# **3. Разработка ФУНКЦИональных схем основных узлов сумматора-умножителя**

**3.1. Логический синтез одноразрядного четверичного умножителя**

Одноразрядный четверичный умножитель – это комбинационное устройство, имеющее 5 двоичных входов (2 разряда из регистра Мн, 2 разряда из регистра Мт и управляющий вход *h*) и 4 двоичных выхода.

Принцип работы ОЧУ представлен с помощью таблицы истинности (таблица 3.1).

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

Разряды множимого закодированы: 0 – 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 3.1.1 выделено восемь безразличных наборов, т.к. на входы ОЧУ из разрядов множителя не может поступить код «11».

Таблица 3.1.1 – Таблица истинности ОЧУ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Мн** | | **Мт** | | **Упр.** | **Старшие разряды** | | **Младшие разряды** | | **Пример операции в четверичной с/с** |
| ***x1*** | ***x2*** | ***y1*** | ***y2*** | ***h*** | ***p1*** | ***p2*** | ***p3*** | ***p4*** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0 | 0 | 0 | 0 | 0 | 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 | х | х | x | х | 0·3=00 |
| 0 | 0 | 1 | 1 | 1 | х | x | х | х | Выход – код «00» |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1·0=00 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | Выход – код «00» |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1·1=01 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | Выход – код «01» |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1·2=02 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | Выход – код «02» |
| 0 | 1 | 1 | 1 | 0 | х | x | х | x | 1·3=03 |
| 0 | 1 | 1 | 1 | 1 | x | х | x | х | Выход – код «03» |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2·0=00 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | Выход – код «00» |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 2·1=02 |

*Продолжение таблицы 3.1.1*

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

**Минимизация функции алгоритмом Рота:**

Определим множество единичных кубов

*L* =

и множество безразличных кубов

.

Минимизацию безразличных кубов проведём с помощью карты Карно. Для безразличных кубов заполненная карта приведена на рисунке 3.1.1, где символом «x» отмечены наборы, на которых функция не определена.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | x | x | x | x | x | x | x | x |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | |  |  |  |  |  |
|  |  |  | *h* | |  | | *h* | |  |  |  |

Рисунок 3.1.1 – Минимизация безразличных кубов с помощью карты Карно

Множество безразличных наборов после минимизации:

.

Сформируем множество *С0* = *L* ⋃ *N*:

.

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 3.1.2.

В результате сформируем новое множество кубов:

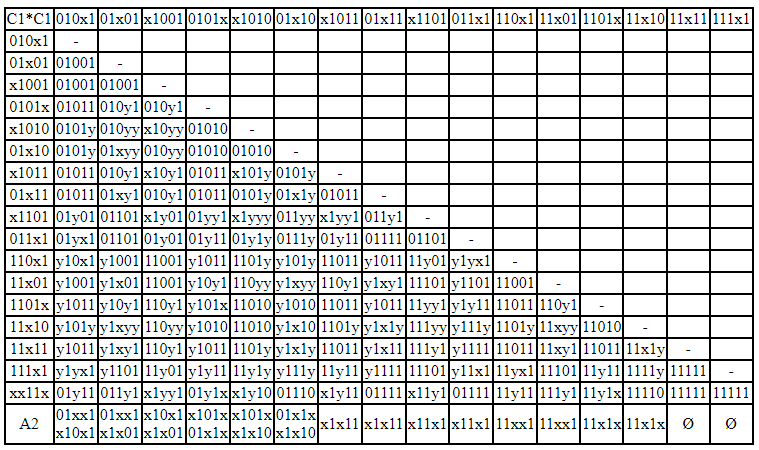
.

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

В таблице 3.1.3 приведён следующий шаг поиска простых импликант с помощью операции *С1\*С1*.

Таблица 3.1.2 – Поиск простых импликант (C0 \* C0)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C0\*C0 | 01001 | 01010 | 01011 | 01101 | 11001 | 11010 | 11011 | 11101 | xx11x |
| 01001 | - |  |  |  |  |  |  |  |  |
| 01010 | 010yy | - |  |  |  |  |  |  |  |
| 01011 | 010y1 | 0101y | - |  |  |  |  |  |  |
| 01101 | 01y01 | 01yyy | 01yy1 | - |  |  |  |  |  |
| 11001 | y1001 | y10yy | y10y1 | y1y01 | - |  |  |  |  |
| 11010 | y10yy | y1010 | y101y | y1yyy | 110yy | - |  |  |  |
| 11011 | y10y1 | y101y | y1011 | y1yy1 | 110y1 | 1101y | - |  |  |
| 11101 | y1y01 | y1yyy | y1yy1 | y1101 | 11y01 | 11yyy | 11yy1 | - |  |
| xx11x | 01yy1 | 01y10 | 01y11 | 011y1 | 11yy1 | 11y10 | 11y11 | 111y1 | - |
| A1 | 010x1 01x01 x1001 | 0101x x1010 01x10 | x1011 01x11 | x1101 011x1 | 110x1 11x01 | 1101x 11x10 | 11x11 | 111x1 | Ø |

\*Таблица 3.1.3 – Поиск простых импликант (C1 \* C1)

В результате образовалось множество *С2* кубов второй размерности:

.

##### 

Множество *Z1* кубов, не участвовавших в образовании новых кубов, пустое.

В таблице 3.1.4 приведён следующий шаг поиска простых импликант – операция *С2\*С2*.

Таблица 3.1.4 – Поиск простых импликант *С2\*С2*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C2\*C2 | 01xx1 | x10x1 | x1x01 | x101x | 01x1x | x1x10 | x1x11 | x11x1 | 11xx1 | 11x1x |
| 01xx1 | - |  |  |  |  |  |  |  |  |  |
| x10x1 | 010x1 | - |  |  |  |  |  |  |  |  |
| x1x01 | 01x01 | x1001 | - |  |  |  |  |  |  |  |
| x101x | 01011 | x1011 | x10y1 | - |  |  |  |  |  |  |
| 01x1x | 01x11 | 01011 | 01xy1 | 0101x | - |  |  |  |  |  |
| x1x10 | 01x1y | x101y | x1xyy | x1010 | 01x10 | - |  |  |  |  |
| x1x11 | 01x11 | x1011 | x1xy1 | x1011 | 01x11 | x1x1y | - |  |  |  |
| x11x1 | 011x1 | x1yx1 | x1101 | x1y11 | 01111 | x111y | x1111 | - |  |  |
| 11xx1 | y1xx1 | 110x1 | 11x01 | 11011 | y1x11 | 11x1y | 11x11 | 111x1 | - |  |
| 11x1x | y1x11 | 11011 | 11xy1 | 1101x | y1x1x | 11x10 | 11x11 | 11111 | 11x11 | - |
| xx11x | 01111 | x1y11 | x11y1 | x1y1x | 0111x | x1110 | x1111 | x1111 | 11111 | 1111x |
| A3 | x1xx1 | x1xx1 | x1xx1 | x1x1x | x1x1x | x1x1x | Ø | Ø | Ø | Ø |

В результате образовалось множество *С3* кубов второй размерности:

.

Множество *Z2* кубов, не участвовавших в образовании новых кубов:

.

В таблице 3.1.5 приведён следующий шаг поиска простых импликант – операция *С3\*С3*.

Таблица 3.1.5 – Поиск простых импликант *С3\*С3*

|  |  |  |
| --- | --- | --- |
| C3\*C3 | x1xx1 | x10xx |
| x1xx1 | - |  |
| x1x1x | x1x11 | - |
| xx11x | x1111 | x111x |
| A4 |  |  |

Новых кубов (четвертой размерности) не образовалось.

Получено множество *=* .

Множество простых импликант:

*Z* =*Z1* ⋃ *Z2* ⋃ *Z3* = *.*

Следующий этап – поиск *L-*экстремалей на множестве простых импликант (таблица 3.1.6). Для этого используется операция # (вычитание).

Таблица 3.1.6 – Поиск *L*-экстремалей

|  |  |  |  |
| --- | --- | --- | --- |
| z#(Z-z) | x1xx1 | x1x1x | xx11x |
| x1xx1 | - | x1x10 | x011x xx110 |
| x1x1x | x1x01 | - | x011x x0110 |
| xx11x | x1x01 | x1010 | - |
| Остаток | x1x01 | x1010 | x011x x0110 |

В таблице 3.1.6 из каждой простой импликанты поочерёдно вычитаются все остальные простые импликанты *z#(Z-z)*.

Получили кубы, “подозрительные” на *L*-экстремальность. Проверяем в таблице 3.1.7.

Таблица 3.1.7 – Проверка на *L-*экстремальность

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| z#(Z-z) n L | 01001 | 01010 | 01011 | 01101 | 11001 | 11010 | 11011 | 11101 |
| x1x01 | 01001 | Ø | Ø | 01101 | 11001 | Ø | Ø | 11101 |
| x1010 | Ø | 01010 | Ø | Ø | Ø | 11010 | Ø | Ø |
| x011x | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| x0110 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

*E* = .

Далее необходимо проанализировать, какие из исходных единичных кубов не покрыты найденной *L*-экстремалью. Анализ осуществляется с помощью таблицы 3.1.8.

.

Таблица 3.1.8 – Поиск непокрытых наборов

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L#E | 01001 | 01010 | 01011 | 01101 | 11001 | 11010 | 11011 | 11101 |
| x1xx1 | Ø | 01010 | Ø | Ø | Ø | 11010 | Ø | Ø |
| x1x1x | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| Остаток | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |

Из таблицы 3.1.8 видим, что найденная *L-*экстремаль покрывает все исходные наборы. Поиск минимального покрытия завершён.

**Минимизация функции :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | 1 | 1 | 1 |  | 1 | 1 |  |  |  |
|  | 1 | 1 | x | x | x | x | 1 | 1 |  |  |
|  |  |  |  | x | x | x | x |  |  |  |
|  |  |  |  |  | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | |  |  |  |  |  |
|  |  |  | *h* | |  | | *h* | |  |  |  |

Рисунок 3.1.2 – Минимизация функции при помощи карты Вейча

Минимизировав функцию при помощи карты Вейча (рисунок 3.1.2), получили *.*

**Минимизация функции :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | 1 |  |  |  |  |  |
|  |  |  | x | x | x | x |  |  |  |  |
|  |  |  |  | x | x | x | x |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | |  |  |  |  |  |
|  |  |  | *h* | |  | | *h* | |  |  |  |

Рисунок 3.1.3 – Минимизация функции при помощи карты Вейча

Минимизировав функцию при помощи карты Вейча (рисунок 3.1.3), получили *.*

**Минимизация функции :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 1 |  |  |  |  |  |  |
|  |  |  | x | x | x | x |  |  |  |  |
|  |  |  |  | x | x | x | x |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | |  |  |  |  |  |
|  |  |  | *h* | |  | | *h* | |  |  |  |

Рисунок 3.1.4 – Минимизация функции при помощи карты Вейча

Минимизировав функцию при помощи карты Вейча (рисунок 3.1.4), получили

*.*

Для проведения оценки эффективности минимизации переключательных функций необходимо посчитать цену схемы до минимизации и цену схемы после минимизации:

.

.

.

.

Функциональная схема ОЧУ приведена в приложении Б.