СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Предмет «Арифметические и логические основы вычислительной техники» является основополагающим в вопросах организации ЭВМ, а, следовательно, и неотъемлемой частью подготовки специалиста в области информационных технологий.

Целью курсового проекта является ознакомление с принципами построения отдельных составных частей ЭВМ и их взаимосвязи, а также получение практических знаний о правильном оформлении .

В данном курсовом проекте выполнен синтез сумматора-умножителя первого типа для алгоритма А в прямых кодах на два разряда одновременно.

Для достижения цели необходимо:

1) Разработать алгоритм умножения и оценить погрешности вычислений.
2) Разработать структурную схему сумматора-умножителя первого типа с блоком сравнения порядков.
3) Разработать функциональные схемы основных узлов сумматора-умножителя в заданных логических базисах.
4) Разработать комбинационную схему на основе мультиплексора.
5) Рассчитать время умножения.

1. РАЗРАБОТКА АЛГОРИТМА УМНОЖЕНИЯ

1. Перевод сомножителей из десятичной системы счисления в четверичную.
Мн = 84,19; Мт = 55,13;

Множимое

_84	4		
84	_21	4	
0	20	_5	4
	1	4	1
		1	

```
  0.19
*   4
 0.76
*   4
 3.04
```

$Мн_4$ = 1110,03.
В соответствии с кодировкой множимого:
$Мн_{2/4}$ =00000011,1110.

Множитель

_55	4		
52	_13	4	
3	12	3	
	1		

```
  0.13
*   4
 0.52
*   4
 2.08
*   4
 0.32
```

$Мт_4$ = 121.113
В соответствии с обычной весомозначной кодировкой множителя:
$Мт_{2/4}$ = 011001.010111.

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

Мн = 0.000000111110 $Р_{мн}$ = 0.0011 $+10_4$ – закодировано по заданию
Мт = 0.110111001000 $Р_{мт}$ = 0.0011 $+03_4$ – закодировано традиционно

3. Умножение двух чисел с плавающей запятой на два разряда множителя одновременно в прямых кодах. Это сводится к сложению порядков, формированию знака произведения, преобразованию разрядов множителя согласно алгоритму и перемножению мантисс сомножителей.

Порядок произведения будет следующим:

$$\begin{aligned} P_{мн} &= \quad 0.0011 \quad +10_4 \\ P_{мт} &= \quad \underline{0.0011} \quad \underline{+03_4} \\ P_{Мн \cdot Мт} &= \; 0.0010 \quad +13_4 \end{aligned}$$

Результат закодирован в соответствии с заданием на кодировку множимого.

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

$$\text{зн Мн} \oplus \text{зн Мт} = 0 \oplus 0 = 0$$

Для умножения мантисс необходимо преобразовать множитель. При умножении чисел в прямых кодах диада $11(3_4)$ преобразуется в триаду $10\bar{1}$. Преобразованный множитель имеет вид $Мт_4^п = 1\bar{1}2\bar{1}020$ или $Мт_2^п = 010\bar{1}10\bar{1}001000$. Перемножение мантисс по алгоритму “А” приведено в таблице 1.1.

Таблица 1.1 - Перемножение мантисс

Четверичная с/с			**Двоично-четверичная с/с**			**Комментарий**
1			**2**			**3**
0.	0000000		0.	11 11 11 11 11 11 11		$\Sigma_0^ч = 0$
<u>0.</u>	<u>0000000</u>		<u>0.</u>	<u>11 11 11 11 11 11 11</u>		$П_1^ч = Мн \cdot 0$
0.	0000000		0.	11 11 11 11 11 11 11		$\Sigma_1^ч$
0.	0000000	0	0.	11 11 11 11 11 11 11	11	$\Sigma_1^ч \cdot 2^{-1}$
<u>0.</u>	<u>0222012</u>	<u>0</u>	<u>0.</u>	<u>11 01 01 01 11 00 01</u>		$П_2^ч = Мн \cdot 2$
0.	0222012	0	0.	11 01 01 01 11 00 01	11	$\Sigma_2^ч$
0.	0022201	20	0.	11 11 01 01 01 11 00	01 11	$\Sigma_2^ч \cdot 2^{-1}$
<u>0.</u>	<u>0000000</u>		<u>0.</u>	<u>11 11 11 11 11 11 11</u>		$П_3^ч = Мн \cdot 0$
0.	0022201	20	0.	11 11 01 01 01 11 00	01 11	$\Sigma_3^ч$
0.	0002220	120	0.	11 11 11 01 01 01 11	00 01 11	$\Sigma_3^ч \cdot 2^{-1}$

Продолжение таблицы 1.1

1			2			3
3.	3222331		1.	10 01 01 01 10 10 00		$П_4^{ч} = Мн \cdot \overline{1}$
3.	3231211	120	1.	10 01 10 00 01 00 00	00 01 11	$\Sigma_4^{ч}$
3.	3323121	1120	1.	10 10 01 10 00 01 00	00 00 01 11	$\Sigma_4^{ч} \cdot 2^{-1}$
0.	0222012		0.	11 01 01 01 11 00 01		$П_5^{ч} = Мн \cdot 2$
0.	0211133	1120	0.	11 01 00 00 00 10 10	00 00 01 11	$\Sigma_5^{ч}$
0.	0021113	31120	0.	11 11 01 00 00 00 10	10 00 00 01 11	$\Sigma_5^{ч} \cdot 2^{-1}$
3.	3222331		1.	10 01 01 01 10 10 00		$П_6^{ч} = Мн \cdot \overline{1}$
3.	3310110	31120	1.	10 10 00 11 00 00 11	10 00 00 01 11	$\Sigma_6^{ч}$
3.	3331011	031120	1.	10 10 10 00 11 00 00	11 10 00 00 01 11	$\Sigma_6^{ч} \cdot 2^{-1}$
0.	0111003		0.	11 00 00 00 11 11 10		$П_7^{ч} = Мн \cdot 1$
0.	0102020	031120	0.	11 00 11 01 11 01 11	11 10 00 00 01 11	$\Sigma_7^{ч}$

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

$Мн \cdot Мт_4 = 1020200.31120 \qquad P_{Мн \cdot Мт} = 0;$
$Мн \cdot Мт_{10} = 4640.8359.$

Результат прямого перемножения операндов дает результат:

$Мн_{10} \cdot Мт_{10} = 84.19 * 55.13 = 4641.3947..$

Абсолютная погрешность

$\Delta = 4641.3947 – 4640.8359 = 0.5588.$

Относительная погрешность

$\delta = \frac{\Delta}{Мн \cdot Мт} = \frac{0.5588}{4641.3947} = 0.00012 \ (\delta = 0.012\ \%).$

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 ПЕРВОГО ТИПА

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

Сразу после начала алгоритма обнуляются все регистры.

Режим работы «Сумма» (Mul/sum – «1»). В начале есть необходимость выравнивания порядков. Для этого в блоке сравнения порядков они вычитаются и, в зависимости от знака суммы и функции сравнения ее с нулем, в регистр множимого поступает мантисса, которая будет сдвигаться. Формируется ее дополнительных код и попадает в аккумулятор. В регистр множимого заносится вторая мантисса.

Далее выполняется выравнивание порядков посредством некоторого количества сдвигов мантиссы в аккумуляторе, равном разности порядков.

По завершению сложения проверяется переполнение. Если переполнение было выявлено, происходит обратный сдвиг содержимого аккумулятора, а порядок результата уменьшается на единицу.

При необходимости выравнивания порядков в регистре-аккумуляторе может выполняться сдвиг мантиссы первого слагаемого.

Если устройство работает как *умножитель* (Mul/sum – «0»), то в порядок результата заносится сумма порядков множимого и множителя в дополнительном коде. Параллельно с этим их мантиссы заносятся в соответствующие регистры.

Далее пошагово выполняется сама операция умножения: сдвиг мантиссы и преобразование разрядов множителя, формирования дополнительного кода частичного произведения и занесение его в регистр аккумулятор, после разряды последнего сдвигаются, а регистр множителя служит его «продолжением». Операция продолжается, пока разряды множителя не иссякнут.

3. РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ СХЕМ ОСНОВНЫХ УЗЛОВ СУММАТОРА – УМНОЖИТЕЛЯ

3.1 Логический синтез одноразрядного четверичного сумматора

ОЧС – это комбинационное устройство с 5 двоичными входами (2 разряда из одного слагаемого, 2 разряда другого и вход переноса) и 3 двоичными выходами.

Принцип работы ОЧС представлен с помощью таблицы истинности (таблица 1.4).

Разряды обоих слагаемых закодированы: 0-11; 1-00; 2-01; 3-10. В таблице 3.n выделено 16 безразличных наборов, т.к. на входы ОЧС со старших выходов ОЧУ не могут прийти «2» и «3».

Таблица 3.1 – 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	1	1+1+0=02
0	0	0	0	1	0	1	0	1+1+1=03
0	0	0	1	0	x	x	x	1+2+0=03
0	0	0	1	1	x	x	x	1+2+1=10
0	0	1	0	0	x	x	x	1+3+0=10
0	0	1	0	1	x	x	x	1+3+1=11
0	0	1	1	0	0	0	0	1+0+0=01
0	0	1	1	1	0	0	1	1+0+1=02
0	1	0	0	0	0	1	0	2+1+0=03
0	1	0	0	1	1	1	1	2+1+1=10
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	0	1	2+0+0=02
0	1	1	1	1	0	1	0	2+0+1=03
1	0	0	0	0	1	1	1	3+1+0=10
1	0	0	0	1	1	0	0	3+1+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	0	1	0	3+0+0=03
1	0	1	1	1	1	1	1	3+0+1=10
1	1	0	0	0	0	0	0	0+1+0=01

1	1	0	0	1	0	0	1	0+1+1=02
1	1	0	1	0	x	x	x	0+2+0=02
1	1	0	1	1	x	x	x	0+2+1=03
1	1	1	0	0	x	x	x	0+3+0=03
1	1	1	0	1	x	x	x	0+3+1=10
2	1	1	1	0	0	1	1	0+0+0=00
1	1	1	1	1	0	0	0	0+0+1=01

Минимизацию переключательных функций(ПФ) проведём с помощью карт Вейча. На картах символом «x» отмечены наборы, на которых функция может принимать произвольное значение (безразличные наборы).

Для функции S_1 заполненная карта приведена на рисунке 3.1.

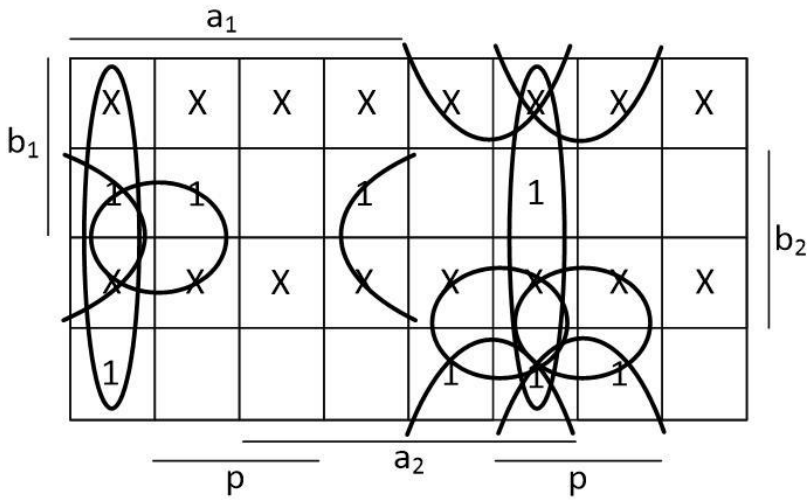

Рисунок 3.1 – Минимизация функции S_1 при помощи карты Вейча

Исходя из результатов минимизации S_1 могут быть получены следующие тупиковые формы:

$$F_{1\,min} = a_1\overline{a_2}b_2 + \overline{a_1}a_2\overline{b_2} + a_1b_2\overline{p} + \overline{a_1}\overline{b_2}p + \overline{a_1}a_2p + a_1\overline{a_2}\overline{p},$$
$$F_{2\,min} = a_1\overline{a_2}b_2 + \overline{a_1}a_2\overline{b_1} + a_1b_2\overline{p} + \overline{a_1}\overline{b_1}p + \overline{a_1}a_2p + a_1\overline{a_2}\overline{p},$$
$$F_{3\,min} = a_1\overline{a_2}b_2 + \overline{a_1}a_2\overline{b_1} + a_1b_2\overline{p} + \overline{a_1}\overline{b_2}p + \overline{a_1}a_2p + a_1\overline{a_2}\overline{p},$$
$$F_{4\,min} = a_1\overline{a_2}b_2 + \overline{a_1}a_2\overline{b_2} + a_1b_2\overline{p} + \overline{a_1}\overline{b_1}p + \overline{a_1}a_2p + a_1\overline{a_2}\overline{p}.$$

Для реализации выхода S_1 на функциональной схеме будем использовать тупиковую форму $F_{1\,min}$. Для этого приведем ее к базису, заданному по условию.

$$S_1 = a_1\overline{a_2}b_2 + \overline{a_1}a_2\overline{b_2} + a_1b_2\overline{p} + \overline{a_1}\overline{b_2}p + \overline{a_1}a_2p + a_1\overline{a_2}\overline{p} =$$

$$= \overline{a_1 + \overline{a_2} + b_2} + \overline{\overline{a_1} + a_2 + \overline{b_2}} + \overline{a_1 + b_2 + \overline{p}} + \overline{\overline{a_1} + \overline{b_2} + p} + \\ + \overline{\overline{a_1} + a_2 + p} + \overline{a_1 + \overline{a_2} + \overline{p}}.$$

Эффективность минимизации для всех переключательных функций будем оценивать, как отношение цен по Квайну до и после минимизации.

Тогда эффективность минимизации переключательной функции S_1:

$$К_{s_1} = \frac{68}{39} = 1{,}74$$

Функциональная схема выхода S_1 представлена в приложении Б.

Для функции S_2 заполненная карта приведена на рисунке 3.2.

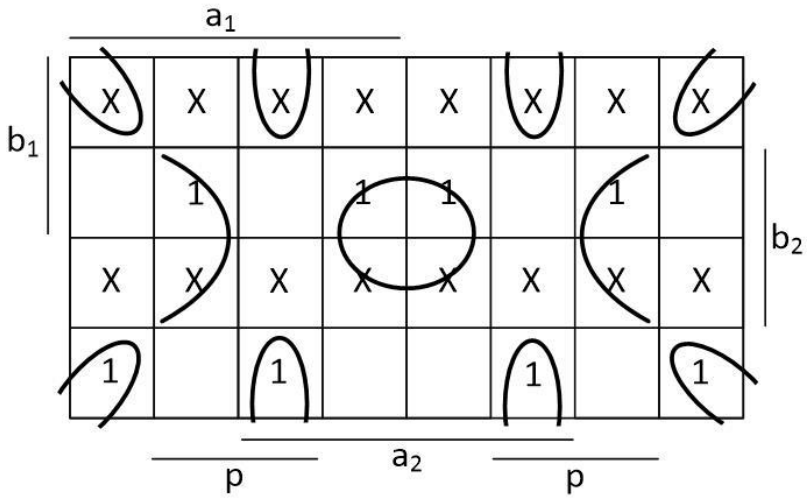

Рисунок 3.2 – Минимизация функции S_2 при помощи карты Вейча

Следовательно:

$$S_2 = \overline{a_2}\overline{b_2}\overline{p} + a_2 b_2 \overline{p} + a_2 \overline{b_2} p + \overline{a_2} b_2 p = p(a_2 \oplus b_2) + \overline{p}\left(\overline{a_2 \oplus b_2}\right) \\ = \overline{p \oplus a_2 \oplus b_2}.$$

Эффективность минимизации переключательной функции S_2:

$$К_{s_2} = \frac{68}{4} = 17$$

Функциональная схема выхода S_2 представлена в приложении Б.

Для функции П заполненная карта приведена на рисунке 3.3.

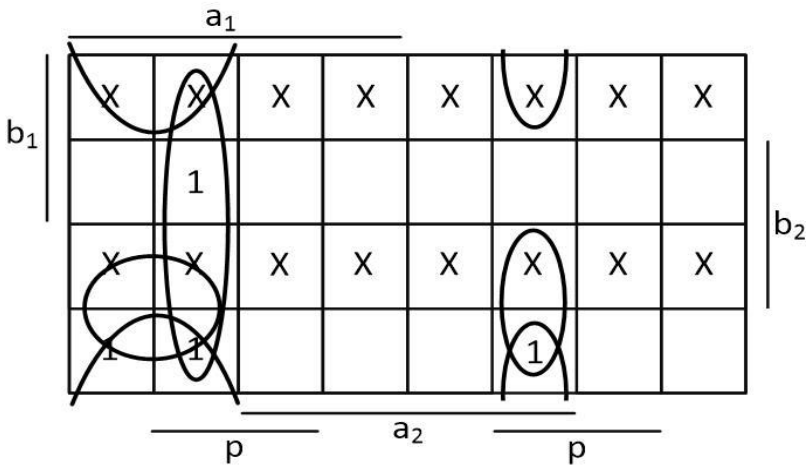

Рисунок 3.3 – Минимизация функции П при помощи карты Вейча

Исходя из результатов минимизации П могут быть получены следующие тупиковые формы:

$$F_{1\,min} = a_1\overline{a_2}\overline{b_1} + a_1\overline{a_2}p + \overline{a_1}a_2\overline{b_1}p,$$
$$F_{2\,min} = a_1\overline{a_2}\overline{b_1} + a_1\overline{a_2}p + \overline{a_1}a_2\overline{b_2}p,$$
$$F_{3\,min} = a_1\overline{a_2}\overline{b_2} + a_1\overline{a_2}p + \overline{a_1}a_2\overline{b_1}p,$$
$$F_{4\,min} = a_1\overline{a_2}\overline{b_2} + a_1\overline{a_2}p + \overline{a_1}a_2\overline{b_2}p.$$

Для реализации выхода П на функциональной схеме будем использовать тупиковую форму $F_{1\,min}$. Для этого приведем ее к базису, заданному по условию.

$$\text{П} = a_1\overline{a_2}\overline{b_1} + a_1\overline{a_2}p + \overline{a_1}a_2\overline{b_1}p = \overline{\overline{a_1} + a_2 + b_1} + \overline{\overline{a_1} + a_2 + \overline{p}} + \overline{a_1 + \overline{a_2} + b_1 + \overline{p}}.$$

Эффективность минимизации переключательной функции П:

$$\text{К}_\text{П} = \frac{35}{21} = 1{,}67$$

Функциональная схема выхода П представлена в приложении Б.

3.2 Логический синтез одноразрядного четверичного умножителя

ОЧУ – это комбинационное устройство с 5 двоичными входами (2 разряда из регистра Мн, два разряда из регистра Мт и управляющий вход h) и 4 двоичными выходами.

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

Разряды множимого закодированы: 0-11; 1-00; 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 3.2 выделено 8 безразличных наборов, т.к. на входы ОЧУ из разрядов множителя не может прийти код «11».

Таблица 3.2 – Таблица истинности ОЧУ

X_1	X_2	Y_1	Y_2	h	P_1	P_2	P_3	P_4	**Пример операции в четверичной с/с**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0	0	0	0	1	1	1	1	1*0 = 00
0	0	0	0	1	1	1	0	0	Выход - код «01»
0	0	0	1	0	1	1	0	0	1*1= 01
0	0	0	1	1	1	1	0	0	Выход - код «01»
0	0	1	0	0	1	1	0	1	1*2 = 02
0	0	1	0	1	1	1	0	0	Выход - код «01»
0	0	1	1	0	x	x	x	x	1*3 = 03
0	0	1	1	1	x	x	x	x	Выход - код «01»
0	1	0	0	0	1	1	1	1	2*0 = 00
0	1	0	0	1	1	1	0	1	Выход - код «02»
0	1	0	1	0	1	1	0	1	2*1 = 02
0	1	0	1	1	1	1	0	1	Выход - код «02»
0	1	1	0	0	1	1	0	1	2*2 = 10
0	1	1	0	1	0	0	1	1	Выход - код «02»
0	1	1	1	0	1	1	0	1	2*3 = 12
0	1	1	1	1	x	x	x	x	Выход - код «02»
1	0	0	0	0	x	x	x	x	3*0 = 00
1	0	0	0	1	1	1	1	1	Выход - код «03»
1	0	0	1	0	1	1	1	0	3*1 = 03
1	0	0	1	1	1	1	1	0	Выход - код «03»
1	0	1	0	0	1	1	1	0	3*2 = 12
1	0	1	0	1	0	0	0	1	Выход - код «03»
1	0	1	1	0	1	1	1	0	3*3 = 21
1	0	1	1	1	x	x	x	x	Выход - код «03»

1	1	0	0	0	x	x	x	x	0*0 = 00
1	1	0	0	1	1	1	1	1	Выход - код «00»
1	1	0	1	0	1	1	1	1	0*1 = 00
1	1	0	1	1	1	1	1	1	Выход - код «00»
1	1	1	0	0	1	1	1	1	0*2 = 00
1	1	1	0	1	1	1	1	1	Выход - код «00»
1	1	1	1	0	x	x	x	x	0*3 = 00
1	1	1	1	1	x	x	x	x	Выход - код «00»

Минимизацию переключательной функции P_3 проведем при помощи алгоритма Квайна - Мак-Класки. Для начала определим множества единичных единичных кубов (L) и множество безразличных наборов (N).

$$L = \left\{ \begin{array}{llll} 00000, & 00101, & 01100, & 10100, \\ 00001, & 01000, & 01110, & 10110, \\ 00010, & 01001, & 10001, & 11001, \\ 00011, & 01010, & 10010, & 11010, \\ 00100, & 01011, & 10011, & 11011, \end{array} \begin{array}{l} 11100, \\ 11101 \end{array} \right\}. \qquad N = \left\{ \begin{array}{l} 10100, 10100, \\ 10110, 10110, \\ 11001, 11001, \\ 11010, 11010 \end{array} \right\}.$$

Выполним разбиение множеств на группы, содержащие одинаковое количество «1».

$$K_0^0 = \{ 00000 \}. \qquad K_1^0 = \left\{ \begin{array}{l} 00001 \\ 00010 \\ 00100 \\ 01000 \\ 10000 \end{array} \right\}. \qquad K_2^0 = \left\{ \begin{array}{l} 01011, 11001, \\ 01110, 11010, \\ 10011, 11100, \\ 10110, 00111 \end{array} \right\}.$$

$$K_3^0 = \left\{ \begin{array}{l} 00011, 10001, \\ 00101, 10010, \\ 01001, 10100, \\ 01010, 00110, \\ 01100, 11000 \end{array} \right\} \qquad K_4^0 = \left\{ \begin{array}{l} 11011 \\ 11101 \\ 10111 \\ 11110 \end{array} \right\} \qquad K_5^0 = \{ 11111 \}.$$

Теперь попарно склеим соседние (K_i и K_{i+1}) множества.

$$K_1^1 = \left\{ \begin{matrix} \text{x0000, x1000, x1100,} \\ \text{x0001, x0011, x0110,} \\ \text{x0010, x1001, x1011,} \\ \text{x0100, x1010, x1110,} \end{matrix} \text{x0111} \right\}. \quad K_2^1 = \left\{ \begin{matrix} \text{0x000, 1x000, 1x100 ,} \\ \text{0x001, 0x011, 0x110,} \\ \text{0x010, 1x001, 1x011,} \\ \text{0x100, 1x010, 1x110,} \end{matrix} \text{1x111} \right\}.$$

$$K_3^1 = \left\{ \begin{matrix} \text{00x00 ,10x00 ,11x00 ,} \\ \text{00x01 ,00x11, 10x11 ,} \\ \text{00x10 ,01x10 ,11x01 ,} \\ \text{01x00 ,10x10, 11x10,} \end{matrix} \text{11x11} \right\}. \quad K_4^1 = \left\{ \begin{matrix} \text{000x0, 100x0, 100x1,} \\ \text{000x1, 001x1, 101x0, 111x0 ,} \\ \text{001x0, 010x1, 110x0, 111x1} \\ \text{010x0, 011x0, 110x1,} \end{matrix} \right\}.$$

$$K_5^1 = \left\{ \begin{matrix} \text{0000x, 1000x, 1100x,} \\ \text{0001x, 0101x, 1011x,} \\ \text{0010x, 1001x, 1101x,} \\ \text{0100x, 0011x, 1110x,} \end{matrix} \text{1111x} \right\}.$$

На первом этапе алгоритма простых импликант не выявлено. Теперь проведем склеивание внутри полученных множеств.

$$K_1^2 = \left\{ \begin{matrix} \text{x000x} \\ \text{x001x} \\ \text{x100x} \\ \text{x101x} \\ \text{x011x} \end{matrix} \right\}. \quad K_2^2 = \left\{ \begin{matrix} \text{x00x0} \\ \text{x01x0} \\ \text{x10x0} \\ \text{x10x1} \\ \text{x11x0} \end{matrix} \right\}. \quad K_3^2 = \left\{ \begin{matrix} \text{x0x10} \\ \text{x1x00} \\ \text{x0x11} \\ \text{x1x10} \end{matrix} \right\}. \quad K_4^2 = \left\{ \begin{matrix} \text{xx001} \\ \text{xx010} \\ \text{xx100} \\ \text{xx011} \\ \text{xx110} \end{matrix} \right\}.$$

$$K_5^2 = \left\{ \begin{matrix} \text{0x00x} \\ \text{0x01x} \\ \text{1x00x} \\ \text{1x01x} \\ \text{1x11x} \end{matrix} \right\}. \quad K_6^2 = \left\{ \begin{matrix} \text{000xx} \\ \text{001xx} \\ \text{010xx} \\ \text{100xx} \\ \text{110xx} \\ \text{111xx} \end{matrix} \right\}. \quad K_7^2 = \left\{ \begin{matrix} \text{0x0x0} \\ \text{0x0x1} \\ \text{0x1x0} \\ \text{1x0x0} \\ \text{1x1x0} \end{matrix} \right\}. \quad K_8^2 = \left\{ \begin{matrix} \text{0x00x} \\ \text{0x01x} \\ \text{1x00x} \\ \text{1x01x} \\ \text{1x11x} \end{matrix} \right\}.$$

На втором этапе простых импликант также не выявлено. Проводим склеивание внутри полученных множеств.

$$K_1^3 = \left\{ \begin{matrix} xx00x \\ xx01x \end{matrix} \right\}. \qquad K_2^3 = \left\{ \begin{matrix} x0xx0 \\ x1xx0 \end{matrix} \right\}. \qquad K_3^3 = \left\{ \begin{matrix} xx1x0 \\ xx0x0 \\ xx1x0 \end{matrix} \right\}. \qquad K_4^3 = \left\{ \begin{matrix} xxx00 \\ xxx10 \end{matrix} \right\}.$$

$$K_5^3 = \left\{ \begin{matrix} xx0x1 \\ xx0x0 \end{matrix} \right\}. \qquad K_6^3 = \left\{ x0x1x \right\}. \qquad K_7^3 = \left\{ \begin{matrix} 0xxx0 \\ 1xxx0 \end{matrix} \right\}. \qquad K_8^3 = \left\{ \begin{matrix} 0x0xx \\ 1x0xx \end{matrix} \right\}.$$

$$K_9^3 = \left\{ 1xx1x \right\}. \qquad K_{10}^3 = \left\{ 00xxx \right\}. \qquad K_{11}^3 = \left\{ 11xxx \right\}.$$

Выявлены следующие простые импликанты: 1xx1x, x0x1x, 00xxx, 11xxx. Продолжим склеивание в полученных множествах.

K_1^4 = { xxxx0 }. K_2^4 = { xx0xx }.

Таким образом, множество простых импликант имеет вид : 1xx1x, x0x1x, 00xxx, 11xxx, xxxx0, xx0xx.

Теперь построим импликантную таблицу (таблица 3.3) и найдем минимальное покрытие.

Таблица 3.3 – Поиск минимального покрытия

Простые импликанты	Минтермы																					
	00000	00001	00010	00011	000100	00101	01000	01001	01010	01011	01100	01110	10001	10010	10011	10100	10110	11001	11010	11011	11100	11101
xxxx0	*		*		*		*		*		*	*		*		*	*		*		*	
xx0xx	*	*	*	*			*	*	*	*			*	*	*							
11xxx																		*	*	*	*	*
00xxx	*	*	*	*	*	*																
1xx1x														*	*		*		*	*		
x0x1x			*	*										*	*		*					

Следовательно:

$$P_1 = \overline{y_1} + \bar{h} + \overline{(x_1 \oplus x_2)} = \overline{y_1 h (x_1 \oplus x_{2)}}$$

Эффективность минимизации переключательной функции P_1:

$$К_{P_1} = \frac{214}{5} = 42{,}8$$

Т.к. таблица истинности для выхода P_2 полностью повторяет выход P_1, то он будет иметь такую же минимальную функцию. Функциональная схема выходов P_1 и P_2 представлена в приложении В.

Минимизацию переключательной функции P_3 проведем при помощи алгоритма извлечения (метода Рота). Для этого определим множество единичных кубов (L) и множество безразличных наборов (N)

$$L = \left\{ \begin{array}{ll} 00000, & 10101, \\ 01000, & 11000, \\ 01100, & 11001, \\ 10000, & 11010, \\ 10001, & 11011, \\ 10010, & 11100, \\ 10011, & 11101 \end{array} \right\}. \qquad N = \left\{ \begin{array}{ll} 00110, & 10110, \\ 00111, & 10111, \\ 01110, & 11110, \\ 01111, & 11111 \end{array} \right\}.$$

Предварительное склеивание безразличных наборов (N) для упрощения алгоритма минимизации приведено на рисунке 3.4.

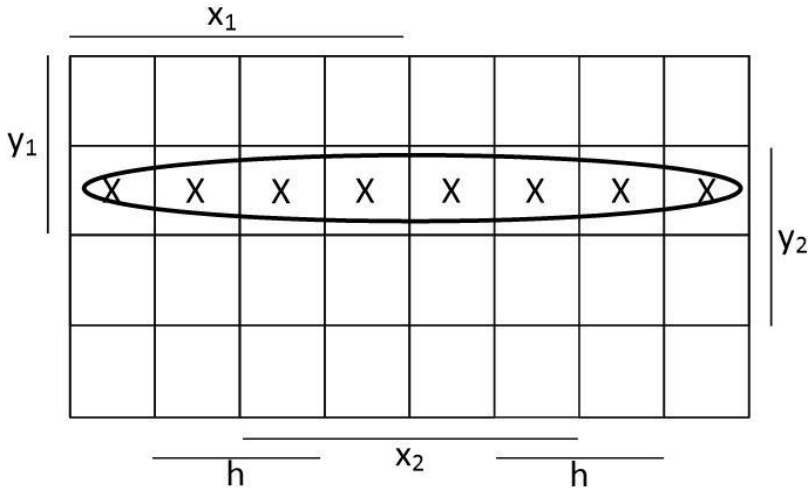

Рисунок 3.4 – Склеивание множества N при помощи карты Вейча

Следовательно:

$N = \{$ xx11x $\}$.

Сформируем множество $C_0 = L \cup N$:

$$C_0 = \left\{ \begin{array}{l} 00000,\ 10000,\ 10011,\ 11001,\ 11101, \\ 01000,\ 10001,\ 10101,\ 11010,\ 11101, \\ 01100,\ 10010,\ 11000,\ 11011,\ \text{xx11x} \end{array} \right\}.$$

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 3.4.

Таблица 3.4 – Поиск простых импликант ($C_0 * C_0$)

C0*C0	00000	01000	01100	10000	10001	10010	10011	10101	11000	11001	11010	11011	11100	11101	xx11x
00000	-														
01000	0y000	-													
01100		01y00	-												
10000	y0000			-											
10001				1000y	-										
10010				100y0		-									
10011					100y1	1001y	-								
10101					10y01			-							
11000		y1000		1y000					-						
11001					1y001				1100y	-					
11010						1y010			110y0		-				
11011							1y011			110y1	1101y	-			
11100			y1100						11y00				-		
11101								1y101		11y01			1110y	-	
xx11x			011y0			10y10	10y11	101y1			11y10	11y11	111y0	111y1	-

В результате этой операции сформируется новое множество кубов C_1:

$$C_1 = \left\{ \begin{array}{lllll} 0x000, & 1000x, & 1001x, & 101x1, & 1101x, \\ x0000, & 100x0, & 1x010, & 1100x, & 11x10, \\ 01x00, & 1x000, & 10x10, & 110x0, & 11x11, \\ x1000, & 100x1, & 1x011, & 11x00, & 1110x, \\ x1100, & 10x01, & 10x11, & 110x1, & 111x0, \\ 011x0, & 1x001, & 1x101, & 11x01, & 111x1, \end{array} \; xx11x \right\}.$$

Множество Z_0 кубов, не участвовавших в образовании новых кубов, пустое.

В приложении Д приведён следующий шаг поиска простых импликант с помощью операции $C_1 * C_1$.

В результате образовалось множество C_2 кубов второй размерности:

$$C_2 = \left\{ \begin{array}{llll} xx000, & 1x0x0, & 10x1x, & 11x0x, \\ x1x00, & 10xx1, & 1xx10, & 11xx0, \\ x11x0, & 1x0x1, & 1xx11, & 11xx1, \\ 100xx, & 1xx01, & 1x1x1, & 11x1x, \\ 1x00x, & 1x01x, & 110xx, & 111xx, \end{array} \; xx11x \right\}.$$

Множество Z_1 кубов, не участвовавших в образовании новых кубов, пустое.

В таблице 3.5 приведен следующий шаг поиска простых импликант – операция $C_2 * C_2$.

В результате этой операции сформируется новое множество кубов C_3:

C_3 = { 1x0xx, 1xxx, 1xx1x, 11xxx, xx11x }.

Множество Z_2 кубов, не участвовавших в образовании новых кубов, имеет вид:

Z_2 = { xx000, x1x00, x11x0 }.

В таблице 3.6 приведен следующий шаг поиска простых импликант – операция $C_3 * C_3$.

Таблица 3.5 – Поиск простых импликант ($C_2 * C_2$)

C2*C2	xx000	x1x00	x11x0	100xx	1x00x	1x0x0	10xx1	1x0x1	1xx01	1x01x	10x1x	1xx10	1xx11	1x1x1	110xx	11x0x	11xx0	11xx1	11x1x	111xx
xx000	-																			
x1x00		-																		
x11x0			-																	
100xx				-																
1x00x					-															
1x0x0						-														
10xx1							-													
1x0x1						1x0xy		-												
1xx01									-											
1x01x					1x0yx					-										
10x1x											-									
1xx10												-								
1xx11									1xxy1			1xx1y	-							
1x1x1								1xyx1						-						
110xx				1y0xx											-					
11x0x																-				
11xx0																	-			
11xx1							1yxx1										11xxy	-		
11x1x											1yx1x					11xyx			-	
111xx															11yxx					-
xx11x										1xy1x										

Таблица 3.6 – Поиск простых импликант (C_3 * C_3)

C3*C3	1x0xx	1xxx1	1xx1x	11xxx
1x0xx	-			
1xxx1		-		
1xx1x			-	
11xxx				-
xx11x				

Новых кубов (четвёртой размерности) не образовалось.

Множество Z_3 кубов, не участвовавших в образовании новых кубов, имеет вид:

Z_3 = { 1x0xx, 1xxx1, 1xx1x, 11xxx, xx11x }.

Итоговое множество простых импликант имеет вид:

$Z = Z_2 \cup Z_3$ = { xx000, x1x00, x11x0, 1x0xx, 1xxx1, 1xx1x, 11xxx, xx11x }.

Следующий этап – поиск L-экстремалей на множестве простых импликант (таблица 3.7). Для этого используется операция # (решётчатое вычитание).

Таблица 3.7 – Поиск L-экстремалей

z#(Z-z)	xx000	x1x00	x11x0	1x0xx	1xxx1	1xx1x	11xxx	xx11x
xx000	-	x1100	x11x0	1x01x 1x0x1	1xxx1	1xx1x	111xx 11x1x 11xx1	xx11x
x1x00	x0000	-	x1110	1x01x 1x0x1	1xxx1	1xx1x	1111x 111x1 11x1x 11xx1	xx11x
x11x0	x0000	Ø	-	1x01x 1x0x1	1xxx1	10x1x 1x01x 1xx11	11111 111x1 1101x 11x11 11xx1	x011x xx111
1x0xx	00000	Ø	x1110	-	1x1x1	1011x 1x111	11111 111x1 11111 111x1	x011x xx111

1xxx1	00000	Ø	x1110	1x010	-	10110	Ø	0011x x0110 0x111
1xx1x	00000	Ø	01110	Ø	1x101	-	Ø	0011x 00110 0x111
11xxx	00000	Ø	01110	Ø	10101	10110	-	0011x 00110 0x111
xx11x	00000	Ø	Ø	Ø	10101	Ø	Ø	-

В результате операции образовано множество (*E*) L-экстремалей:

E = { xx000, 1xxx1, 11xxx }.

Теперь необходимо проверить, нет ли среди L-экстремалей таких, что стали L-экстремалями за счет безразличных наборов (таблица 3.8).

Таблица 3.8 – Проверка L-экстремалей

z#(Z-z) ∩ L	00000	01000	01100	10000	10001	10010	10011	10101	11000	11001	11010	11011	11100	11101
00000	00000	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
10101	Ø	Ø	Ø	Ø	Ø	Ø	Ø	10101	Ø	Ø	Ø	Ø	Ø	Ø
0011x	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
00110	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
0x111	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

По результатам таблицы 3.8 L-экстремалями, не связанными с безразличными наборами, стали кубы xx000 и 1xxx1 (остаток от вычитания из них всех остальных простых импликант – 00000 и 10101 соответственно – относятся к множеству единичных наборов L исходного задания функции). Эти кубы обязательно должны войти в минимальное покрытие.

Теперь проанализируем, есть ли среди исходных единичных кубов множества (*L*) те, что не покрыты L-экстремалями (таблица 3.9).

Таблица 3.9 – Проверка L-экстремалей

L#E	00000	01000	01100	10000	10001	10010	10011	10101	11000	11001	11010	11011	11100	11101
xx000	Ø	Ø	01100	Ø	10001	10010	10011	10101	Ø	11001	11010	11011	11100	11101
1xxx1	Ø	Ø	01100	Ø	Ø	10010	Ø	Ø	Ø	Ø	11010	Ø	11100	Ø

Множество кубов, не покрытых L-экстремалями, имеет вид:

L' = { 01100, 10010, 11010, 11100 }.

Чтобы их покрыть, воспользуемся множеством простых импликант, не являющихся L-экстремалями (таблица 3.10).

Таблица 3.10 – Покрытие оставшихся кубов

L'∩Ž	01100	10010	11010	11100
x1x00	01100	Ø	Ø	11100
x11x0	01100	Ø	Ø	11100
1x0xx	Ø	10010	11010	Ø
1xx1x	Ø	10010	11010	Ø
11xxx	Ø	Ø	11010	11100
xx11x	Ø	Ø	Ø	Ø

Исходя из результатов минимизации P_3 могут быть получены следующие тупиковые формы:

$$F_{1\,min} = x_2\overline{y_2}\bar{h} + x_1\overline{y_2} + \overline{y_1}\,\overline{y_2}\bar{h} + x_1h,$$
$$F_{2\,min} = x_2y_1\bar{h} + x_1\overline{y_2} + \overline{y_1}\,\overline{y_2}\bar{h} + x_1h,$$
$$F_{3\,min} = x_2\overline{y_2}\bar{h} + x_1y_2 + \overline{y_1}\,\overline{y_2}\bar{h} + x_1h,$$
$$F_{4\,min} = x_2y_1\bar{h} + x_1y_2 + \overline{y_1}\,\overline{y_2}\bar{h} + x_1h.$$

Для реализации выхода P_3 на функциональной схеме будем использовать тупиковую форму $F_{3\,min}$. Для этого приведем ее к базису, заданному по условию.

$$F_{3\,min} = x_2\overline{y_2}\bar{h} + x_1y_2 + \overline{y_1y_2}\bar{h} + x_1h = \overline{y_2}\bar{h}(\overline{y_1} + x_2) +$$
$$+ x_1(y_2 + h) = \overline{y_2}\bar{h}\overline{y_1\overline{x_2}} + x_1\overline{\overline{y_2}\bar{h}} = \overline{\overline{\overline{y_2}\bar{h}\overline{y_1\overline{x_2}}}\;\overline{x_1\overline{y_2}\bar{h}}},$$

Эффективность минимизации переключательной функции P_3:

$$К_{P_3} = \frac{110}{19} = 5{,}79$$

Функциональная схема выхода P_3 представлена в приложении В.

Минимизацию переключательной функции P_4 проведем при помощи карты Вейча

Для функции P_4 заполненная карта приведена на рисунке 3.5.

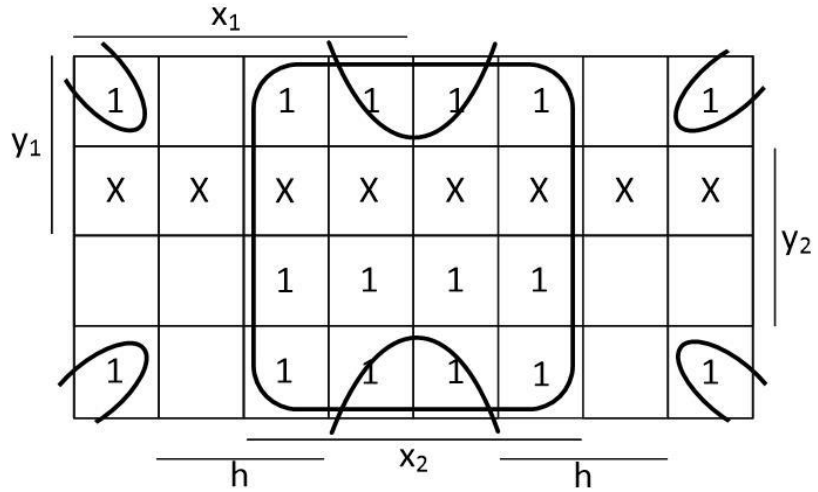

Рисунок 3.5 – Минимизация функции P_4 при помощи карты Вейча

Следовательно:

$$P_4 = x_2 + \overline{y_2}\overline{h} = \overline{\overline{x_2}\overline{\overline{y_2}}\overline{h}}$$

Эффективность минимизации переключательной функции P_4:

$$К_{P_4} = \frac{137}{8} = 17{,}125$$

Функциональная схема выхода P_4 представлена в приложении В.

4. СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ НА ОСНОВЕ МУЛЬТИПЛЕКСОРА

Мультиплексор – это логическая схема, имеющая n информационных входов, m управляющих входов и один выход. При этом должно выполняться условие $n = 2^m$.

Принцип работы мультиплексора состоит в следующем:

На выход мультиплексора может быть пропущен без изменений любой (один) логический сигнал, поступающий на один из информационных входов.

Порядковый номер информационного входа, значение которого в данный момент должно быть передано на выход, определяется двоичным кодом, поданным на управляющие входы.

Переключательные функции пяти переменных можно просто реализовать на мультиплексоре «один из восьми». Такой мультиплексор имеет три управляющих сигналов и, следовательно, два в степени три, т.е. восемь информационных. Реализация ОЧС потребует три ПФ.

Синтез логических схем для ПФ ОЧС приведён в таблице 4.1.

Таблица 4.1 – Таблица истинности для синтеза ПФ

N	a_1	a_2	b_1	b_2	p	**П**	**Выход**	S_1	**Выход**	S_2	**Выход**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	0	0	0	0	0	«0»	0	p	1	$\bar{p}$
0	0	0	0	0	1	0		1		0	
0	0	0	0	1	0	x		x		x	
0	0	0	0	1	1	x		x		x	
1	0	0	1	0	0	x	«0»	x	«0»	x	p
1	0	0	1	0	1	x		x		x	
1	0	0	1	1	0	0		0		0	
1	0	0	1	1	1	0		0		1	
2	0	1	0	0	0	0	p	1	«1»	0	p
2	0	1	0	0	1	1		1		1	
2	0	1	0	1	0	x		x		x	
2	0	1	0	1	1	x		x		x	
3	0	1	1	0	0	x	«0»	x	p	x	$\bar{p}$
3	0	1	1	0	1	x		x		x	
3	0	1	1	1	0	0		0		1	
3	0	1	1	1	1	0		1		0	

4	1	0	0	0	0	1	«1»	1	$\bar{p}$	1	$\bar{p}$
4	1	0	0	0	1	1		0		0	
4	1	0	0	1	0	x		x		x	
4	1	0	0	1	1	x		x		x	
5	1	0	1	0	0	x	p	x	«1»	x	p
5	1	0	1	0	1	x		x		x	
5	1	0	1	1	0	0		1		0	
5	1	0	1	1	1	1		1		1	
6	1	1	0	0	0	0	«0»	0	«0»	0	p
6	1	1	0	0	1	0		0		1	
6	1	1	0	1	0	x		x		x	
6	1	1	0	1	1	x		x		x	
7	1	1	1	0	0	x	«0»	x	$\bar{p}$	x	$\bar{p}$
7	1	1	1	0	1	x		x		x	
7	1	1	1	1	0	0		1		1	
7	1	1	1	1	1	0		0		0	

Схема ОЧС на основе мультиплексоров представлена в приложении Г.

5. ОЦЕНКА РЕЗУЛЬАТОВ РАЗРАБОТКИ

Время умножения на n разрядов равно:

$$t_n = n * t_1,$$

где t_1 – время умножения на один разряд.

Время умножения на один разряд состоит из: времени сдвига ($t_{\text{сдв}}$), времени работы преобразователя множителя ($t_{\text{пм}}$), времени формирования дополнительного кода ($t_{\text{ФДК}}$), времени работы ОЧУ ($t_{\text{0чу}}$), времени работы цепочки из n ОЧС ($t_{\text{0чс}}$) и времени срабатывания аккумулятора ($t_{\text{ак}}$). Время работы ОЧУ и ОЧС равны, соответственно,

$$t_{\text{0чу}} = 6 * \tau_{\text{э}}, \qquad t_{\text{0чс}} = 4 * \tau_{\text{э}}$$

где $\tau_{\text{э}}$ – время работы одного логического элемента.

Таким образом

$$t_1 = t_{\text{сдв}} + t_{\text{пм}} + t_{\text{ФДК}} + t_{\text{0чу}} + n * t_{\text{0чс}} + t_{\text{ак}} = t_{\text{сдв}} + t_{\text{пм}} + t_{\text{ФДК}} + +6 * \tau_{\text{э}} + \ n * 4 * \tau_{\text{э}} + t_{\text{ак}}.$$

$$t_n = n * t_1 = n * (t_{\text{пм}} + t_{\text{ФДК}} + 6 * \tau_{\text{э}} + \ n * 4 * \tau_{\text{э}} + t_{\text{ак}} + t_{\text{сдв}}) = n * (t_{\text{пм}} + t_{\text{ФДК}} + \tau_{\text{э}} * (6 + \ 4 * n) + t_{\text{ак}} + t_{\text{сдв}}).$$

ЗАКЛЮЧЕНИЕ

Цель данного курсового проекта была достигнута. Все знания, полученные по дисциплине «Арифметические и логические основы вычислительной техники», были успешно применены на практике в ходе курсового проектирования. В частности, в ходе построения сумматора-умножителя.

Была произведена минимизация булевых функций различными способами (картами Карно (Вейча) и алгоритмом извлечения (Рота)). В качестве главного достоинства карт Карно (Вейча) можно выделить простоту и минимальные затраты времени. Однако применение данного способа для функций многих переменных будет затруднительно. Алгоритмом извлечения же можно минимизировать функции от любого числа переменных. Данный алгоритм является машинно-ориентированным и легко программируемым, но как минус можно отметить его сложность и громоздкость при ручном исполнении.

При построении функциональных схем были использованы различные логические базисы, что позволило отточить навыки булевой алгебры, а частности, использование правила де Моргана.

Использование такого устройства как мультиплексор для построения комбинационной схемы значительно упростило ее и позволило свести к минимуму использование логических элементов в схеме.

Для определения частоты тактового импульса было рассчитано время умножения на один разряд.

Таким образом, все задачи выполнены. Но на этом разработка данного устройства не завершена. В будущем необходимо предусмотреть округление результата сложения двух чисел и спроектировать устройство управления.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. 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.

2. Луцик, Ю. А. Арифметические и логические основы вычислительной техники : метод. пособие / Ю. А. Луцик, И. В. Лукьянова. – Минск : МРТИ, 2004. – 35 с.

3. Искра, Н. А. И86 Арифметические и логические основы вычислительной техники : пособие / Н. А. Искра, И. В. Лукьянова, Ю. А. Луцик. – Минск : БГУИР, 2016. – 75 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Сумматор-умножитель первого типа. Схема электрическая структурная

ПРИЛОЖЕНИЕ Б

(обязательное)

Одноразрядный четверичный сумматор. Схема электрическая функциональная

ПРИЛОЖЕНИЕ В

(обязательное)

Одноразрядный четверичный умножитель. Схема электрическая функциональная

ПРИЛОЖЕНИЕ Г

(обязательное)

Одноразрядный четверичный сумматор. Реализация на мультиплексорах. Схема электрическая функциональная

ПРИЛОЖЕНИЕ Д

(обязательное)

Поиск простых импликант $C_1 * C_1$

ПРИЛОЖЕНИЕ Е

(обязательное)

Сумматор-умножитель. Ведомость документов