Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Арифметические и логические основы

цифровых устройств

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_\_ Ю. А. Луцик

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ПРОЕКТИРОВАНИЕ И ЛОГИЧЕСКИЙ СИНТЕЗ СУММАТОРАУМНОЖИТЕЛЯ ДВОИЧНО-ЧЕТВЕРИЧНЫХ ЧИСЕЛ

БГУИР КР 1-40 02 01 130 ПЗ

Студент А. П. Якуш (гр. 458301)

Руководитель Ю. А. Луцик

МИНСК 2025

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

Якуша Алексея Павловича

1. Тема работы: «Проектирование и логический синтез сумматора-умножителя двоично-десятичных чисел»
2. Срок сдачи студентом законченной работы: 10 мая 2025 г.
3. Исходные данные к работе:
4. исходные сомножители: Мн = 92,27; Мт = 33,34;
5. алгоритм умножения: В;
6. метод умножения – умножение закодированного двоично-четверичного множимого на два разряда двоичного множителя одновременно в дополнительных кодах;
7. тип реализуемой структурной схемы – 1;
8. коды четверичных цифр множимого для перехода к двоично-четверичной системе кодирования:
9. – 00, – 11, – 01, – 10;
10. логический базис для реализации ОЧС: И-НЕ; метод минимизации – алгоритм Рота;
11. логический базис для реализации ОЧУ: И, НЕ; метод минимизации – карты Карно-Вейча;
12. 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:

* Сумматор-умножитель первого типа. Схема электрическая структурная.
* Одноразрядный четверичный сумматор. Схема электрическая функциональная.
* Одноразрядный четверичный умножитель. Схема электрическая функциональная.
* Одноразрядный четверичный сумматор. Реализация на мультиплексорах. Схема электрическая функциональная.
* Преобразователь множителя. Схема электрическая функциональная.

КАЛЕНДАРНЫЙ ПЛАН

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

**Содержание**

Введение

1. РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ
2. РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА-УМНОЖИТЕЛЯ

**Введение**

1. **РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ**
2. Перевод сомножителей из десятичной системы счисления в четверичную

**Множимое**

|  |  |
| --- | --- |
| \* | 0,27 |
| 4 |
| \* | 1,08 |
| 4 |
|  | 0,32 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| - | 92 | 4 |  |  |  |  |
| 8 | - | 23 | 4 |  |  |
| - | 12 | 20 | - | 5 | 4 |
| 12 |  | 3 | 4 | 1 |
|  | 0 |  |  |  | 1 |  |

.

В соответствии с кодировкой множимого:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 33 | 4 |  |  |
| 32 | - | 8 | 4 |
|  | 1 | 8 | 2 |
|  |  | 0 |  |
|  |  |  |  |  |

**Множитель**

|  |  |
| --- | --- |
| \* | 0,34 |
| 4 |
| \* | 1,36 |
| 4 |
| \* | 1,44 |
| 4 |
|  | 1,76 |

.

В соответствии с обычной весомозначной кодировкой множителя:

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

|  |  |  |
| --- | --- | --- |
| Мн = 0,111110001100 |  | – закодировано по заданию |
| Мт = 0,100001010101 | 11 | – закодировано традиционно |

1. Умножение двух чисел с плавающей запятой на два разряда множителя одновременно в прямых кодах

Порядок произведения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | = |  |  |  |
| = | 0.0011 |  |  |
| = | | 0.0111 |  |  |

Результат закодирован в соответствии с заданием на кодировку множимого.

Знак произведения определяется суммой по модулю два знаков сомножителей, т. е.:

зн Мн ⊕ зн Мт = 0 ⊕ 0 = 0.

1. Для умножения мантисс необходимо предварительно преобразовать множитель. При умножении чисел в прямых кодах диада 11() заменяется на триаду , диада 10() заменяется на триаду . Преобразованный множитель имеет вид: или . Перемножение мантисс по алгоритму «В» представлено в таблице 1.1

Таблица 1.1 – Перемножение мантисс

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Четверичная с/с | | Двоичная с/с | | Комментарии |
| 1 | | 2 | | 3 |
| 0 | 000000000000 | 0 | 000000000000000000000000 |  |
| 0 | 000000113010 | 0 | 000000000000111110001100 | Мн |
| 0 | 000000113010 | 0 | 000000000000111110001100 |  |
| 0 | 000001130100 | 0 | 000000000011111000110000 |  |
| 3 | 333333101320 | 1 | 101010101010110011100100 | -2Мн |
| 0 | 000000232020 | 0 | 000000000000011001000100 |  |
| 0 | 000002320200 | 0 | 000000000001100100010000 |  |
| 0 | 000000000000 | 0 | 000000000000000000000000 | 0 |
| 0 | 000002320200 | 0 | 000000000001100100010000 |  |
| 0 | 000023202000 | 0 | 000000000110010001000000 |  |
| 0 | 000000113010 | 0 | 000000000000111110001100 | Мн |
| 0 | 000023321010 | 0 | 000000000110100111001100 |  |
| 0 | 000233210100 | 0 | 000000011010011100110000 |  |
| 0 | 000000113010 | 0 | 000000000000111110001100 | Мн |
| 0 | 000233323110 | 0 | 000000011010100110111100 |  |
| 0 | 002333231100 | 0 | 000001101010011011110000 |  |
| 0 | 000000113010 | 0 | 000000000000111110001100 | Мн |
| 1 | | 2 | | 3 |
| 0 | 003000010110 | 0 | 000010000000001100111100 |  |
| 0 | 030000101100 | 0 | 001000000000110011110000 |  |
| 0 | 000000113010 | 0 | 000000000000111110001100 | Мн |
| 0 | 030000220110 | 0 | 001000000000010100111100 |  |

После окончания умножения необходимо оценить погрешность вычислений. Для этого полученное произведение (= 0,030000220110, = 7) приводится к нулевому порядку, а затем переводится в десятичную систему счисления:

= 0;

= 3074.5195.

Результат прямого перемножения даёт следующее значение:

3076,2818.

Абсолютная погрешность:

.

Относительная погрешность:

1. **РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ СУММАТОРА-УМНОЖИТЕЛЯ**
2. Структура сумматора-умножителя первого типа строится на базе заданных узлов ОЧУ, ОЧС и аккумулятора (накапливающего сумматора). Управление режимами работы схемы осуществляется внешним сигналом Mul/sum, который определяет вид текущей арифметической операции (умножение или сложение).
3. Если устройство работает как сумматор (на входе Mul/sum – «1»), то оба слагаемых последовательно (за два такта) заносятся в регистр множимого, а на управляющие входы h ОЧУ поступает «1».
4. Если устройство работает как умножитель (на входе Mul/sum – «0»), то множимое и множитель помещаются в соответствующие регистры, а на управляющие входы h ОЧУ поступает «0».
5. Частичные суммы хранятся в аккумуляторе, где и происходят их очередные сдвиги.

СТРУКТУРНАЯ СХЕМА СУММАТОРА-УМНОЖИТЕЛЯ ПРЕДСТАВЛЕНА В **ПРИЛОЖЕНИИ A.**

1. **РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА-УМНОЖИТЕЛЯ**
2. **Логический синтез одноразрядного четверичного умножителя**

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

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

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

Управляющий вход *h* определяет тип операции:

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

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

В таблице 3.1 выделено 8 безразличных наборов, так как на входы ОЧУ из разрядов множителя не может поступить код «11».

Таблица 3.1 – таблица истинности ОЧУ

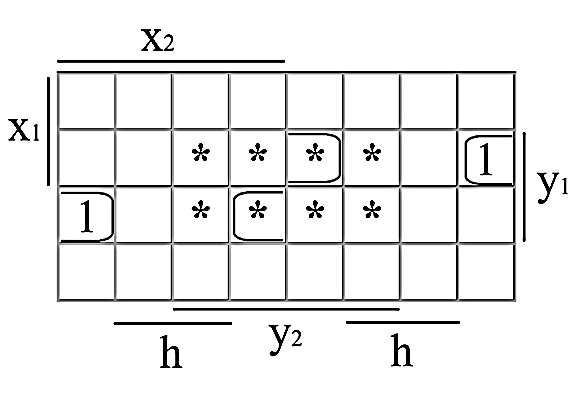
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Мн** | | **Мт** | | **Упр.** | **Старшие разряды** | | **Младшие разряды** | | **Пример операции в четверичной с/с** |
| ***x1*** | ***x2*** | ***y1*** | ***y2*** | ***h*** | ***P1*** | ***P2*** | ***P3*** | ***P4*** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 \* 0 = 00 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | Выход - код «00» |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 \* 1 = 00 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | Выход - код «00» |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 \* 2 = 00 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | Выход - код «00» |
| 0 | 0 | 1 | 1 | 0 | x | x | x | x | 0 \* 3 = 00 |
| 0 | 0 | 1 | 1 | 1 | x | x | x | x | Выход - код «00» |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 \* 0 = 00 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | Выход - код «01» |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 \* 1 = 02 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | Выход - код «01» |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 2 \* 2 = 10 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | Выход - код «01» |
| 0 | 1 | 1 | 1 | 0 | x | x | x | x | 2 \* 3 = 12 |
| 0 | 1 | 1 | 1 | 1 | x | x | x | x | Выход - код «01» |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 \* 0 = 00 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | Выход - код «10» |

*Продолжение таблицы 3.1*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 3 \* 1 = 03 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | Выход – код «10» |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 3 \* 2 = 12 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | Выход – код «10» |
| 1 | 0 | 1 | 1 | 0 | x | x | x | x | 3 \* 3 = 21 |
| 1 | 0 | 1 | 1 | 1 | x | x | x | x | Выход – код «11» |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 \* 0 = 00 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | Выход – код «11» |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 \* 1 = 01 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | Выход – код «11» |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 \* 2 = 02 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | Выход – код «11» |
| 1 | 1 | 1 | 1 | 0 | x | x | x | x | 1 \* 3 = 03 |
| 1 | 1 | 1 | 1 | 1 | x | x | x | x | Выход – код «11» |

Минимизация функций ОЧУ проведена с помощью карт Карно-Вейча.

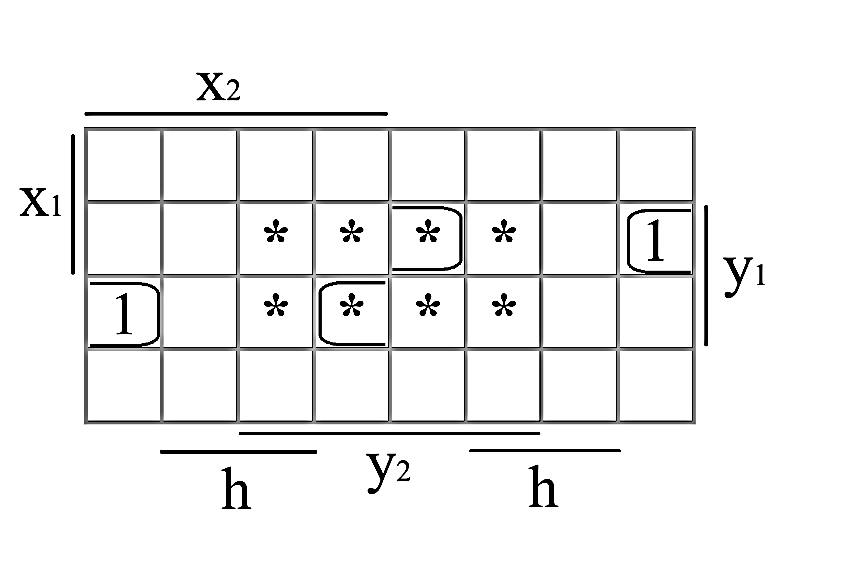
**Карты Вейча минимизации функций ОЧУ**

****Минимизация функции *P1***

*P1* = x2y1+x1y1

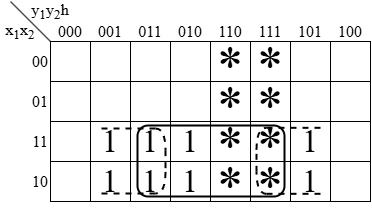
Эффективность минимизации по количеству входов логических элементов:

В базисе И, НЕ: *P1* =

****Минимизация функции *P2***

*P2* = x2y1+x1y1 = *P1*

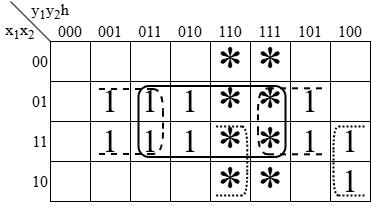
**Карты Карно минимизации функций ОЧУ**

**Минимизация функции *P3***

*P3 =*

Эффективность минимизации по количеству входов логических элементов:

В базисе И, НЕ: *P3 =*

**Минимизация функции *P4***

*P4=*

Эффективность минимизации по количеству входов логических элементов:

В базисе И, НЕ: *P4* =

1. **Логический синтез одноразрядного четверичного сумматора**

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

Принцип работы ОЧС представлен с помощью таблицы истинности (таблица 3.2).

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

В таблице имеется 16 безразличных наборов т.к. со старших разрядов ОЧУ не могут прийти коды «2» и «3».

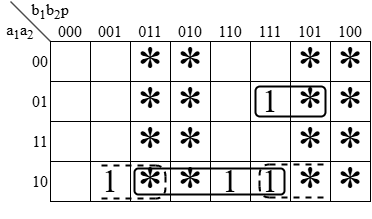
Таблица 3.2 – таблица истинности ОЧС

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

*Продолжение таблицы 3.2*

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

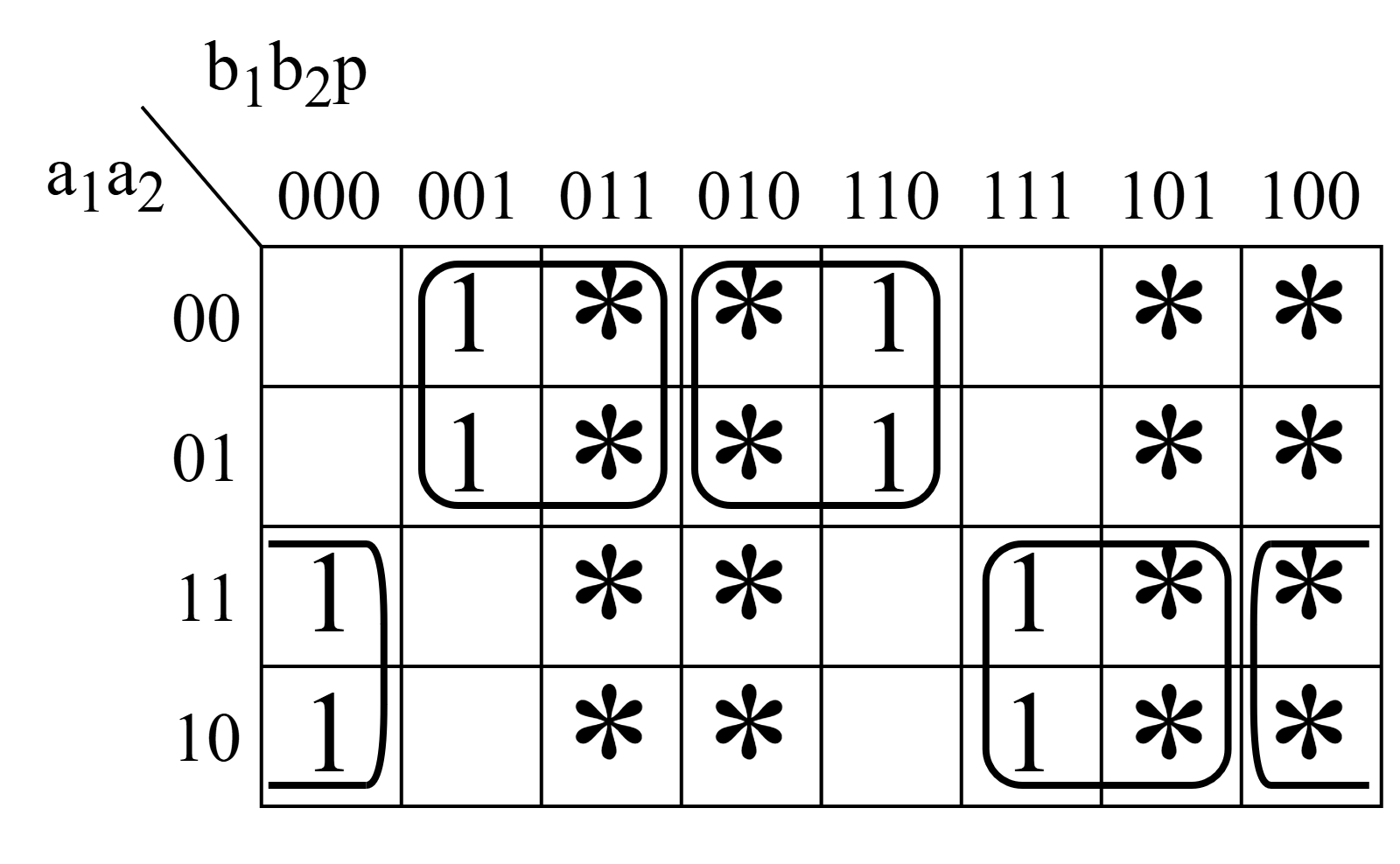
**Карты Карно минимизации функций ОЧС**

**Минимизация функции П**

П =

Эффективность минимизации по количеству входов логических элементов:

В базисе И-НЕ: П = ;

**Минимизация функции S1**

S1=

Эффективность минимизации по количеству входов логических элементов:

В базисе И-НЕ: S1=;

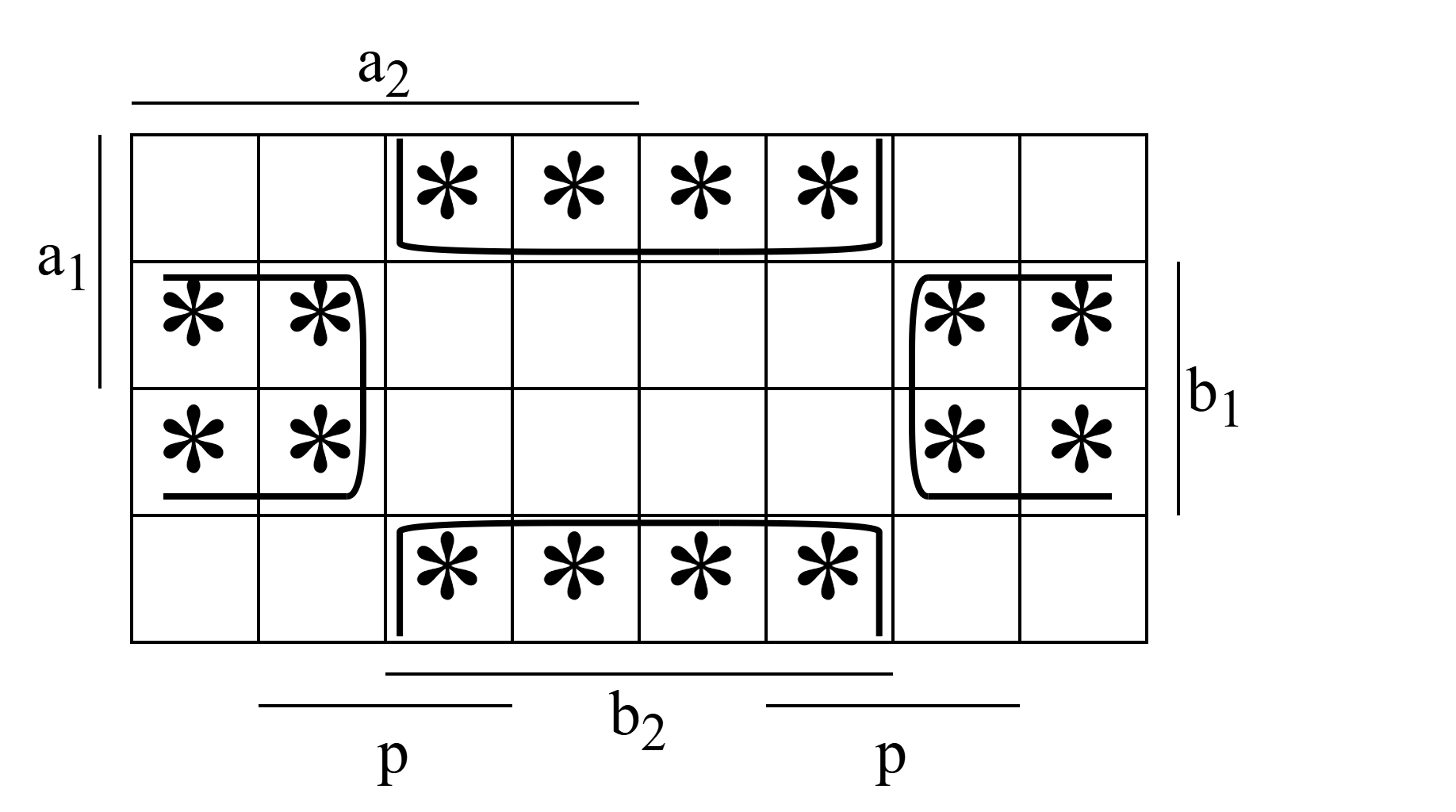
**Минимизация функции S2**

Определим множество единичных и безразличных кубов

L: {00001, 00110, 00111, 01000, 10111, 11000, 11001, 11110},

N: {00010, 00011, 00100, 00101, 01010, 01011, 01100, 01101, 10010, 10011, 10100, 10101, 11010, 11011, 11100, 11101}.

Минимизируем множество N безразличных кубов с помощью карт Вейча, в результате чего получим множество безразличных кубов N:



N =  = {xx01x, xx10x}.

Сформируем множество С0 = L U N:

С0 = {00001, 00110, 00111, 01000, 10111, 11000, 11001, 11110, xx01x, xx10x};

Первым этапом алгоритма Рота является нахождение множества Z простых импликант.

Для того, чтобы это выполнить, необходимо произвести операцию умножения (\*) над множествами Ci. Выполнять следует до тех пор, пока будут образовываться новые кубы большей размерности.

Первый шаг умножения (C0\*C0) приведён в таблице 3.2.1.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C0\*C0 | 00001 | 00110 | 00111 | 01000 | 10111 | 11000 | 11001 | 11110 | xx01x | xx10x |
| 00001 | - |  |  |  |  |  |  |  |  |  |
| 00110 |  | - |  |  |  |  |  |  |  |  |
| 00111 |  | 0011y | - |  |  |  |  |  |  |  |
| 01000 |  |  |  | - |  |  |  |  |  |  |
| 10111 |  |  | y0111 |  | - |  |  |  |  |  |
| 11000 |  |  |  | y1000 |  | - |  |  |  |  |
| 11001 |  |  |  |  |  | 1100y | - |  |  |  |
| 11110 |  |  |  |  |  |  |  | - |  |  |
| xx01x | 000y1 | 00y10 | 00y11 | 010y0 | 10y11 | 110y0 | 110y1 | 11y10 | - |  |
| xx10x | 00y01 | 00yx0 | 001y1 | 01y00 | 101y1 | 11y00 | 11y01 | 111y0 |  | - |
| A1: | 000x1 00x01 | 0011x 00x10 001x0 | x0111 00x11 001x1 | x1000 010x0 01x00 | 10x11 101x1 | 1100x 110x0 11x00 | 110x1 11x01 | 11x10 111x0 | ∅ | ∅ |

Таблица 3.2.1 – нахождение простых импликант (C0\*C0).

В результате умножения C0\*C0 сформируем множество A1:

A1 = {000x1, 00x01, 0011x, 00x10, 001x0, x0111, 00x11, 001x1, x1000, 010x0, 01x00, 10x11, 101x1, 1100x, 110x0, 11x00, 110x1, 11x01, 11x10, 111x0, xx01x, xx10x};

На данном этапе умножения простых импликант не было выявлено:

Z0 = ∅;

B1 = C0/Z0 = C0;

Сформируем множество C1 = A1∪B1 кубов первой размерности:

C1 = {000x1, 00x01, 0011x, 00x10, 001x0, x0111, 00x11, 001x1, x1000, 010x0, 01x00, 10x11, 101x1, 1100x, 110x0, 11x00, 110x1, 11x01, 11x10, 111x0, xx01x, xx10x};

Второй шаг умножения (C1\* C1) приведён в таблице 3.2.2.

В результате сформируем множество C2 кубов второй размерности:

A2 = { 00xx1, 00x1x, 001xx, x0x11, x01x1, x10x0, x1x00, 110xx, 11x0x, 11xx0 };

На данном этапе умножения простых импликант не было выявлено:

Z1 = ∅;

B2 = C1/Z1 = C1;

C2 = A2UB2 = { 00xx1 00x1x 001xx x0x11 x01x1 x10x0 x1x00 110xx 11x0x 11xx0 xx01x xx10x };

Следующий шаг умножения (C2\* C2) приведён в таблице 3.2.3.

Таблица 3.2.3 – нахождение простых импликант (C2\*C2).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C2\*C2 | 00xx1 | 00x1x | 001xx | x0x11 | x01x1 | x10x0 | x1x00 | 110xx | 11x0x | 11xx0 |
| 00xx1 | - |  |  |  |  |  |  |  |  |  |
| 00x1x |  | - |  |  |  |  |  |  |  |  |
| 001xx |  |  | - |  |  |  |  |  |  |  |
| x0x11 |  |  |  | - |  |  |  |  |  |  |
| x01x1 |  |  |  |  | - |  |  |  |  |  |
| x10x0 |  |  |  |  |  | - |  |  |  |  |
| x1x00 |  |  |  |  |  |  | - |  |  |  |
| 110xx |  |  |  |  |  |  |  | - |  |  |
| 11x0x |  |  |  |  |  |  |  |  | - |  |
| 11xx0 |  |  |  |  |  |  |  |  |  | - |
| xx01x |  |  |  |  |  |  |  |  |  |  |
| xx10x |  |  |  |  |  |  |  |  |  |  |
| A3: | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |

Кубов третьей размерности образовано не было. На этом этап поиска простых импликант завершается.

A3 = ∅;

Z2 = {00xx1, 00x1x, 001xx, x0x11, x01x1, x10x0, x1x00, 110xx, 11x0x, 11xx0};

B3 = C2/Z2 = {xx01x, xx10x} = Z3;

Сформируем множество Z простых импликант:

Z = Z0 U Z1 U Z2 U Z3 = { 00xx1, 00x1x, 001xx, x0x11, x01x1, x10x0, x1x00, 110xx, 11x0x, 11xx0, xx01x, xx10x};

Следующий этап – поиск L-экстремалей на множестве простых импликант (таблица 3.2.5). Для этого используется операция вычитания (#).

В таблице 3.2.5 из каждой простой импликанты поочерёдно вычитаются все остальные простые импликанты: z#(Z\z);

Таблица 3.2.5 – поиск L-экстремалей.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| z#(Z\z) | 00xx1 | 00x1x | 001xx | x0x11 | x01x1 | x10x0 | x1x00 | 110xx | 11x0x | 11xx0 | xx01x | xx10x |
| 00xx1 | - | zzzz0 00x10 | zzzz0 001x0 | 1zzzz 10x11 | 1zzzz 101x1 | x10x0 | x1x00 | 110xx | 11x0x | 11xx0 | 11zz0 1x01x x101x xx010 | 11zz0 1x10x x110x xx100 |
| 00x1x | zzz0z 00x01 | - | zzz0z 00100 | 10x11 | 101x1 | x10x0 | x1x00 | 110xx | 11x0x | 11xx0 | 11zzz 1x010 x1010 1x01x x101x | 1x10x x110x xx100 |
| 001xx | 00001 | zz0zz 00010 | - | 10x11 | 101x1 | x10x0 | x1x00 | 110xx | 11x0x | 11xx0 | 1x010 1x01x x101x | 11zzz 1x100 x1100 1x10x x110x |
| x0x11 | 00001 | 00010 | 00100 | - | zzz0z 10101 | x10x0 | x1x00 | 110xx | 11x0x | 11xx0 | z1zz0 1101x 1x010 x101x | 1x100 x1100 1x10x x110x |
| x01x1 | 00001 | 00010 | 00100 | zz1zz 10011 | - | x10x0 | x1x00 | 110xx | 11x0x | 11xx0 | 1101x 1x010 x101x | z1zz0 1110x 1x100 x110x |
| x10x0 | 00001 | 00010 | 00100 | 10011 | 10101 | - | zz1zz x1100 | zzzz1 110x1 | zz1z1 1110x 11x01 | zz1z0 111x0 | zzzz1 z0zzz zzzz1 11011 10010 x1011 | 1110x 1x100 x110x |
| x1x00 | 00001 | 00010 | 00100 | 10011 | 10101 | zzz1z x1010 | - | 110x1 | zzzz1 zzzzy 11101 11x01 | zzz1z 11110 | 10010 x1011 | zzzz1 z0zzz zzzz1 11101 10100 x1101 |
| 110xx | 00001 | 00010 | 00100 | 10011 | 10101 | 0zzzz 01010 | x1100 | - | zzyzz zz1zz 11101 | 11110 | 0zzzz 01011 10010 | 11101 10100 x1101 |
| 11x0x | 00001 | 00010 | 00100 | 10011 | 10101 | 01010 | 0zzzz 01100 | zzz1z 11011 | - | 11110 | 01011 10010 | zzzzz 0zzzz 11101 10100 |
| 11xx0 | 00001 | 00010 | 00100 | 10011 | 10101 | 01010 | 01100 | 11011 | 11101 | - | 01011 10010 | 11101 10100 |
| xx01x | 00001 | zzzzz ∅ | 00100 | zzzzz ∅ | 10101 | zzzzz ∅ | 01100 | zzzzz ∅ | 11101 | 11110 | - | 11101 10100 |
| xx10x | 00001 | ∅ | zzzzz ∅ | ∅ | zzzzz ∅ | ∅ | zzzzz ∅ | ∅ | zzzzz ∅ | 11110 | 01011 10010 | - |
| Остатки: | 00001 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 11110 | 10010 01011 | 11101 10100 |

Результат операции (последняя строка таблицы) указывает на то, что L-экстремалями потенциально могут стать следующие импликанты:

E = {00xx1, 11xx0. xx10x, xx01x};

Необходимо проверить, нет ли среди полученных импликант таких, которые дают остатки от вычитания только из множества N. Для этого в таблице 3.2.6 остатки простых импликант, полученные в таблице 3.2.5 пересекаются с кубами из множества L – операция пересечения ().

Таблица 3.2.6 – определение L-экстремалей.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| z#(Z/z)∩L | 00001 | 00110 | 00111 | 01000 | 10111 | 11000 | 11001 | 11110 |
| 00001 | 00001 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| 11110 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 11110 |
| 10010 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| 01011 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| 01101 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| 10100 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |

По результатам таблицы 3.2.6 L-экстремалями, стали кубы 00хх1, 11xx0 (остатки от вычитания из них всех остальных кубов принадлежат множеству L).

Далее необходимо проанализировать, какие из исходных кубов множествa L не покрыты найденными L-экстремалями. Это делается вычитанием из каждого отдельного куба множества L найденных L-экстремалей (таблица 3.2.7).

Таблица 3.2.7 – нахождение множества L’ непокрытых кубов.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L#E | 00001 | 00110 | 00111 | 01000 | 10111 | 11000 | 11001 | 11110 |
| 00xx1 | zzzzz ∅ | 00110 | zzzzz ∅ | 01000 | 10111 | 11000 | 11001 | 11110 |
| 11xx0 | ∅ | 00110 | ∅ | 01000 | 10111 | zzzzz ∅ | 11001 | zzzzz ∅ |
| Остатки: | ∅ | 00110 | ∅ | 01000 | 10111 | ∅ | 11001 | ∅ |

Из таблицы 3.2.7 видно, что L-экстремалью не покрыты кубы L’ = {00110, 01000, 10111, 11001}. Чтобы их покрыть, воспользуемся множеством простых импликант, не являющихся L-экстремалями (таблица 3.2.8).

= Z/E = {00x1x, 001xx, x0x11, x01x1, x10x0, x1x00, 110xx, 11x0x, xx01x, xx10x};

Таблица 3.2.8 – нахождение покрытия кубов L’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00110 | 01000 | 10111 | 11001 |
| 00x1x | 00110 | ∅ | ∅ | ∅ |
| 001xx | 00110 | ∅ | ∅ | ∅ |
| x0x11 | ∅ | ∅ | 10111 | ∅ |
| x01x1 | ∅ | ∅ | 10111 | ∅ |
| x10x0 | ∅ | 01000 | ∅ | ∅ |
| x1x00 | ∅ | 01000 | ∅ | ∅ |
| 110xx | ∅ | ∅ | ∅ | 11001 |
| 11x0x | ∅ | ∅ | ∅ | 11001 |
| xx01x | ∅ | ∅ | ∅ | ∅ |
| xx10x | ∅ | ∅ | ∅ | ∅ |

Из таблицы 3.2.8 видно, что каждый из четырёх кубов L’ может быть покрыт двумя равнозначными способами, следовательно существуют 16 минимальных форм:

Fmin1 = +++++;

Fmin2 = +++++;

Fmin3 = +++++;

Fmin4 = +++++;

Fmin5 = +++++;

Fmin6 = +++++;

Fmin7 = +++++;

Fmin8 = +++++;

Fmin9 = +++++;

Fmin10 = +++++;

Fmin11 = +++++;

Fmin12 = +++++;

Fmin13 = +++++;

Fmin14 = +++++;

Fmin15 = +++++;

Fmin16 = +++++.

Функциональную схему построим по Fmin1.

Эффективность минимизации по количеству входов логических элементов:

В базисе И-НЕ:

S2 = ;

1. **Логический синтез преобразователя множителя**

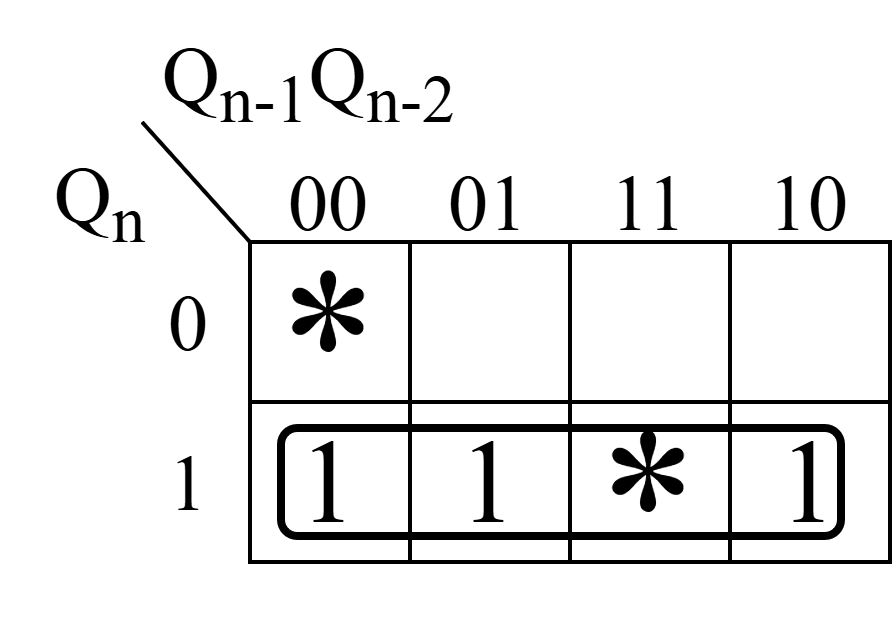
Преобразователь множителя – это устройство, преобразующее диады множителя в соответствии с методом умножения. В результате чего при умножении в дополнительных кодах диады 11(34) и 10(24) заменяются на триады 10(14) и 10(14). Принцип работы ПМ представлен таблицей истинности (таблица 3.3).

Таблица истинности для функции *F1*(знак выходной диады) содержит 2 безразличных набора, т.к. для диады 00(04) знак не важен.

Таблица 3.3 – таблица истинности преобразователя множителя

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Входная диада | | Ст. разряд предыдущей диады | Знак | Выходная диада | |
| ***Q­n*** | ***Qn-1*** | ***Qn-2*** | ***F1*** | ***y1*** | ***y2*** |
| 0 | 0 | 0 | x | 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 | x | 0 | 0 |

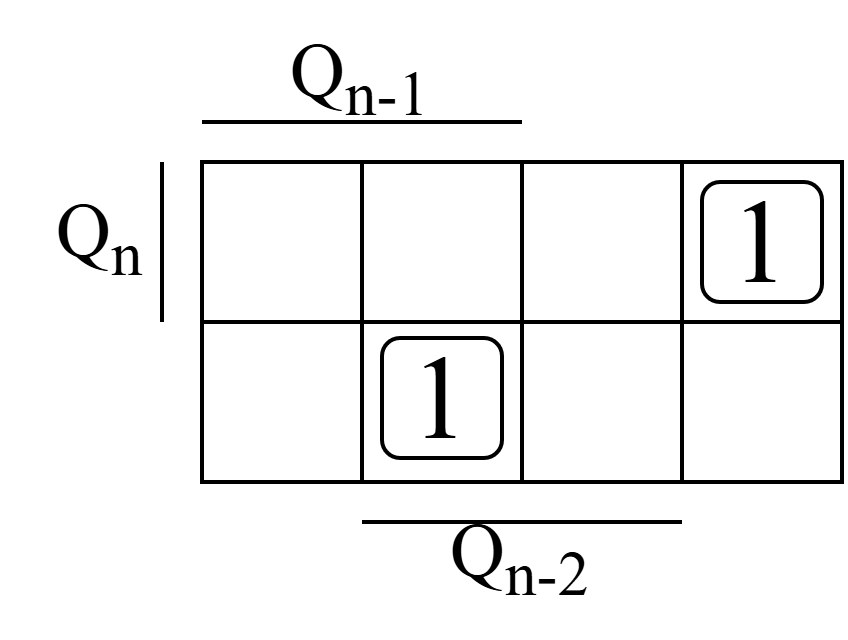
**Карты Карно минимизации функций ПМ**

**Минимизация функции F1**

*F1* = *Qn*

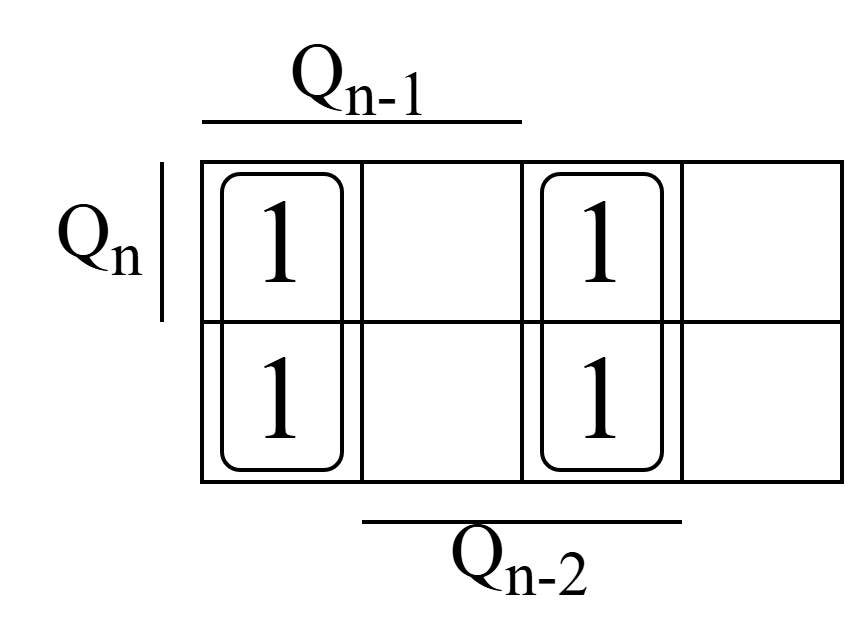
Эффективность минимизации по количеству входов логических элементов:

**Карты Вейча минимизации функций ПМ**

**Минимизация функции y­1**

*y1* =

Эффективность минимизации по количеству входов логических элементов:

**Минимизация функции y­2**

*y2* =

Эффективность минимизации по количеству входов логических элементов:

1. **Синтез комбинационных схем устройства на основе мультиплексоров**

Мультиплексор – логическая схема, имеющая n информационных входов, m управляющих и один выход. При это выполняется условие, что m = 2n. Принцип работы заключается в том, что на выход мультиплексора без изменений пропускается один логический сигнал, выходящий из одного информационного входа. Порядковый номер информационного входа, значение которого должно быть передано на выход определяется двоичным кодом, подаваемым на управляющие выходы.

Переключательную функцию от 5 переменных монжно реализовать на мультимплексоре один из восьми. В таком случае, управляющее поле будет состоять из 3 переменных, следовательно число групп с одинаковыми значениями переменных будет равно восьми.

Реализация нескольких функций требует отдельного мультиплексора для каждой.

Для управляющего поля мультиплексора возьмём перменные a1, a2, b1.

Таблица истинности для синтеза ПФ ОЧС приведена в таблице

Таблица 4 – таблица истинности ПФ ОЧС.

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

*Продолжение таблицы 4.*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| 1 | 0 | 1 | 0 | 0 | x | *“1”* | x | *p* | x | *p* |
| 1 | 0 | 1 | 0 | 1 | x | x | x |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | *“0”* | 1 |  | 1 | *“1”* |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | x | x | x |
| 1 | 1 | 0 | 1 | 1 | x | x | x |
| 1 | 1 | 1 | 0 | 0 | x | *“0”* | x | *p* | x |  |
| 1 | 1 | 1 | 0 | 1 | x | x | x |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |