Для примера рассмотрим логическую функцию (1), в которой аргумент зависит от 5 переменных. Рассмотрим алгоритм практической реализации (создания) конечного устройства от этапа анализа логической функции до моделирования устройства, реализующего данную функцию.

$$F = \neg(C \wedge B \vee D) \vee \neg(C \wedge D) \wedge (A \wedge B \vee C \wedge D) \vee A \wedge \neg C \wedge B \wedge \neg D \wedge (A \wedge D \vee \neg(B \wedge C) \vee C) \vee \neg(E \wedge C) \wedge \neg(E \vee C \wedge B) \quad (1)$$

Подготовительным этапом является анализ заданной логической функции, в котором необходимо построить ее таблицу истинности. Для простоты и точности результата расчет проведем в среде MathCad, используя стандартные логические операции умножения, сложения и инверсии. На рисунке 1 приведена таблица истинности для функции 1, содержащей $2^n = 32$ значения, где n – количество переменных (входов).

На следующем этапе необходимо провести минимизацию (1). Здесь можно воспользоваться несколькими методами: методом уравнений Де-Моргана или используя метод карт Карно. В итоге будет найдена дизъюктивно-нормальная форма (ДНФ), сведённая к минимуму входных значений и не содержащая в записи скобок. Метод Де-Моргана является простым, но для нахождения ДНФ сложных (содержащих большое количество членов и входных значений), этот метод использовать не рационально. Поэтому воспользуемся методом карт Карно. Составим карту Карно для функции (1), используя таблицы истинности на рисунке 1.

Таблица 1 – карта Карно.

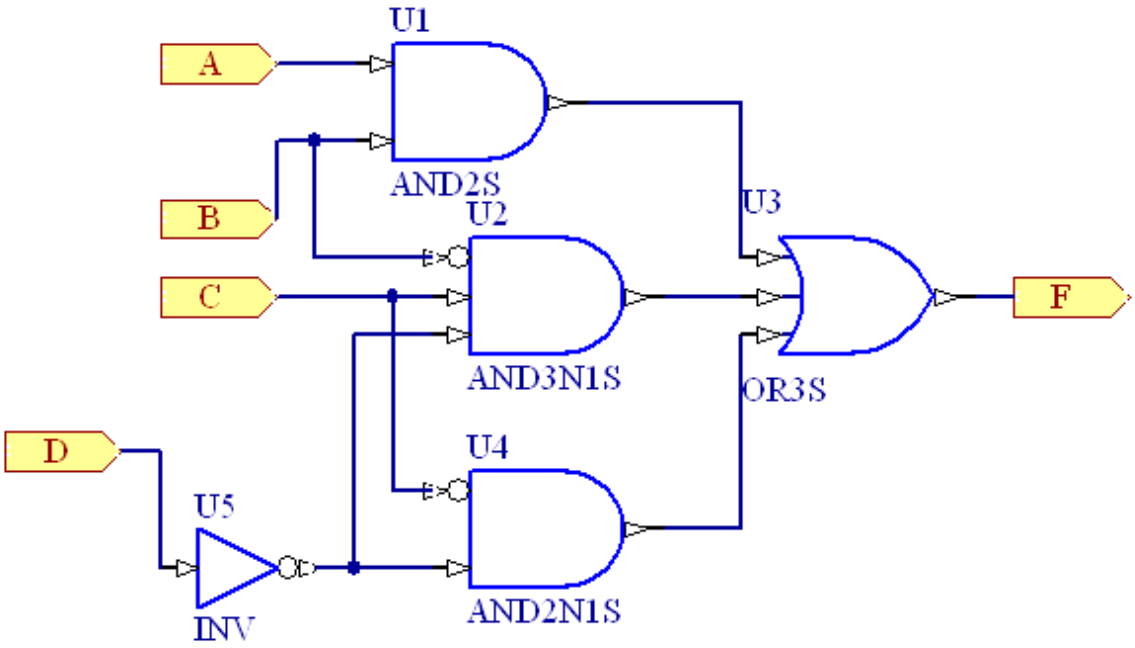
	000	001	011	010	110	111	101	100
00	1	1	0	0	0	0	1	1
01	1	1	0	0	0	0	0	0
11	1	1	1	1	1	1	1	1
10	1	1	0	0	0	0	1	1

В таблице 1 штриховкой показаны объединенные термы. Всего три терма. Далее запишем минимизированную логическую функцию, используя таблицу 1.

$$F = \neg C \wedge \neg D \vee A \wedge B \vee \neg B \wedge C \wedge \neg D \quad (2)$$

Анализируя выражение (2) можно сказать, что функция не зависит от пятого входного значения (переменная *E*). Проверка с использованием таблицы истинности подтверждает правильность выражения (2).

Далее переходим к этапу схемотехнического проектирования с использованием логических блоков. Для этого используем пакет AltiumDesigner. Создадим проект FPGA, составим логическую схему, состоящую из трех элементов AND, реализующих логическую конъюнкцию, одного трех-входового дизъюнктора (элемент OR) и одного инвертора (элемент INV).



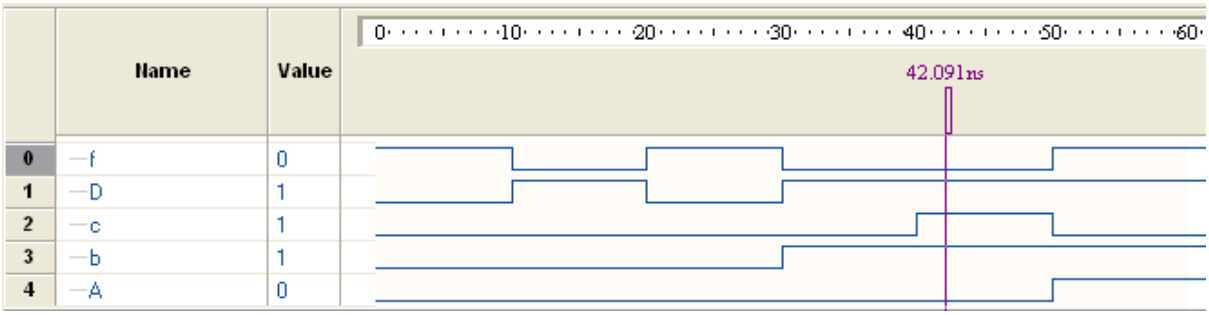
Присвоим каждому входному сигналу свое состояние, переключающиеся по истечению 10 ns. Это реализуется с использованием языка VHDL. Для входа A:

A <= '0';

wait for 20 ns;

Аналогично для остальных входов.

Далее запустим процесс симуляцы и проверим правильность работы проектируемого устройства. Для этого воспользуемся …… (че-то там синее фиг знает как называется). В результате получим временные диаграммы состояния сигналов на входе и выходе устройства (рисунок).



В момент времени t=/….. происходит то-то, в момент t2= то то

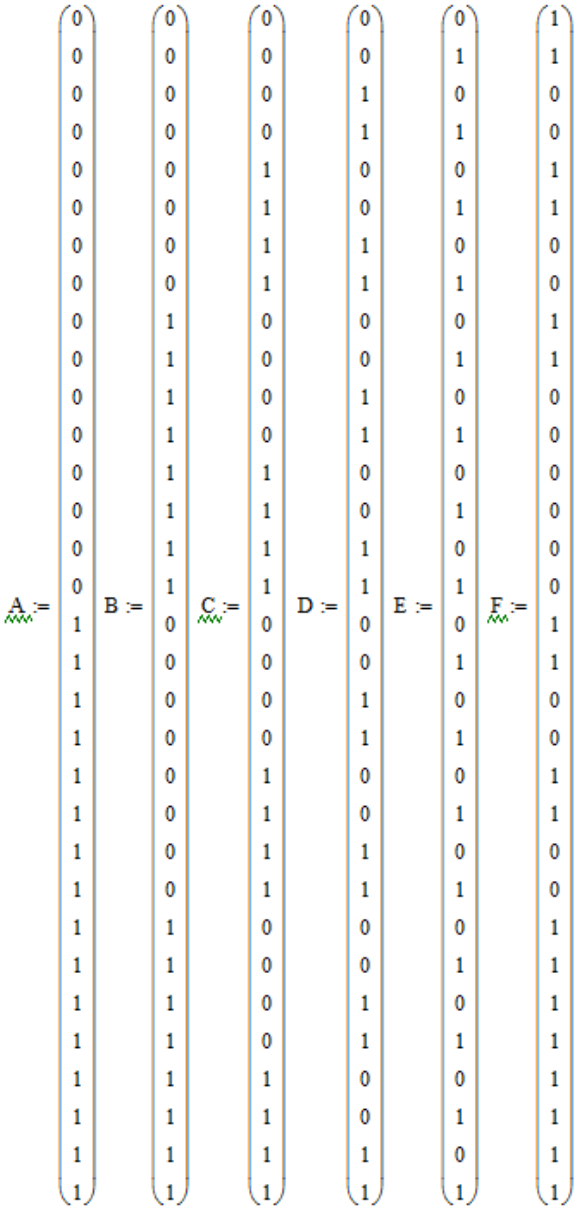


Рисунок 1 – таблица истинности для функции (1).

Для создания логической схемы (проведения схемотехнического проектирования) исходную логическую функцию необходимо упростить (минимизировать). Здесь необходимо учитывать следующее: использование стандартных булевых операций является допустимым, но не рациональным. Поэтому следует выбрать другой метод минимизации, позволяющий быстро и наглядно показать упрощение логических функций. Из курса основ дискретной математики известно несколько подобных методов: . Наиболее оптимальным и простым является метод минимизации логических функций с использованием карт Карно. Фактически, данная карта представляет собой геометрическое тело, грани которого замыкаются по краям и введенные значения могут быть объеденены в …..

После минимизации логической функции и проверки ее на соответствие исходной таблицы истинности, можно приступать к этапу схемотехнического проектирования. На данном этапе становится актуальным выбор среды, в которой будет проведено моделирование. В данной работе проводится сравнение двух САПР, позволяющих проводить данный вид анализа (моделирования). Первый это Altium Designer, фирмы….

$$F = \neg(C \wedge B \vee D) \vee \neg(C \wedge D) \wedge (A \wedge B \vee C \wedge D) \vee A \wedge \neg C \wedge B \wedge \neg D \wedge (A \wedge D \vee \neg(B \wedge C) \vee C) \vee \neg(E \wedge C) \wedge \neg(E \vee C \wedge B)$$

[¬(C ∨B D)] [¬(C D) (A ∨B ∧C D)] ((C ∧A ∧B D)) [A ¬∧C ∧B ¬∧D [A ∨D ¬(B ∨C) C)] [(E ∧C) ¬(E ∧C →B)]]

```
--------------------------------------------------------------
-- VHDL Testbench for fpga_project1
-- 2014 3 4 19 21 45
-- Created by "EditVHDL"
-- "Copyright (c) 2002 Altium Limited"
--------------------------------------------------------------

Library IEEE;
Use     IEEE.std_logic_1164.all;
Use     IEEE.std_logic_textio.all;
Use     STD.textio.all;
--------------------------------------------------------------

--------------------------------------------------------------
entity Testfpga_project1 is
end Testfpga_project1;
--------------------------------------------------------------

--------------------------------------------------------------
architecture stimulus of Testfpga_project1 is
    file RESULTS: TEXT open WRITE_MODE is "results.txt";
    procedure WRITE_RESULTS(
        A: std_logic;
        B: std_logic;
        C: std_logic;
        D: std_logic;
        F: std_logic
    ) is
        variable l_out : line;
    begin
        write(l_out, now, right, 15);
        write(l_out, A, right, 2);
        write(l_out, B, right, 2);
        write(l_out, C, right, 2);
        write(l_out, D, right, 2);
        write(l_out, F, right, 2);
        writeline(RESULTS, l_out);
    end procedure;

    component fpga_project1
        port (
            A: in std_logic;
            B: in std_logic;
            C: in std_logic;
            D: in std_logic;
            F: out std_logic
        );
    end component;

    signal A: std_logic;
    signal B: std_logic;
    signal C: std_logic;
    signal D: std_logic;
    signal F: std_logic;

begin
    DUT:fpga_project1 port map (
        A => A,
        B => B,
        C => C,
        D => D,
        F => F
    );

    STIMULUS0:process
    begin
      A <= '0';
      wait for 20 ns;
```

```
  A <= '0';
  wait for 20 ns;
  A <= '0';
  wait for 20 ns;
  A <= '0';
  wait for 20 ns;
  A <= '1';
  wait for 20 ns;
  A <= '1';
  wait for 20 ns;
  A <= '1';
  wait for 20 ns;
  A <= '1';
  wait for 20 ns;
    wait;
end process;

WRITE_RESULTS(
    A,
    B,
    C,
    D,
    F
);

STIMULUS1:process
begin
  B <= '0';
  wait for 20 ns;
  B <= '0';
  wait for 20 ns;
  B <= '0';
  wait for 20 ns;
  B <= '0';
  wait for 20 ns;
  B <= '1';
  wait for 20 ns;
  B <= '1';
  wait for 20 ns;
  B <= '1';
  wait for 20 ns;
  B <= '1';
  wait for 20 ns;
    wait;
end process;

WRITE_RESULTS(
    A,
    B,
    C,
    D,
    F
);

STIMULUS2:process
begin
  C <= '0';
  wait for 20 ns;
  C <= '0';
  wait for 20 ns;
  C <= '0';
  wait for 20 ns;
  C <= '0';
  wait for 20 ns;
  C <= '1';
  wait for 20 ns;
  C <= '1';
  wait for 20 ns;
```

```
      C <= '1';
      wait for 20 ns;
      C <= '1';
      wait for 20 ns;
        wait;
    end process;

    WRITE_RESULTS(
        A,
        B,
        C,
        D,
        F
    );

    STIMULUS3:process
    begin
      D <= '0';
      wait for 20 ns;
      D <= '1';
      wait for 20 ns;
      D <= '0';
      wait for 20 ns;
      D <= '0';
      wait for 20 ns;
      D <= '1';
      wait for 20 ns;
      D <= '1';
      wait for 20 ns;
      D <= '1';
      wait for 20 ns;
      D <= '1';
      wait for 20 ns;
        wait;
    end process;

    WRITE_RESULTS(
        A,
        B,
        C,
        D,
        F
    );


end architecture;
-------------------------------------------------------------

-------------------------------------------------------------
```