таблице 2.2.

Таблица 2.2 – Режимы работы формирователя дополнительного кода.

Сигналы на входах ФДК		Результаты на выходах ФДК
F_1	F_2	
0	0	Дополнительный код множимого
0	1	Дополнительный код слагаемого
1	0	Меняется знак множимого
1	1	Меняется знак слагаемого

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	=	1 2	(+1 в случае переноса из
max		*max*		*max*	предыдущего ОЧУС)
Мн		Мт		перенос	

Так как на входы ОЧУС из регистра Мт не могут поступить коды «3», в таблице истинности работы ОЧУС будут содержаться 36 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).

3 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА-УМНОЖИТЕЛЯ

3.1 Логический синтез одноразрядного четверичного сумматора-умножителя

ОЧУС – это комбинационное устройство, имеющее 5 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 множимого закодированы: 0 – 00, 1 – 01, 2 – 11, 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, 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	x_1	x_2	y_1	y_2	***h***	***P***	Q_1	Q_2	
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	1 * 0 + 0 = 00
0	0	1	0	0	1	0	0	1	Выход – код «01»
0	0	1	0	1	0	0	0	0	1 * 1 + 0 = 01
0	0	1	0	1	1	0	0	1	Выход – код «01»
0	0	1	1	0	0	0	1	1	1 * 2 + 0 = 02
0	0	1	1	0	1	0	1	1	Выход – код «01»
0	0	1	1	1	0	x	x	x	1 * 3 + 0 = 03
0	0	1	1	1	1	x	x	x	Выход – код «01»
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»

Продолжение таблицы 3.1.1

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	1	1	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	1	1	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	x	x	x	0 * 0 + 1 = 00
1	0	0	0	0	1	x	x	x	Выход – код «00»
1	0	0	0	1	0	x	x	x	0 * 1 + 1 = 01
1	0	0	0	1	1	x	x	x	Выход – код «00»
1	0	0	1	0	0	0	0	1	0 * 2 + 1 = 01
1	0	0	1	0	1	x	x	x	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	x	x	x	1 * 0 + 1 = 01
1	0	1	0	0	1	x	x	x	Выход – код «01»
1	0	1	0	1	0	x	x	x	1 * 1 + 1 = 02
1	0	1	0	1	1	x	x	x	Выход – код «01»
1	0	1	1	0	0	1	1	0	1 * 2 + 1 = 03
1	0	1	1	0	1	x	x	x	Выход – код «01»
1	0	1	1	1	0	x	x	x	1 * 2 + 1 = 03
1	0	1	1	1	1	x	x	x	Выход – код «01»
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	2 * 0 + 1 = 01
1	1	1	0	0	1	x	x	x	Выход – код «02»
1	1	1	0	1	0	x	x	x	2 * 1 + 1 = 03

Продолжение таблицы 3.1.1

1	1	1	0	1	1	x	x	x	Выход – код «02»
1	1	1	1	0	0	1	0	1	2 * 2 + 1 = 11
1	1	1	1	0	1	x	x	x	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 проведем с помощью карт Вейча. Для функции P заполненная карта приведена на рисунке 3.1.1. В рисунках 3.1.1 – 3.1.3 символом «x» отмечены наборы, на которых функция может принимать произвольное значение (безразличные наборы).

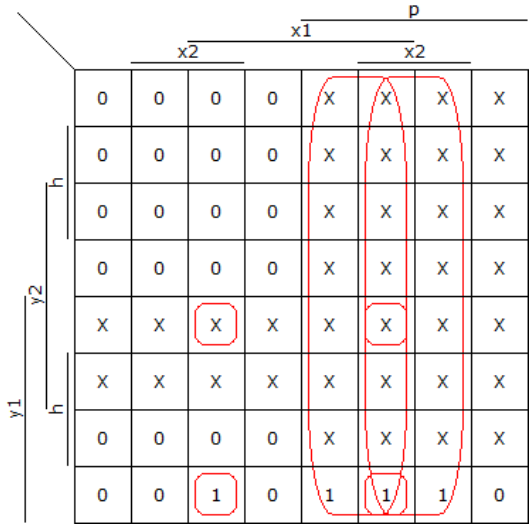


Рисунок 3.1.1 — Минимизация функции P картой Вейча

Следовательно:

$$P = (x_1 x_2 y_1 \overline{h}) + (p x_2) + (p x_1)$$

Запишем результат в базисе ИЛИ-НЕ:

$$P = \overline{\overline{x}_1 + \overline{x}_2 + \overline{y}_1 + h} + \overline{\overline{p} + \overline{x}_2} + \overline{\overline{p} + \overline{x}_1}$$

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

$$K = \frac{6*4+6+4}{1+8+3} = 2{,}83$$

Минимизация функции Q_1:

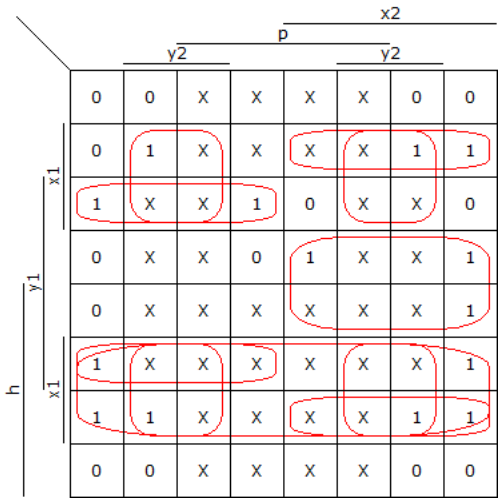


Рисунок 3.1.2 — Минимизация функции Q_1 картой Вейча

Следовательно:

$$Q_1 = (\overline{x}_1 x_2 y_1) + (x_1 h) + (x_1 y_2) + (x_1 x_2 \overline{y}_1) + (x_1 \overline{x}_2 y_1)$$

Запишем результат в базисе ИЛИ-НЕ:

$$Q_1 = \overline{(x_1 + \overline{x}_2 + \overline{y}_1)} + \overline{(\overline{x}_1 + \overline{h})} + \overline{(\overline{x}_1 + \overline{y}_2)} + \overline{(\overline{x}_1 + \overline{x}_2 + y_1)} + \overline{+(\overline{x}_1 + x_2 + \overline{y}_1)}$$

Эффективность минимизации:

$$K = \frac{14*6+14+6}{13+5+3} = 4{,}95$$

Минимизация функции Q_2:

Определим множество единичных кубов:

$$L = \begin{Bmatrix} 001001, 001011, \\ 001100, 001101, \\ 010100, 011000, \\ 011001, 011010, \\ 011011, 011101, \\ 100100, 111100 \end{Bmatrix}$$

Множество безразличных кубов:

$$N = \left\{ \begin{matrix} 000110, 000111, 001110, 001111, 010110, 010111, \\ 011110, 011111, 100000, 100001, 100010, 100011, \\ 100101, 100110, 100111, 101000, 101001, 101010, \\ 101011, 101101, 101110, 101111, 110000, 110001, \\ 110010, 110011, 110101, 110110, 110111, 111000, \\ 111001, 111010, 111011, 111101, 111110, 111111 \end{matrix} \right\}$$

Склеенное множество безразличных кубов:

$$N = \{1xx0xx, xxx11x, 1xx1xx\}$$

Сформируем множество $C_0 = L \cup N$:

$$L = \left\{ \begin{matrix} 001001, 001011, \\ 001100, 001101, \\ 010100, 011000, \\ 011001, 011010, \\ 011011, 011101, \\ 100100, 111100, \\ 1xx0xx, xxx11x, \\ 1xx1xx \end{matrix} \right\}$$

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 ($C_0 * C_0$) приведён в таблице 3.2.2

Таблица 3.2.2

C0*C0	001001	001011	001100	001101	010100	011000	011001	011010	011011	011101	1xx0xx	xxx11x	1xx1xx
001001	-												
001011	0010y1	-											
001100			-										
001101	001y01		00110y	-									
010100					-								
011000						-							
011001	0y1001					01100y	-						
011010						0110y0		-					
011011		0y1011					0110y1	01101y	-				

011101				0y1101			011y01			-			
1xx0xx	y01001	y01011				y11000	y11001	y11010	y11011		-		
xxx11x		001y11	0011y0	0011y1	0101y0			011y10	011y11	0111y1	1xxy1x	-	
1xx1xx			y01100	y01101	y10100					y11101	1xxyxx		-
A1	0010x1 001x01 0x1001 x01001	0x1011 x01011 001x11	00110x 0011x0 x01100	0x1101 0011x1 x01101	0101x0 x10100	01100x 0110x0 x11000	0110x1 011x01 x11001	01101x x11010 011x10	x11011 011x11	0111x1 x11101	1xxx1x 1xxxxx	Ø	Ø

В результате данных преобразований получилось множество C_1.

C_1 = {0010x1; 001x01; 0x1001; x01001; 0x1011; x01011; 001x11; 00110x; 0011x0; x01100; 0x1101; 0011x1; x01101; 0101x0; x10100; 01100x; 0110x0; x11000; 0110x1; 011x01; x11001; 01101x; x11010; 011x10; x11011; 011x11; 0111x1; x11101; 1xxxxx; xxx11x

Множество Z_0 пустое.

В таблице 3.2.3 приведён следующий шаг поиска простых импликант с помощью операции $C_1 * C_1$.

В результате данных преобразований получилось множество C_2.

C_2 = {001xx1; 0x10x1; x010x1; 0x1x01; x01x01; xx1001; xx1011; 0x1x11; x01x11; x0110x; 0011xx; x011x0; xx1101; 0x11x1; x011x1; x101x0; 0110xx; x1100x; x110x0; 011xx1; x110x1; x11x01; x1101x; 011x1x; x11x10; x11x11; x111x1; 1xxxxx; xxx11x}

Множество Z_1 пустое

В таблице 3.2.4 приведён следующий шаг поиска простых импликант – операция C_2*C_2

В результате данных преобразований получилось множество C_3

C_3 = {0x1xx1; x01xx1; xx10x1; xx1x01; xx1x11; x011xx; xx11x1; x110xx; x11xx1; x11x1x; 1xxxxx; xxx11x}

Z_2 = {x101x0}

Таблица 3.2.4

C2*C2	001xx1	0x10x1	x010x1	0x1x01	x01x01	xx1001	xx1011	0x1x11	x01x11	x0110x	0011xx	x011x0	xx1101	0x11x1	x011x1	x101x0	0110xx	x1100x	x110x0	011xx1	x110x1	x11x01	x1101x	011x1x	x11x10	x11x11	x111x1
001xx1	-																										
0x10x1		-																									
x010x1			-																								
0x1x01				-																							
x01x01					-																						
xx1001						-																					
xx1011						xx10y1	-																				
0x1x11				0x1xy1				-																			
x01x11					x01xy1				-																		
x0110x										-																	
0011xx											-																
x011x0												-															
xx1101						xx1y01							-														
0x11x1		0x1yx1												-													
x011x1			x01yx1									x011xy			-												
x101x0																-											
0110xx																	-										
x1100x																		-									
x110x0																			-								
011xx1	0y1xx1																			-							
x110x1			xy10x1																x110xy		-						
x11x01					xy1x01																	-					
x1101x																		x110yx					-				
011x1x																								-			
x11x10																									-		
x11x11									xy1x11															x11xy1			x11x1y	-	
x111x1															xy11x1						x11yx1						-
1xxxxx	y01xx1	yx10x1		yx1x01				yx1x11			y011xx			yx11x1			y110xx			y11xx1				y11x1x			
xxx11x							xx1y11			x011yx			xx11y1										x11y1x				
A3	0x1xx1	0x1xx1	x01xx1	0x1xx1	x01xx1	xx10x1	xx1x11	xx1x11	xx1x11	x011xx	x011xx	x011xx	xx11x1	xx11x1	xx11x1	Ø	x110xx	x110xx	x110xx	x11xx1	x11xx1	x11xx1	x11x1x	x11x1x	x11x1x	Ø	Ø
	x01xx1	xx10x1	xx10x1	xx1x01	xx1x01	xx1x01																					

Таблица 3.2.3

C1*C1	0010x1	001x01	0x1001	x01001	0x1011	x01011	001x11	00110x	0011x0	x01100	0x1101	0011x1	x01101	0101x0	x10100	01100x	0110x0	x11000	0110x1	011x01	x11001	01101x	x11010	011x10	x11011	011x11	0111x1	x11101	1xxxxx
0010x1	-																												
001x01		-																											
0x1001			-																										
x01001				-																									
0x1011			0x10y1		-																								
x01011				x010y1		-																							
001x11		001xy1					-																						
00110x								-																					
0011x0									-																				
x01100										-																			
0x1101			0x1y01								-																		
0011x1	001yx1								0011xy			-																		
x01101				x01y01						x0110y			-																	
0101x0														-																
x10100															-														
01100x																-													
0110x0																	-												
x11000																		-											
0110x1	0y10x1																0110xy		-										
011x01		0y1x01																		-									
x11001				xy1001														x1100y			-								
01101x																0110yx						-							
x11010																		x110y0					-						
011x10																								-					
x11011						xy1011															x110y1		x1101y		-				
011x11							0y1x11													011xy1				011x1y		-			
0111x1												0y11x1							011yx1								-		
x11101													xy1101								x11y01							-	
1xxxxx	y010x1	y01x01	yx1001		yx1011		y01x11	y0110x	y011x0		yx1101	y011x1		y101x0		y1100x	y110x0		y110x1	y11x01		y1101x		y11x10		y11x11	y111x1		-
xxx11x					0x1y11	x01y11		0011yx		x011y0	0x11y1		x011y1		x101y0							011y1x	x11y10		x11y11			x111y1	
A2	001xx1	001xx1	0x10x1	x010x1	xx1011	xx1011	0x1x11	x0110x	0011xx	x0110x	xx1101	0x11x1	xx1101	x101x0	x101x0	0110xx	0110xx	x1100x	011xx1	011xx1	x110x1	x1101x	x1101x	011x1x	x11x11	x11x11	x111x1	x111x1	Ø
	0x10x1	0x1x01	0x1x01	x01x01	0x1x11	x01x11	x01x11	0011xx	x011x0	x011x0	0x11x1	x011x1	x011x1			x1100x	x110x0	x110x0	x110x1	x11x01	x11x01	011x1x	x11x10	x11x10					
	x010x1	x01x01	xx1001	xx1001																									

В таблице 3.2.5 приведён следующий шаг поиска простых импликант – операция C_3*C_3

Таблица 3.2.5

C3*C3	0x1xx1	x01xx1	xx10x1	xx1x01	xx1x11	x011xx	xx11x1	x110xx	x11xx1	x11x1x
0x1xx1	-									
x01xx1		-								
xx10x1			-							
xx1x01				-						
xx1x11				xx1xy1	-					
x011xx						-				
xx11x1			xx1yx1				-			
x110xx								-		
x11xx1		xy1xx1							-	
x11x1x										-
1xxxxx	yx1xx1									
xxx11x										
A4	xx1xx1	xx1xx1	xx1xx1	xx1xx1	Ø	Ø	Ø	Ø	Ø	Ø

В результате данных преобразований получилось множество C_4

C_4 = {xx1xx1; 1xxxxx}

Множество Z_3 пустое.

В таблице 3.2.6 приведён следующий шаг поиска простых импликант с помощью операции $C_4 * C_4$.

Таблица 3.2.6

C4*C4	xx1xx1
xx1xx1	-
1xxxxx	
A5	Ø

На этом процесс выявления простых импликант окончен. Таким образом сформировано множество простых импликант

$Z = Z_0 \cup Z_1 \cup Z_2 \cup Z_3$ = {x101x0; x011xx; x110xx; x11x1x; xxx11x; xx1xx1; 1xxxxx}

Следующий этап – поиск L-экстремалей на множестве простых импликант (таблица 3.2.7). Для этого используется операция # (решетчатое вычитание).

Множество Z может быть избыточным. Прежде всего необходимо выявить обязательные простые импликанты, называемые в алгоритме извлечения L-экстремалями. L-экстремаль – это куб, который (и только он) покрывает не которую вершину из множества L, не покрываемую никаким другим кубом из множества Z.

Таблица 3.2.7

z#(Z-z)	x101x0	x011xx	x110xx	x11x1x	xx1xx1	1xxxxx
x101x0	-	x011xx	x110xx	x11x1x	xx1xx1	10xxxx 1x1xxx 1xx0xx 1xxxx1
x011xx	x101x0	-	x110xx	x11x1x	x11xx1 xx10x1	100xxx 10x0xx 111xxx 1x10xx 1xx0xx 11xxx1 1x0xx1 1xx0x1
x110xx	x101x0	x011xx	-	x1111x	x111x1 x010x1	100xxx 10x0xx 1111xx 1010xx 10x0xx 1x00xx 110xx1 11x1x1 1x0xx1 10x0x1 1x00x1
x11x1x	x101x0	x011xx	x1100x	-	x11101 x010x1	100xxx 10x0xx 11110x 1010xx 10x0xx 1x00xx 110xx1 1101x1 11x101 1x0xx1 10x0x1 1x00x1

xxx11x	x10100	x0110x	x1100x	Ø	x11101 x010x1	1000xx 100x0x 10x0xx 11110x 1010xx 10x0xx 1x00xx 1100x1 110x01 110101 11x101 1x00x1 1x0x01 10x0x1 1x00x1
xx1xx1	x10100	x01100	x11000	Ø	-	1000xx 100x0x 1000xx 10x0x0 111100 1010x0 1000xx 10x0x0 1x00xx 1100x1 110x01 110101 110101 1x00x1 1x0x01 1000x1 1x00x1
1xxxxx	010100	001100	011000	Ø	011101 0010x1	-
Остаток	010100	001100	011000	Ø	011101 0010x1	1000xx 100x0x 1000xx 10x0x0 111100 1010x0 1000xx 10x0x0 1x00xx 1100x1 110x01 110101 110101 1x00x1 1x0x01 1000x1 1x00x1

В таблице 3.2.7 из каждой простой импликанты поочерёдно вычитаются все остальные простые импликанты *Z#(Z-z)*.

Необходимо проверить, нет ли среди полученных *L*-экстремалей таких, которые стали *L*-экстремалями за счёт безразличных кубов. Для этого в таблице 3.2.8 из кубов множества *L* вычитаются остатки простых импликант, полученные в таблице 3.2.8 (результат выполнения операции *Z#(Z\z))*.

По результатам таблицы 3.2.8 *L*-экстремали, не связанные с безразличными наборами, обязательно должны войти в минимальное покрытие.

Таблица 3.2.8

L ⋂ (z # (Z-	001001	001011	001100	001101	010100	011000	011001	011010	011011	011101	100100	111100
010100	-	-	-	-	010100	-	-	-	-	-	-	-
001100	-	-	001100	-	-	-	-	-	-	-	-	-
011000	-	-	-	-	-	011000	-	-	-	-	-	-
00011x	-	-	-	-	-	-	-	-	-	-	-	-
010111	-	-	-	-	-	-	-	-	-	-	-	-
0x0111	-	-	-	-	-	-	-	-	-	-	-	-
011101	-	-	-	-	-	-	-	-	-	011101	-	-
0010x1	001001	001011	-	-	-	-	-	-	-	-	-	-
1000xx	-	-	-	-	-	-	-	-	-	-	-	-
100x0x	-	-	-	-	-	-	-	-	-	-	100100	-
1000xx	-	-	-	-	-	-	-	-	-	-	-	-
10x0x0	-	-	-	-	-	-	-	-	-	-	-	-
111100	-	-	-	-	-	-	-	-	-	-	-	111100
1010x0	-	-	-	-	-	-	-	-	-	-	-	-
1000xx	-	-	-	-	-	-	-	-	-	-	-	-
10x0x0	-	-	-	-	-	-	-	-	-	-	-	-
1x00xx	-	-	-	-	-	-	-	-	-	-	-	-
1100x1	-	-	-	-	-	-	-	-	-	-	-	-
110x01	-	-	-	-	-	-	-	-	-	-	-	-
110101	-	-	-	-	-	-	-	-	-	-	-	-
110101	-	-	-	-	-	-	-	-	-	-	-	-
1x00x1	-	-	-	-	-	-	-	-	-	-	-	-
1x0x01	-	-	-	-	-	-	-	-	-	-	-	-
1000x1	-	-	-	-	-	-	-	-	-	-	-	-
1x00x1	-	-	-	-	-	-	-	-	-	-	-	-

Множество L-экстремалей E = {x101x0; x011xx; x110xx; xx1xx1; 1xxxxx}

В таблице 3.2.9 определяются кубы L, не покрываемые L-экстремалями.

Таблица 3.2.9

L#E	001001	001011	001100	001101	010100	011000	011001	011010	011011	011101	100100	111100
x101x0	001001	001011	001100	001101	-	011000	011001	011010	011011	011101	100100	111100
x011xx	001001	001011	-	-	-	011000	011001	011010	011011	011101	100100	111100
x110xx	001001	001011	-	-	-	-	-	-	-	011101	100100	111100
xx1xx1	-	-	-	-	-	-	-	-	-	-	100100	111100
1xxxxx	-	-	-	-	-	-	-	-	-	-	-	-
Остаток	-	-	-	-	-	-	-	-	-	-	-	-

Множество кубов, не покрываемых L-экстремалями пусто. Таким образом:

$$Q_2 = x_1\overline{x_2}y_1\overline{h} + \overline{x_1}x_2y_1 + x_1x_2\overline{y_1} + x_2h + h.$$

Запишем результат минимизации в логическом ИЛИ, НЕ:

$$Q_2 = \overline{(\overline{x}_1 + x_2 + \overline{y}_1 + \overline{h})} + \overline{(x_1 + \overline{x}_2 + \overline{y}_1)} + \overline{(\overline{x}_1 + \overline{x}_2 + y_1)} + \overline{(\overline{x}_2 + \overline{h})} + h.$$

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 – 01, 2 – 11, 3 – 10;

Таблица 3.2.1 — Таблица истинности ОЧС

a_1	a_2	b_1	b_2	p	$П$	S_1	S_2	*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	0	1	0 + 0 + 1 = 01
0	0	0	1	0	0	0	1	0 + 1 + 0 = 01
0	0	0	1	1	0	1	1	0 + 1 + 1 = 02
0	0	1	0	0	0	1	0	0 + 3 + 0 = 03
0	0	1	0	1	0	1	0	0 + 3 + 1 = 03
0	0	1	1	0	0	1	1	0 + 2 + 0 = 02
0	0	1	1	1	0	1	0	0 + 2 + 1 = 03
0	1	0	0	0	0	0	1	1 + 0 + 0 = 01
0	1	0	0	1	0	1	1	1 + 0 + 1 = 02
0	1	0	1	0	0	1	1	1 + 1 + 0 = 02
0	1	0	1	1	0	1	0	1 + 1 + 1 = 03
0	1	1	0	0	0	1	1	2 + 0 + 0 = 02
0	1	1	0	1	0	1	0	2 + 0 + 1 = 03
0	1	1	1	0	0	1	0	1 + 2 + 0 = 03
0	1	1	1	1	1	0	0	1 + 2 + 1 = 10
1	0	0	0	0	0	1	0	3 + 0 + 0 = 03
1	0	0	0	1	1	0	0	3 + 0 + 1 = 10
1	0	0	1	0	1	0	0	3 + 1 + 0 = 10

Продолжение таблицы 3.2.1

1	0	0	1	1	1	0	1	3 + 1 + 1 = 11
1	0	1	0	0	1	1	1	3 + 3 + 0 = 12
1	0	1	0	1	1	1	0	3 + 3 + 1 = 13
1	0	1	1	0	1	0	1	3 + 2 + 0 = 11
1	0	1	1	1	1	1	1	3 + 2 + 1 = 12
1	1	0	0	0	0	1	1	2 + 0 + 0 = 02
1	1	0	0	1	0	1	0	2 + 0 + 1 = 03
1	1	0	1	0	0	1	0	2 + 1 + 0 = 03
1	1	0	1	1	1	0	0	2 + 1 + 1 = 10
1	1	1	0	0	1	0	1	2 + 3 + 0 = 11
1	1	1	0	1	1	1	1	2 + 3 + 1 = 12
1	1	1	1	0	1	0	0	2 + 2 + 0 = 10
1	1	1	1	1	1	0	1	2 + 2 + 1 = 11

Минимизация функции S_2:

Определим множество единичных кубов:

$$L = \begin{Bmatrix} 00001, 00010, 00011, \\ 00110, 01000, 01001, \\ 01010, 01100, 10011, \\ 10100, 10110, 10111, \\ 11000, 11100, 11101 \\ 11111 \end{Bmatrix}$$

Множество безразличных кубов пустое.

Сформируем множество $C_0 = L \cup N$:

$C_0 = \{00001, 00010, 00011, 00110, 01000, 01001, 01010, 01100, 10011, 10100, 10110, 10111, 11000, 11100, 11101, 11111\}$

Первым этапом алгоритма Рота является нахождение множества простых импликант.

Для реализации этого этапа будем использовать операцию умножения (*) над множествами C_0, C_1 и т. д., пока в результате операции будут образовываться новые кубы большей размерности.

Первый шаг умножения ($C_0 * C_0$) приведён в таблице 3.2.2.

Минимизация функции S_1

Минимизацию функции **S_1** проведем с помощью карт Карно. Для функции **S_1** заполненная карта приведена на рисунке 3.2.1.

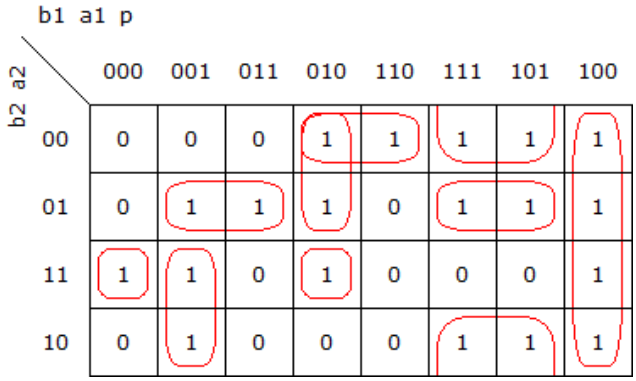


Рисунок 3.2.1 — Минимизация функции S_1 картой Карно

Следовательно: $S_1 = (a_2\overline{b}_2p) + (\overline{a}_1b_1\overline{p}) + (\overline{a}_1\overline{b}_1b_2p) + (a_2\overline{b}_1b_2\overline{p}) + (a_1\overline{b}_1\overline{b}_2\overline{p}) + (\overline{a}_2b_1p) + (a_1\overline{a}_2\overline{b}_2\overline{p})$

Запишем результат в базисе И-Константная единица-Сумма по модулю:

$S_1 = ((a_1 \cdot b_1 \cdot p) \oplus 1) \cdot (((a_1 \oplus 1) \cdot b_1 \cdot (p \oplus 1)) \oplus 1) \cdot ((a_1 \cdot (b_1 \oplus 1) \cdot (p \oplus 1)) \oplus 1) \cdot (((a_1 \oplus 1) \cdot (b_1 \oplus 1) \cdot p) \oplus 1)$

Эффективность минимизации:

$$K = \frac{19*5+19+5}{37} = 3{,}22$$

Минимизация функции S_2

Минимизацию функции **S_2** проведем с помощью карт Карно. Для функции **S_2** заполненная карта приведена на рисунке 3.2.2.

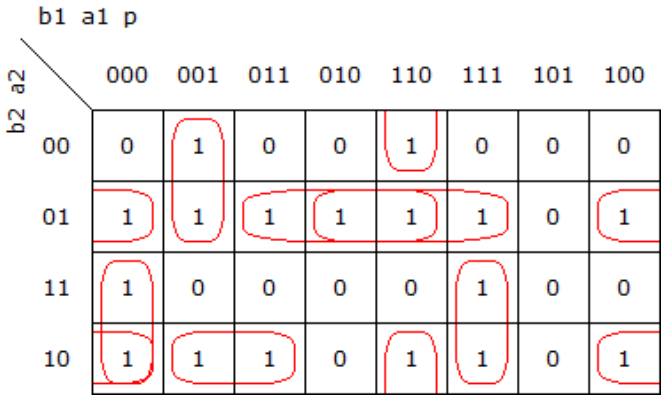


Рисунок 3.2.2 — Минимизация функции S_2 картой Карно

Следовательно: $S_2= (\overline{a}_1\overline{b}_1\overline{b}_2p) + (\overline{a}_2\overline{b}_1b_2p) + (a_1b_1b_2p) + (a_1a_2\overline{b}_2) + (a_1\overline{a}_2b_1\overline{p}) + (\overline{a}_1\overline{b}_1b_2\overline{p}) + (\overline{a}_1\overline{a}_2b_2\overline{p}) + (a_2\overline{b}_2\overline{p})$

Запишем результат в базисе И-Константная единица-Сумма по модулю:

$S_2 = ((a_1 \cdot b_1 \cdot b_2 \cdot (p \oplus 1)) \oplus 1) \cdot ((a_2 \cdot b_1 \cdot (b_2 \oplus 1) \cdot (p \oplus 1)) \oplus 1) \cdot (((a_1 \oplus 1) \cdot (b_1 \oplus 1) \cdot (b_2 \oplus 1) \cdot (p \oplus 1)) \oplus 1) \cdot (((a_1 \oplus 1) \cdot (a_2 \oplus 1) \cdot b_2) \oplus 1) \cdot (((a_1 \oplus 1) \cdot a_2 \cdot (b_1 \oplus 1) \cdot p) \oplus 1) \cdot ((a_1 \cdot b_1 \cdot (b_2 \oplus 1) \cdot p) \oplus 1) \cdot ((a_1 \cdot a_2 \cdot (b_2 \oplus 1)) \oplus 1)$

Эффективность минимизации:

$$K = \frac{16*5+16+5}{43} = 2$$

Минимизация функции П

Минимизацию функции П проведем с помощью карт Карно. Для функции П заполненная карта приведена на рисунке 3.2.3.

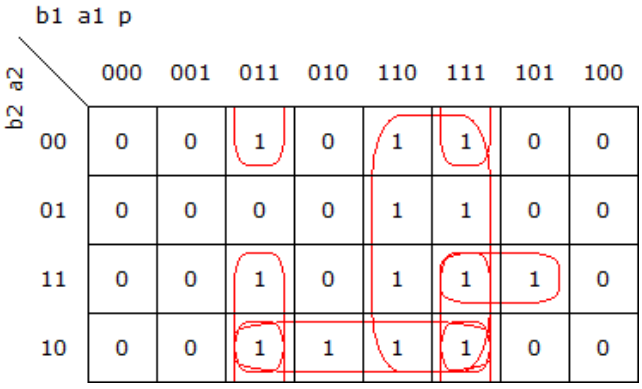


Рисунок 3.2.3 — Минимизация функции П картой Карно

Следовательно:

$П= (a_2b_1b_2p) + (a_1\overline{a}_2p) + (a_1\overline{a}_2b_2) + (a_1b_2p) + (a_1b_1)$

Запишем результат в базисе И-Константная единица-Сумма по модулю:

$П = (((a_2 \oplus 1) \cdot (b_1 \oplus 1) \cdot (b_2 \oplus 1) \cdot (p \oplus 1) \oplus 1) \cdot (((a_1 \oplus 1) \cdot a_2 \cdot (p \oplus 1)) \oplus 1) \cdot (((a_1 \oplus 1) \cdot a_2 \cdot (b_2 \oplus 1)) \oplus 1) \cdot (((a_1 \oplus 1) \cdot (b_2 \oplus 1) \cdot (p \oplus 1)) \oplus 1) \cdot (((a_1 \oplus 1) \cdot (b_1 \oplus 1)) \oplus 1)$

Эффективность минимизации:

$$K = \frac{16*5+16+5}{92} = 1{,}1$$

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 (3_4) и 10 (2_4) на триады $1\overline{01}$ и $1\overline{10}$.

Таблица 3.3.1 – Таблица истинности ПМ

Q1	Q2	П	S	y1	y2	Пример операции в четверичной с/с
1	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	000 → +00
0	0	1	0	0	1	001 → +01
0	1	0	0	0	1	001 → +01
0	1	1	1	1	0	010 → -10
1	0	0	1	1	0	010 → -10
1	0	1	1	0	1	011 → -01
1	1	0	1	0	1	011 → -01
1	1	1	1	0	0	100 → +00

Функция *S*:

Для функции *S* заполненная карта Вейча приведена на рисунке 3.3.2.

Q1 \ Q2П	00	01	11	10
0	0	0	1	0
1	1	1	1	1

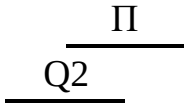
Рисунок 3.3.2 – Минимизация функции *S* при помощи карты Вейча

Следовательно:

$$S = Q1$$

Функция y_1

Для функции y_1 заполненная карта Карно приведена на рисунке 3.3.3.



Q1				
0	0	0	0	1
0	1	0	0	

Рисунок 3.3.3 – Минимизация функции y_1 при помощи карты Карно

Следовательно:

$$y_1 = Q1\overline{Q2}\overline{П} + \overline{Q1}Q2П$$

Функция y_2

Для функции y_2 заполненная карта Вейча приведена на рисунке 3.3.4.

Q1 \ Q2П	00	01	11	10
0	0	1	0	1
1	0	1	0	1

Рисунок 3.3.4 – Минимизация функции y_2 при помощи карты Вейча

Следовательно:

$$y_2 = Q2\overline{П} + \overline{Q2}П$$

4.СИНТЕЗ ОЧС НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ

Мультиплексор – это логическая схема, имеющая n входов, m управляющих входов и один выход. При этом должно выполняться равенство $n = 2^m$. На выход мультиплексора может быть пропущен без изменений любой (один) логический сигнал, поступающий на информационные входы. Порядковый номер информационного входа, значение с которого в данный момент должно быть передано на выход, должно быть передано на выход, определяется двоичным кодам на управляющих входах. Для синтеза ОЧС будем использовать мультиплексор “один из восьми” (1 из 8).

Таблица 4.1 – таблица истинности для синтеза ПФ ОЧС

a1	**a2**	**b1**	**b2**	**p**	**П**	**Выход:**	**S1**	**Выход:**	**S2**	**Выход:**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	0	0	0	0	«0»	0	b_2p	0	$b_2 + p$
0	0	0	0	1	0		0		1	
0	0	0	1	0	0		0		1	
0	0	0	1	1	0		1		1	
0	0	1	0	0	0	«0»	1	«1»	0	$\overline{b_2} + p$
0	0	1	0	1	0		1		0	
0	0	1	1	0	0		1		1	
0	0	1	1	1	0		1		0	
0	1	0	0	0	0	«0»	0	$b_2 + p$	1	$\overline{b_2p}$
0	1	0	0	1	0		1		1	
0	1	0	1	0	0		1		1	
0	1	0	1	1	0		1		0	
0	1	1	0	0	0	b_2p	1	$\overline{b_2p}$	1	$\overline{b_2 + p}$
0	1	1	0	1	0		1		0	
0	1	1	1	0	0		1		0	
0	1	1	1	1	1		0		0	
1	0	0	0	0	0	$b_2 + p$	1	$\overline{b_2 + p}$	0	$b_2 + p$
1	0	0	0	1	1		0		0	
1	0	0	1	0	1		0		0	
1	0	0	1	1	1		0		1	
1	0	1	0	0	1	«1»	1	$\overline{b_2} + p$	1	$\overline{\overline{b_2}\,p}$
1	0	1	0	1	1		1		0	
1	0	1	1	0	1		0		1	
1	0	1	1	1	1		1		1	

1	1	0	0	0	0	b_2p	1	$\overline{b_2p}$	1	$\overline{b_2+p}$
1	1	0	0	1	0		1		0	
1	1	0	1	0	0		1		0	
1	1	0	1	1	1		0		0	
1	1	1	0	0	1	«1»	0	$\overline{b_2}\,p$	1	$\overline{b_2}+p$
1	1	1	0	1	1		1		1	
1	1	1	1	0	1		0		0	
1	1	1	1	1	1		0		1	

Функциональная схема ОЧС на мультиплексорах представлена в приложении Г.

5. ОЦЕНКА РЕЗУЛЬТАТОВ РАЗРАБОТКИ

Формула расчёта временных затрат на умножение:

$T = n * (T_{ПМ} + T_{ФДК} + m*T_{ОЧУС} + T_{ОЧС} + T_{сдвига})$, где

$T_{ПМ}$ – время преобразования множителя;

$T_{ФДК}$ – время формирования дополнительного кода множимого;

$T_{ОЧС}$ – время формирования единицы переноса в ОЧС;

$T_{ОЧУС}$ – время формирования единицы переноса в ОЧУС;

$T_{сдвига}$ – 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 первого 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

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

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

Луцик, Ю. А. Учебное пособие по курсу «Арифметические и логические основы вычислительной техники» / Ю. А. Луцик, И. В. Лукьянова, М. П. Ожигина. – Минск : МРТИ, 2001. – 77 с.

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

Усатенко, С. Т. Выполнение электрических схем по ЕСКД : справочник / С. Т. Усатенко, Т. К. Каченюк, М. В. Терехова. – М. : Изд-во стандартов, 1989. – 325 с.

ПРИЛОЖЕНИЕ А

(обязательное**)**

Сумматор-умножитель второго типа. Схема электрическая структурная

Рисунок А.1 – Схема структурная второго типа (алгоритм «Г»)

ПРИЛОЖЕНИЕ Б

(обязательное**)**

Одноразрядный четверичный сумматор. Схема электрическая функциональная

Рисунок Б.1 – Одноразрядный четверичный сумматор. Схема электрическая функциональная

ПРИЛОЖЕНИЕ В

(обязательное**)**

Одноразрядный четверичный умножитель-сумматор. Схема электрическая функциональная

Рисунок В.1 – Одноразрядный четверичный умножитель-сумматор. Схема электрическая функциональная

ПРИЛОЖЕНИЕ Г

(обязательное**)**

Однозарядный четверичный сумматор. Схема электрическая функциональная на основе мультиплексоров

Рисунок Г.1 – Однозарядный четверичный сумматор. Схема электрическая функциональная на основе мультиплексоров

ПРИЛОЖЕНИЕ Д

(обязательное**)**

Однозарядный четверичный сумматор. Схема электрическая функциональная на основе мультиплексоров

Рисунок Д.1 – Однозарядный четверичный сумматор. Схема электрическая функциональная на основе мультиплексоров

ПРИЛОЖЕНИЕ Е

(обязательное**)**

Преобразователь множителя. Схема электрическая функциональная

Рисунок Е.1 – Преобразователь множителя. Схема электрическая функциональная

ПРИЛОЖЕНИЕ Ж
(обязательное**)**
Ведомость документов