МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Лабораторная работа

по дисциплине «Цифровые устройства приводных систем"

Вариант 2

Выполнил студент:
Анзимиров Кирилл Львович

Группа:
М70-404С-22

Проверил:
Кривелев А.В.

Москва 2025

Содержание

Цель работы

Разработать параметрическое описание минимизированной полностью определённой четырёхместной функции на основе транзисторов (1) минимальная сложность из МДНФ и МКНФ, 2) на основе ПЗУ), встроенных примитивов (максимальная сложность из МДНФ и МКНФ, Жегалкин, Пирс и Шеффер) и пользовательского примитива (таблица истинности).

Исходные данные

Булева функция задана в виде:
$y = f(x_3, x_2, x_1, x_0) = V(0, 2, 6, 7, 9, 11)$

Аналитические выражения

Минимизация функции

Построим для заданной функции таблицу истинности с минтермами m_i и макстермами M_i (табл.1).

i	$x_3x_2x_1x_0$	F	m_i	M_i
0	0000	1	$\overline{x_3}\overline{x_2}\overline{x_1}\overline{x_0}$	
1	0001	0		$x_3 + x_2 + x_1 + \overline{x_0}$
2	0010	1	$\overline{x_3}\overline{x_2}x_1\overline{x_0}$	
3	0011	0		$x_3 + x_2 + \overline{x_1} + \overline{x_0}$
4	0100	0		$x_3 + \overline{x_2} + x_1 + x_0$
5	0101	0		$x_3 + \overline{x_2} + x_1 + \overline{x_0}$
6	0110	1	$\overline{x_3}x_2x_1\overline{x_0}$	
7	0111	1	$\overline{x_3}x_2x_1x_0$	
8	1000	0		$\overline{x_3} + x_2 + x_1 + x_0$
9	1001	1	$x_3\overline{x_2}\overline{x_1}x_0$	
10	1010	0		$\overline{x_3} + x_2 + \overline{x_1} + x_0$
11	1011	1	$x_3\overline{x_2}x_1x_0$	
12	1100	0		$\overline{x_3} + \overline{x_2} + x_1 + x_0$
13	1101	0		$\overline{x_3} + \overline{x_2} + x_1 + \overline{x_0}$
14	1110	0		$\overline{x_3} + \overline{x_2} + \overline{x_1} + x_0$
15	1111	0		$\overline{x_3} + \overline{x_2} + \overline{x_1} + \overline{x_0}$

Таблица 1: Таблица истинности с минтермами и макстермами для заданной функции

С помощью карты Карно(табл.2) минимизуируем функцию $y = f(x_3, x_2, x_1, x_0)$

$x_3x_2\backslash x_1x_0$	00	01	11	10
00	1	0	0	1
01	0	0	1	1
11	0	0	0	0
10	0	1	1	0

Таблица 2: Карта Карно

Минимизированное выражение в МДНФ:

$$y = x_0\overline{x_2}x_3 + x_1x_2\overline{x_3} + \overline{x_0x_2x_3}$$

Минимизированное выражение в МКНФ:

$$y = (\overline{x_3} + x_0)(\overline{x_3} + \overline{x_2})(x_1 + \overline{x_2})(x_3 + x_2 + \overline{x_0})$$

Сложность найденых минимизированных функций: МДНФ - сложность 9, МКНФ - сложность 9.

Преобразуем МДНФ в полином Жегалкина:

$$y = 1 \oplus x_0 \oplus x_2 \oplus x_3 \oplus x_0x_2 \oplus x_1x_2 \oplus x_2x_3 \oplus x_2x_1x_3$$

Реализации на языке Verilog HDL

1. Параметрическое описание МДНФ на транзисторах

```
module inv(input x, output y);
  supply1 vcc;
  supply0 gnd;

  pmos(y, vcc, x);
  nmos(y, gnd, x);
endmodule


module mdnt_transistors (input [3:0] x, output y);
  supply1 vcc;
  supply0 gnd;

  wire nx1, nx2, nx3, nx0, w1, w2, w3;

  inv i1 (x[2], nx2);
  inv i2 (x[3], nx3);
  inv i3 (x[0], nx0);
  inv i4 (x[1], nx1);

  pmos(w1, vcc, x[3]);  pmos(w1, vcc, nx0);
  pmos(w2, w1, x[3]);  pmos(w2, w1, x[2]);
  pmos(w3, w2,  x[2]);  pmos(w3, w2,  nx1);
  pmos(y,  w3,  nx3);   pmos(y,  w3,  nx2);  pmos(y, w3, x[0]);

  pulldown(y);
endmodule
```

Листинг 1: Реализация МДНФ на транзисторах

2. Параметрическое описание на основе ПЗУ

```
module rom_based(input [3:0] x, output y);
 supply0 gnd;
supply1 vcc;
wire[3:0] nx;
wire[15:0] m;
wire ny;

pmos(nx[0],vcc,x[0]); pmos(nx[1],vcc,x[1]); pmos(nx[2],vcc,x[2]); pmos(nx[3],vcc,x[3]);
nmos(nx[0],gnd,x[0]); nmos(nx[1],gnd,x[1]); nmos(nx[2],gnd,x[2]); nmos(nx[3],gnd,x[3]);

nmos(m[0],gnd, x[3]); nmos(m[0],gnd, x[2]); nmos(m[0],gnd, x[1]); nmos(m[0],gnd, x[0]); rnmos(m[0],vcc,vcc);
nmos(m[1],gnd, x[3]); nmos(m[1],gnd, x[2]); nmos(m[1],gnd, x[1]); nmos(m[1],gnd, nx[0]); rnmos(m[1],vcc,vcc);
nmos(m[2],gnd, x[3]); nmos(m[2],gnd, x[2]); nmos(m[2],gnd, nx[1]); nmos(m[2],gnd, x[0]); rnmos(m[2],vcc,vcc);
nmos(m[3],gnd, x[3]); nmos(m[3],gnd, x[2]); nmos(m[3],gnd, nx[1]); nmos(m[3],gnd, nx[0]); rnmos(m[3],vcc,vcc);
nmos(m[4],gnd, x[3]); nmos(m[4],gnd, nx[2]); nmos(m[4],gnd, x[1]); nmos(m[4],gnd, x[0]); rnmos(m[4],vcc,vcc);
nmos(m[5],gnd, x[3]); nmos(m[5],gnd, nx[2]); nmos(m[5],gnd, x[1]); nmos(m[5],gnd, nx[0]); rnmos(m[5],vcc,vcc);
nmos(m[6],gnd, x[3]); nmos(m[6],gnd, nx[2]); nmos(m[6],gnd, nx[1]); nmos(m[6],gnd, x[0]); rnmos(m[6],vcc,vcc);
nmos(m[7],gnd, x[3]); nmos(m[7],gnd, nx[2]); nmos(m[7],gnd, nx[1]); nmos(m[7],gnd, nx[0]); rnmos(m[7],vcc,vcc);
nmos(m[8],gnd, nx[3]); nmos(m[8],gnd, x[2]); nmos(m[8],gnd, x[1]); nmos(m[8],gnd, x[0]); rnmos(m[8],vcc,vcc);
nmos(m[9],gnd, nx[3]); nmos(m[9],gnd, x[2]); nmos(m[9],gnd, x[1]); nmos(m[9],gnd, nx[0]); rnmos(m[9],vcc,vcc);
nmos(m[10],gnd, nx[3]); nmos(m[10],gnd, x[2]); nmos(m[10],gnd, nx[1]); nmos(m[10],gnd, x[0]); rnmos(m[10],vcc,vcc);
nmos(m[11],gnd, nx[3]); nmos(m[11],gnd, x[2]); nmos(m[11],gnd, nx[1]); nmos(m[11],gnd, nx[0]); rnmos(m[11],vcc,vcc);
nmos(m[12],gnd, nx[3]); nmos(m[12],gnd, nx[2]); nmos(m[12],gnd, x[1]); nmos(m[12],gnd, x[0]); rnmos(m[12],vcc,vcc);
nmos(m[13],gnd, nx[3]); nmos(m[13],gnd, nx[2]); nmos(m[13],gnd, x[1]); nmos(m[13],gnd, nx[0]); rnmos(m[13],vcc,vcc);
nmos(m[14],gnd, nx[3]); nmos(m[14],gnd, nx[2]); nmos(m[14],gnd, nx[1]); nmos(m[14],gnd, x[0]); rnmos(m[14],vcc,vcc);
nmos(m[15],gnd, nx[3]); nmos(m[15],gnd, nx[2]); nmos(m[15],gnd, nx[1]); nmos(m[15],gnd, nx[0]); rnmos(m[15],vcc,vcc);

rnmos(ny,vcc,vcc);
nmos(ny,gnd,m[0]);
nmos(ny,gnd,m[2]);
nmos(ny,gnd,m[6]);
nmos(ny,gnd,m[7]);
nmos(ny,gnd,m[9]);
```

```
 nmos(ny,gnd,m[11]);
 pmos(y,vcc,ny);
 nmos(y,gnd,ny);

 endmodule
```

Листинг 2: Реализация на основе ПЗУ

3.1 МДНФ с использованием assign

```
module y_mdnf_assign(input [3:0] x, output z);

assign z = (x[3] & ~x[2] & x[0])
         | (~x[3] & x[2] & x[1])
         | (~x[3] & ~x[2] & ~x[0]);

endmodule
```

Листинг 3: МДНФ с использованием assign

3.2 Полином Жегалкина на примитивах

```
module zhegalkin_primitives (input [3:0] x, output z);

wire y1, y2, y3, y4, nx0, nx1, nx2, nx3;

not(nx0, x[0]);
not(nx1, x[1]);
not(nx2, x[2]);
not(nx3, x[3]);

and(y1, x[0], x[2]);
and(y2, x[1], x[2]);
and(y3, x[2], x[3]);
and(y4, x[1], x[2], x[3]);

xor(z, 1'b1, x[0], x[2], x[3], y1, y2, y3, y4);

endmodule
```

Листинг 4: Полином Жегалкина на примитивах

3.2 Полином Жегалкина с использованием assign

```
module zhegalkin_assign(input [3:0] x, output y);
  assign y =  1'b1^ x[0] ^ x[2] ^ x[3] ^ (x[0] & x[2]) ^
              (x[2] & x[1]) ^ (x[3] & x[2]) ^
              (x[3] & x[2] & x[1]);
endmodule
```

Листинг 5: Полином Жегалкина с использованием assign

3.3 Базис Пирса

```
module pierce (input [3:0] x, output z);

wire y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, nx0, nx1, nx2, nx3;

not(nx0, x[0]);
not(nx1, x[1]);
not(nx2, x[2]);
not(nx3, x[3]);

and(y1, nx3, nx2, nx1, x[0]);
and(y2, nx3, nx2, x[1], x[0]);
and(y3, nx3, x[2], nx1, nx0);
and(y4, x[3], nx2, nx1, nx0);
and(y5, x[3], nx2, x[1], nx0);
and(y6, x[3], x[2], nx1, nx0);
and(y7, x[3], x[2], nx1, x[0]);
and(y8, x[3], x[2], x[1], nx0);
and(y9, x[3], x[2], x[1], x[0]);
and(y10, nx3, x[2], nx1, x[0]);

nor(z, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10);

endmodule
```

Листинг 6: Реализация в базисе Пирса

3.4 Базис Шеффера

```
module sheffer (input [3:0] x, output z);

wire y1, y2, y3, y4, y5, y6, nx0, nx1, nx2, nx3;

not (nx0, x[0]);
not (nx1, x[1]);
not (nx2, x[2]);
not (nx3, x[3]);

or (y1, x[3], x[2], x[1], x[0]);
or (y2, x[3], x[2], nx1, x[0]);
or (y3, x[3],nx2, nx1, x[0]);
or (y4, x[3], nx2, nx1, nx0);
or (y5, nx3, x[2], x[1], nx0);
or (y6, nx3, x[2], nx1, nx0);

nand (z, y1, y2, y3, y4, y5, y6);

endmodule
```

Листинг 7: Реализация в базисе Шеффера

4.0 Пользовательский примитив (UDP)

```
primitive tabl(y, x3, x2, x1, x0);
  output y;
  input  x3, x2, x1, x0;

  table
    0 0 0 0 : 1;
    0 0 0 1 : 0;
    0 0 1 0 : 1;
    0 0 1 1 : 0;
    0 1 0 0 : 0;
    0 1 0 1 : 0;
    0 1 1 0 : 1;
    0 1 1 1 : 1;
    1 0 0 0 : 0;
    1 0 0 1 : 1;
    1 0 1 0 : 0;
    1 0 1 1 : 1;
    1 1 0 0 : 0;
    1 1 0 1 : 0;
    1 1 1 0 : 0;
    1 1 1 1 : 0;
  endtable
endprimitive

module udp_module(input [3:0] x, output y);
  tabl udp_inst(y, x[3], x[2], x[1], x[0]);
endmodule
```

Листинг 8: Пользовательский примитив

4.1 Пользовательский примитив с буфером

```
module tabl_with_buffer(input [3:0] x, input z, output y);
  wire w;
  tabl udp_inst(w, x[3], x[2], x[1], x[0]);
  bufif1 b(y, w, z);
endmodule
```

Листинг 9: Пользовательский примитив с буфером

Тестовый модуль

```
`timescale 1ns/1ns
module comprehensive_test;

  reg [3:0] x = 0;
  reg z = 0;
  wire y_mdnt_trans, y_rom, y_mdnf_prim, y_mdnf_assign;
  wire y_zheg_prim, y_zheg_assign, y_pierce, y_sheffer;
  wire y_udp, y_udp_buf;


  mdnt_transistors        m1(x, y_mdnt_trans);
  rom_based               m2(x, y_rom);
  mdnf_primitives         m3(x, y_mdnf_prim);
  mdnf_assign             m4(x, y_mdnf_assign);
  zhegalkin_primitives    m5(x, y_zheg_prim);
  zhegalkin_assign        m6(x, y_zheg_assign);
  pierce                  m7(x, y_pierce);
  sheffer                 m8(x, y_sheffer);
  udp_module              m9(x, y_udp);
  udp_with_buffer         m10(x, z, y_udp_buf);

  initial begin
    $display("================================================
␣␣␣␣================================================================");
    $display("|␣i␣|␣x3␣x2␣x1␣x0|MDNT_TR|␣␣ROM␣␣|␣MDNF_P|␣MDNF_A|␣
       ZHEG_P|␣ZHEG_A|␣PIERCE|␣SHEFF␣|␣␣UDP␣␣|UDP_BUF|");
    $display("================================================
␣␣␣␣================================================================");

    for (integer i = 0; i <= 15; i++) begin
      x = i;
      #1;
      $display("|%2d␣|␣%b␣␣%b␣␣%b␣␣%b␣|␣␣␣%b␣␣␣␣|␣␣␣%b␣␣␣␣|␣␣␣%b␣␣␣␣|␣
         ␣␣%b␣␣␣␣|␣␣␣%b␣␣␣␣|␣␣␣%b␣␣␣␣|␣␣␣%b␣␣␣␣|␣␣␣%b␣␣␣␣|␣␣␣%b␣␣␣␣|␣␣␣%b
         ␣␣␣␣|",
               i, x[3], x[2], x[1], x[0],
               y_mdnt_trans, y_rom, y_mdnf_prim, y_mdnf_assign,
               y_zheg_prim, y_zheg_assign, y_pierce, y_sheffer,
               y_udp, y_udp_buf);
    end
    $display("================================================
␣␣␣␣================================================================");
  end
endmodule
```

Листинг 10: Комплексный тестовый модуль

Результаты моделирования

i	x3	x2	x1	x0	MDNT_TR	ROM	MDNF_P	MDNF_A	ZHEG_P	ZHEG_A	PIERCE	SHEFF	UDP	UDP_BUF
0	0	0	0	0	1	1	1	1	1	1	1	1	1	z
1	0	0	0	1	0	0	0	0	0	0	0	0	0	z
2	0	0	1	0	1	1	1	1	1	1	1	1	1	z
3	0	0	1	1	0	0	0	0	0	0	0	0	0	z
4	0	1	0	0	0	0	0	0	0	0	0	0	0	z
5	0	1	0	1	0	0	0	0	0	0	0	0	0	z
6	0	1	1	0	1	1	1	1	1	1	1	1	1	z
7	0	1	1	1	1	1	1	1	1	1	1	1	1	z
8	1	0	0	0	0	0	0	0	0	0	0	0	0	z
9	1	0	0	1	1	1	1	1	1	1	1	1	1	z
10	1	0	1	0	0	0	0	0	0	0	0	0	0	z
11	1	0	1	1	1	1	1	1	1	1	1	1	1	z
12	1	1	0	0	0	0	0	0	0	0	0	0	0	z
13	1	1	0	1	0	0	0	0	0	0	0	0	0	z
14	1	1	1	0	0	0	0	0	0	0	0	0	0	z
15	1	1	1	1	0	0	0	0	0	0	0	0	0	z

Выводы по работе

В процессе выполнения лабораторной работы были реализованы и исследованы несколько вариантов задания одной и той же булевой функции на языке Verilog HDL:

1. **Транзисторный уровень** — позволил рассмотреть работу схемы на самом низком уровне абстракции, проследить прохождение сигналов через отдельные транзисторы и понять, за счёт чего формируются логические операции и как можно экономить элементную базу.

2. **Реализация на ПЗУ** — продемонстрировала табличный способ задания функции по её таблице истинности. Такой подход прост для анализа и модификации, но обычно требует большего числа аппаратных ресурсов по сравнению с минимизированными логическими схемами.

3. **Использование встроенных примитивов** — показало удобство стандартных логических элементов Verilog: описания остаются наглядными, при этом синтезатор может достаточно эффективно оптимизировать получающуюся схему.

4. **Реализация в различных логических базисах** (И–НЕ, ИЛИ–НЕ, полином Жегалкина) — подтвердила, что одна и та же функция может быть эквивалентно представлена в разных математических формах, что даёт возможность выбирать базис исходя из требований к структуре и сложности схемы.

5. **Пользовательские примитивы** — показали, как можно создавать собственные логические блоки с заданным поведением (по таблице истинности или уравнениям), что повышает гибкость при описании специализированных узлов.

Моделирование всех вариантов показало совпадение выходных значений, что подтверждает корректность разработанных описаний. С точки зрения минимизации аппаратных затрат наиболее выгодной оказалась реализация на основе минимизированной МДНФ, тогда как вариант с ПЗУ является наиболее универсальным и наглядным, так как напрямую опирается на таблицу истинности функции.