ЦИФРОВЫЕ УЗЛЫ КОМБИНАЦИОННОГО ТИПА.

При проектировании с использованием микросхем среднего уровня интеграции, проектировщик обычно покрывает схему функциональными узлами типа дешифратор, мультиплексор, сумматор и т.п.

Рассмотрим некоторые из них.

3.1. Дешифраторы.

Дешифратором называется комбинационная схема, преобразующая входное кодовое слово размерности n в выходное кодовое слово размерности 2^n. На входы управления обычно подаются сигналы разрешения. Обобщенное представление функции дешифратора представлено на **рис.3.1**.

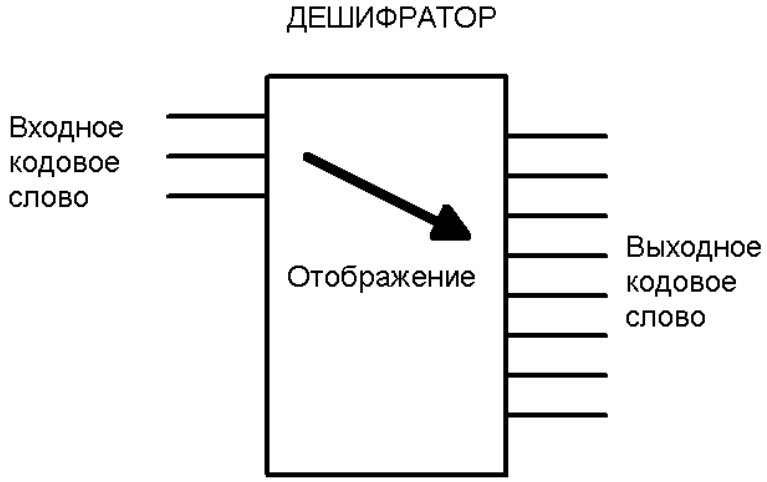
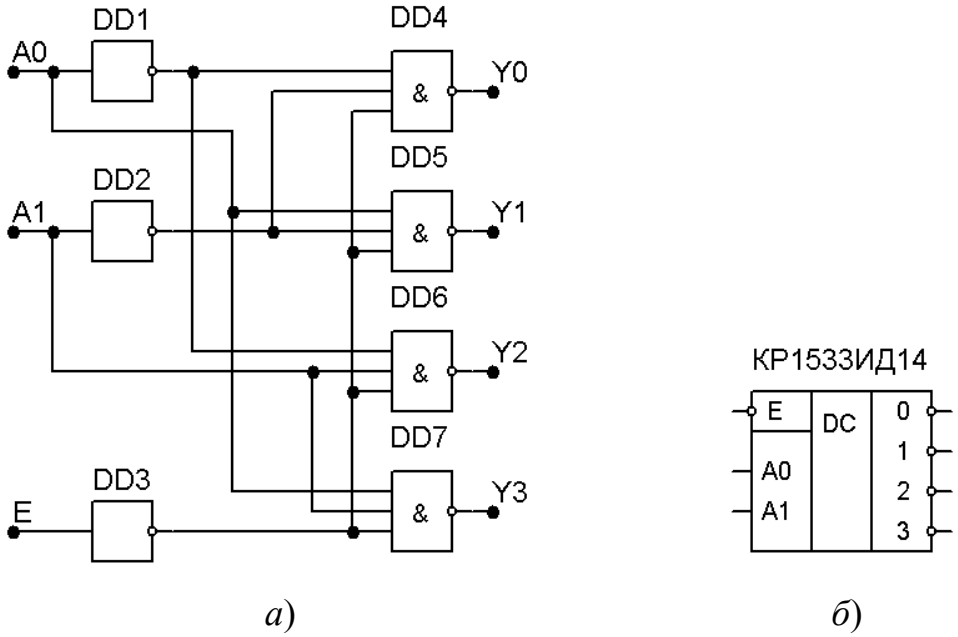


Рис. 3.1. Обобщенное представление функции дешифратора.

Промышленность выпускает микросхемы дешифраторов типа n x 2^n. На следующем рисунке показана схема дешифратора 2 x 2^2 (2 x 4) типа ИД14.



а) схема дешифратора; б) условное графическое обозначение (УГО)

Рис. 3.2. Схема дешифратора КР1533ИД14 и его УГО.

Дешифратор имеет два информационных входа A0 и A1 и вход разрешения E. Если на вход E подать запрещающий сигнал "1", то на выходах Y0 – Y3 будет сигнал "1" при любых сигналах на входах A0 и A1.

Если на вход E подать разрешающий сигнал "0", то на одном из выходов Y0 – Y3 будет сигнал низкого уровня. Номер выхода соответствует двоичному числу, поданному на входы A0 и A1.

Таблица 3.1. Таблица истинности дешифратора типа ИД14.

Входы			Выходы			
E	A1	A0	Y3	Y2	Y1	Y0
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Возможны разные варианты описаний устройств на языке VHDL. Следующее описание на VHDL дешифратора, схема которого приведена на рис. 3.2, выполнено в форме, аналогичной системе уравнений. В этом описании присутствует отдельный оператор присваивания для каждого

выходного сигнала в соответствии с таблицей истинности дешифратора (табл.3.1).

```
library IEEE;  --[1][2]
use IEEE.STD_LOGIC_1164.all;
entity DC_2_4 is
 port(E,A0,A1:in STD_LOGIC;  Y0,Y1,Y2,Y3 : out STD_LOGIC  );
end ;
architecture  FLOW of DC_2_4 is
begin
Y0<= not( not E and not A0 and not  A1 );
Y1<= not( not E and not A0 and  A1 );
Y2<= not( not E and  A0 and  not A1  );
Y3<= not( not E and A0 and A1  );
            end ;
```

Описание на VHDL теста для проверки модели дешифратора имеет следующий вид:

```
 LIBRARY ieee; USE ieee.std_logic_1164.ALL;-- [1][2]
ENTITY dec_2_4_TB IS
END dec_2_4_TB;

ARCHITECTURE behavior OF dec_2_4_TB IS
   COMPONENT DC_2_4
   PORT(
       E : IN  std_logic;
       A0 : IN  std_logic;
       A1 : IN  std_logic;
       Y0 : OUT  std_logic;
       Y1 : OUT  std_logic;
       Y2 : OUT  std_logic;
       Y3 : OUT  std_logic
      );
   END COMPONENT;
  --Inputs
  signal E : std_logic := '0';   signal A0 : std_logic := '0';
  signal A1 : std_logic := '0';
        --Outputs
  signal Y0 : std_logic;  signal Y1 : std_logic;
  signal Y2 : std_logic; signal Y3 : std_logic;
BEGIN
        -- Instantiate the Unit Under Test (UUT)
  uut: DC_2_4 PORT MAP
     (
       E => E,   A0 => A0,       A1 => A1,     Y0 => Y0,
```

```
      Y1 => Y1,  Y2 => Y2,  Y3 => Y3
     );
  -- Stimulus process
  stim_proc: process
  begin
    wait for 20 ns;      A0<='0'; A1<='1';
    wait for 20 ns;   A0<='1'; A1<='0';
    wait for 20 ns;   A0<='1'; A1<='1';
   wait for 20 ns;    A0<='0'; A1<='0';
   wait for 20 ns;E<='1';
   wait for 20 ns; E<='0'; A0<='0'; A1<='1';
    wait;
  end process;
END;
```

Временная диаграмма работы модели дешифратора, полученная в результате моделирования, показана на рисунке 3.3.

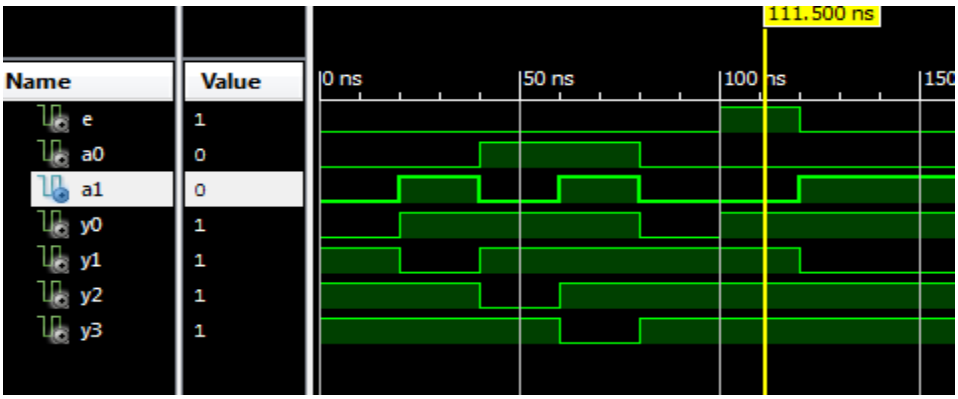


Рис. 3.3. Результаты моделирования дешифратора 2-4.

Для увеличения разрядности кодовых слов на входах и выходах дешифратора применяются схемы каскадирования.

Приведем стандартные схемы каскадирования для дешифраторов типа ИД7 (рис. 3.4) и ИД14 (рис. 3.5) — входы V0, V1, V2 разрешают работу дешифратора при V0=1, V1=0, V2=0.

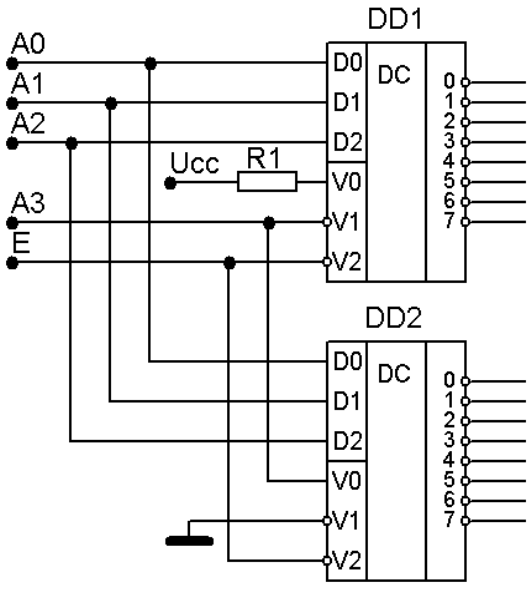


Рис. 3.4. Каскадирование дешифраторов типа ИД7 при помощи входов управления.

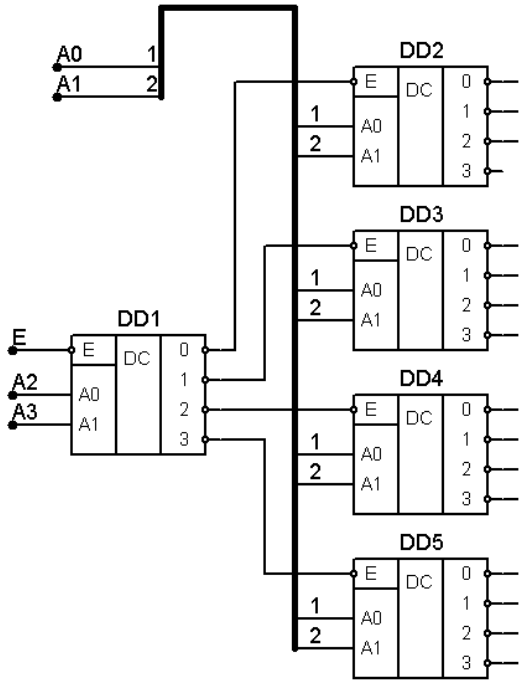


Рис. 3.5. Каскадирование дешифраторов типа ИД14 при помощи дополнительного дешифратора DD1.

3.1.1. Поведенческое описание схемы дешифратора на VHDL.

Описание дешифратора 3 x 8 типа ИД7 на VHDL можно провести, в отличие от ранее приведенного описания, в другом стиле, так называемом **поведенческом**, на основе оператора process с использованием операторов case и if и векторного представления сигналов. Такой стиль для случая многовыходовых дешифраторов позволяет сделать описания устройств компактнее.

```
library IEEE; [2][1][почему сразу не прививать поведенческий стиль?]
use IEEE.STD_LOGIC_1164.all;
entity DEC_1 is
        port( E_n : in STD_LOGIC;
              X : in STD_LOGIC_VECTOR(2 downto 0);
              Y : out STD_LOGIC_VECTOR(7 downto 0) );
end DEC_1;

architecture DEC_1_ARC of DEC_1 is
begin
        process(E_n)
        begin
              if (E_n = '0') then
                    case(X) is
                          when "000" => Y <= "11111110";
                          when "001" => Y <= "11111101";
                          when "010" => Y <= "11111011";
                          when "011" => Y <= "11110111";
                          when "100" => Y <= "11101111";
```

```
                when "101" => Y <= "11011111";
                when "110" => Y <= "10111111";
                when "111" => Y <= "01111111";
                when others => Y <= "11111111";
            end case;
        else Y <= "11111111";
        end if;
    end process;
end DEC_1_ARC;
```

Вход E_n является разрешающим, вектор X – входы {X0, X1 и X2}, вектор Y – выходы {Y0, … Y7}.

3.2. Шифратор и кодовый преобразователь.

Шифратором называется комбинационная схема, преобразующая входное кодовое слово размерности 2^n в выходное кодовое слово размерности n. На входы управления подаются сигналы разрешения.

Шифратор выполняет операцию, обратную к дешифратору. Обобщенное представление его функции представлено на рис.3.6.

На практике широко применяется комбинация схем дешифратора и шифратора для реализации кодовых преобразователей.

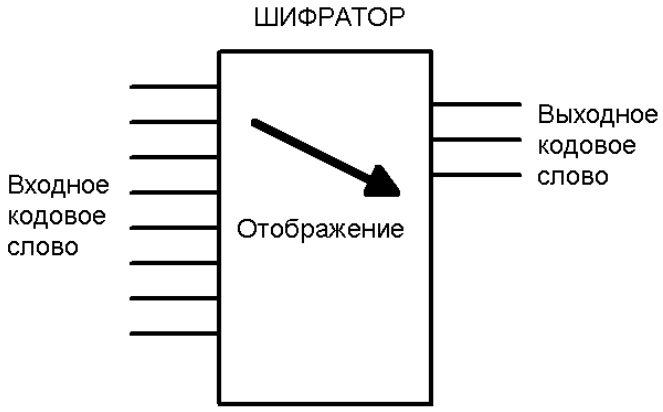


Рис. 3.6 Обобщенное представление функции шифратора.

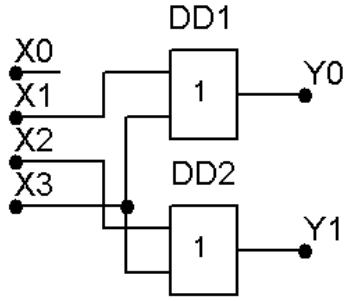
Принципиальная схема шифратора 4 х 2 представлена на рис. 3.7.

Для нормальной работы шифратора необходимо, что бы сигнал приходил только на один его вход. Понятно, что если подать сигналы "1" на любые два входа, то дешифратор будет работать неправильно.

Этого недостатка лишена схема приоритетного шифратора, в котором добавлена специальная схема, выделяющая старшую по приоритету единицу.

На базе приоритетного шифратора строится, например, схема выработки кодов клавиш в клавиатуре компьютера. В этом случае, при одновременном нажатии на несколько клавиш вырабатывается код только одной из них, имеющей максимальный приоритет

Входы				Выходы	
X0	X1	X2	X3	Y0	Y1
1	0	0	0	0	0
0	1	0	0	1	0
0	0	1	0	0	1
0	0	0	1	1	1



а) *б*)

а) - принципиальная схема
б) - таблица истинности

Рис. 3.7. Принципиальная схема шифратора и его таблица истинности.

3.2.1. Приоритетный шифратор.

Работа приоритетного шифратора 4 х 2 определяется таблицей истинности (табл. 3.2). Входные сигналы X0, X1, X2 и X3 и сигнал выбора E имеют высокий логический уровень. Приоритет сигналов убывает в порядке X3, X2, X1, X0.

Выходные сигналы шифратора Y0 и Y1, сигнал переноса E0 и сигнал группового выбора GS имеют высокий уровень.

Сигнал E разрешает работу шифратора. Выходной сигнал E0 подключается к входу разрешения E другого шифратора, имеющего более низкий приоритет. Сигнал GS имеет высокий уровень, если активный сигнал подан хотя бы на один из входов шифратора. Таким образом проводится каскадирование дешифраторов.

Таблица 3.2. Таблица истинности приоритетного шифратора.

E	X3	X2	X1	X0	Y0	Y1	GS	E0
0	x	x	x	x	0	0	0	0
1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0

1	0	0	1	x	1	0	1	0
1	0	1	x	x	0	1	1	0
1	1	x	x	x	1	1	1	0

Если на входе разрешения E сигнал низкого уровня, то при любых сигналах {Xi} на входах шифратор не работает, при этом все сигналы на выходах имеют низкий уровень.

Шифратор включается в работу, если на вход разрешения E подан сигнал высокого уровня. Если ни на один из входов X0, X1, X2 или X3 активный сигнал не подан, то вырабатывается сигнал E0, который подается на вход разрешения шифратора с меньшим приоритетом.

Если на один или несколько входов X0, X1, X2 или X3 поданы сигналы, то на выходах Y0 и Y1 появляется двоичное число, соответствующее номеру старшего по приоритету входу. В этом случае сигнал E0 равен нулю, а сигнал группового выбора GS – единице.

Схема приоритетного шифратора с последовательным запрещением входных сигналов, приведенная на рис. 3.8.

Для построения схемы приоритетного шифратора в общем случае необходимо по таблице 3.2. составить логические функции для выходных сигналов Y0, Y1 и GS. Для этого нужно объединить термы из значений входных сигналов E, X0, X1, X2 и X3, соответствующие единицам в столбцах Y0, Y1 и GS, а затем минимизировать результат. В результате получим следующие логические функции.

$$Y0 = E \cdot (\overline{X2} \cdot X1 + X2 \cdot \overline{X1} \cdot X0 + X3);$$

$$Y1 = E \cdot (X2 \cdot X1 + X2 \cdot X0 + X3);$$

$$GS = E \cdot (X3 + X2 + X1 + X0);$$

Далее необходимо реализовать эти функции при помощи логических элементов.

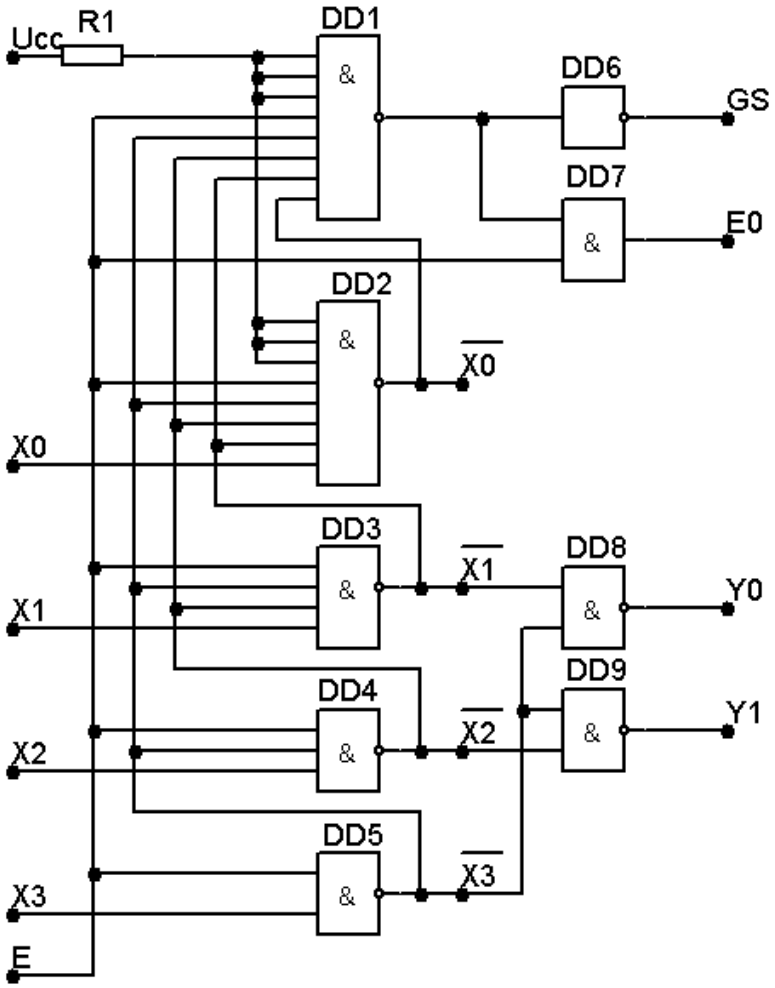


Рис. 3.8. Приоритетный шифратор на четыре входа и два выхода.

3.2.2 Описание приоритетного шифратора на VHDL

Описание приоритетного шифратора на 4 входа (рис.3.8)с именем **encoder_1** на VHDL имеет следующий вид:

```
library IEEE; --[1][2]
use IEEE.STD_LOGIC_1164.all;
entity encoder_1 is
        port(   E : in STD_LOGIC;
                X : in STD_LOGIC_VECTOR(3 downto 0);
                E0 : out STD_LOGIC;
                GS : out STD_LOGIC;
                Y : out STD_LOGIC_VECTOR(1 downto 0)  );
end encoder_1;


architecture encoder_1_ar of encoder_1 is
```

```
begin
        process (X)
        variable K: integer := 3; variable N: integer := 3;
        begin
                for K in 3 downto 0 loop
                        if(X(K) = '1') then
                                N := K;
                                exit;
                        else N := 0;
                        end if;
                end loop;
                        if (E = '1') then
        case (N) is
                when 0 => Y <= "00"; GS <= '0'; E0 <= '1';
                when 1 => Y <= "01"; GS <= '1'; E0 <= '0';
                when 2 => Y <= "10"; GS <= '1'; E0 <= '0';
                when 3 => Y <= "11"; GS <= '1'; E0 <= '0';
                when others => Y <= "00"; GS <= '0'; E0 <= '0';
        end case;
        end if;
end process;
end encoder_1_ar;
```

В этом описании в цикле loop по K выбирается активный вход со старшим приоритетом. Затем, в операторе case проводится назначение выходных сигналов.

Кодовый преобразователь.

Кодовый преобразователь предназначен для преобразования группы входных кодовых слов в группу выходных кодовых слов по заданному закону. Закон преобразования можно задать при помощи системы логических функций от группы логических переменных или при помощи таблицы.

Обобщенное представление функции кодового преобразователя представлено на рис. 3.9.

Входное слово кодового преобразователя имеет разрядность **n**, выходное – разрядность **m**. Кодовый преобразователь описывается логической функцией n переменных с m выходами.

$$Y_m = F(X_n);$$

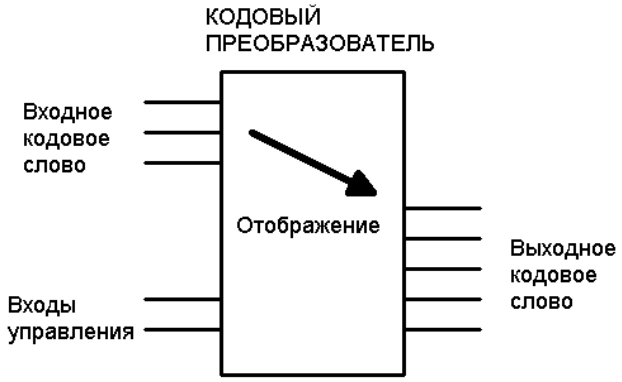


Рис. 3.9. Обобщенное представление функции кодового преобразователя.

Эту функцию необходимо преобразовать в систему из m функций, имеющих по одному выходу

$$Y_1 = f_1(X_1, X_2, ?, X_n);$$
$$Y_2 = f_2(X_1, X_2, ?, X_n);$$
$$?\,?\,?\,?\,?\,?\,?\,?$$
$$Y_m = f_m(X_1, X_2, ?, X_n);$$

Затем, после минимизации, каждая логическая функция реализуется как комбинационная схема.

Создание кодовых преобразователей по рассмотренному принципу на логических элементах средней степени интеграции обычно приводит к неэкономичным результатам. Это связано с тем, что в разных логических функциях появляются одинаковые дублирующие блоки.

Еще одним подходом к созданию кодовых преобразователей является применение комбинации дешифратора и шифратора (рис. 3.10).

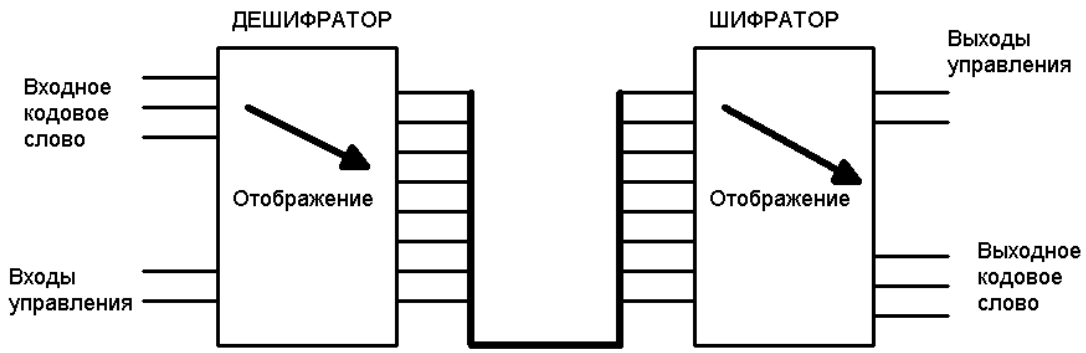


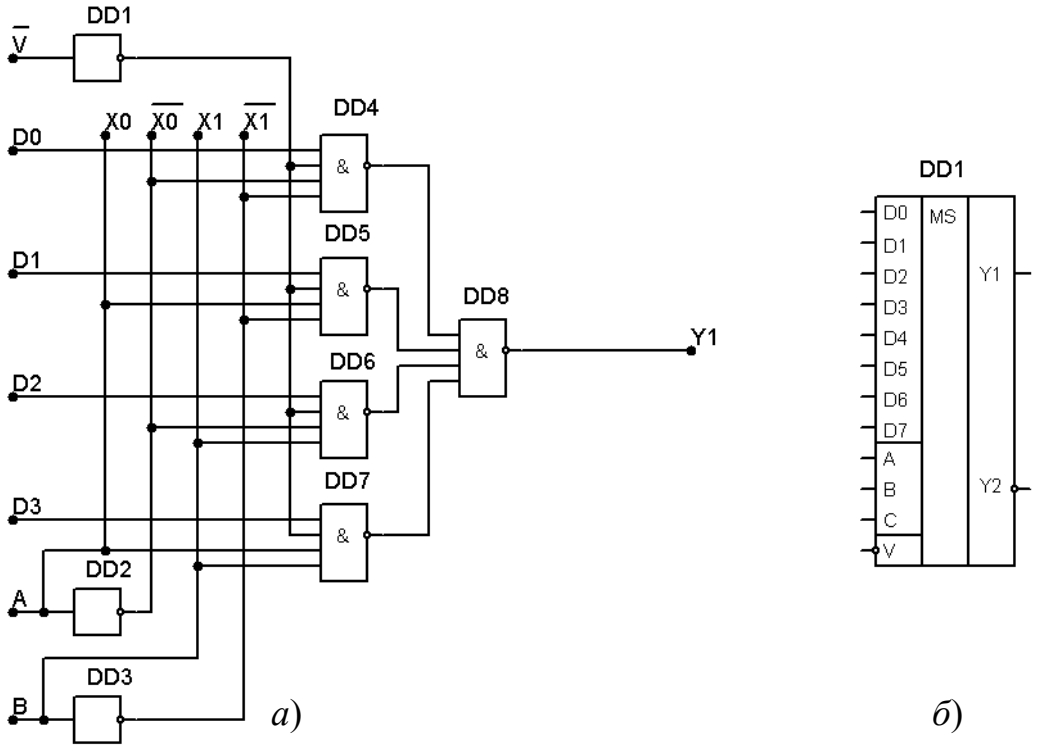
Рис. 3.10. Структура кодового преобразователя на комбинации шифратор - дешифратор.

Дешифратор преобразует входное кодовое слово в код " одна единица из n ", а шифратор преобразует этот код в выходное кодовое слово. Таблица преобразования кодов реализуется соответствующими соединениями между выходами дешифратора и входами шифратора

Мультиплексор.

Мультиплексором называется цифровой переключатель, передающий на выход сигнал с одного из нескольких входов. Выбор входа, подключаемого к выходу, определяется сигналом, поступающим на входы управления. Схема мультиплексора 4 х 1приведена на рис. 3.11.

Демультиплексор выполняет функцию, обратную мультиплексору. Эта схема в соответствии с сигналами управления подключает один вход к одному из нескольких выводов. Схема демультиплексора близка к схеме дешифратора и отличается только наличием сигнального входа. Это, например, дешифратор – демультиплексор 8 х 1 типа ИД4, являющийся, по сути, сдвоенной схемой 4 х 1.



а) – схема мультиплексора 4 х 1 типа КП2;
б) – УГО мультиплексора 8 х 1 типа КП7;

Рис. 3.11. Мультиплексор 4-1.

Для расширения числа входов применяется каскадирование мультиплексоров. Схема мультиплексора 16 х 1, выполненная на двух мультиплексорах 8 х 1, приведена на рис. 3.12.

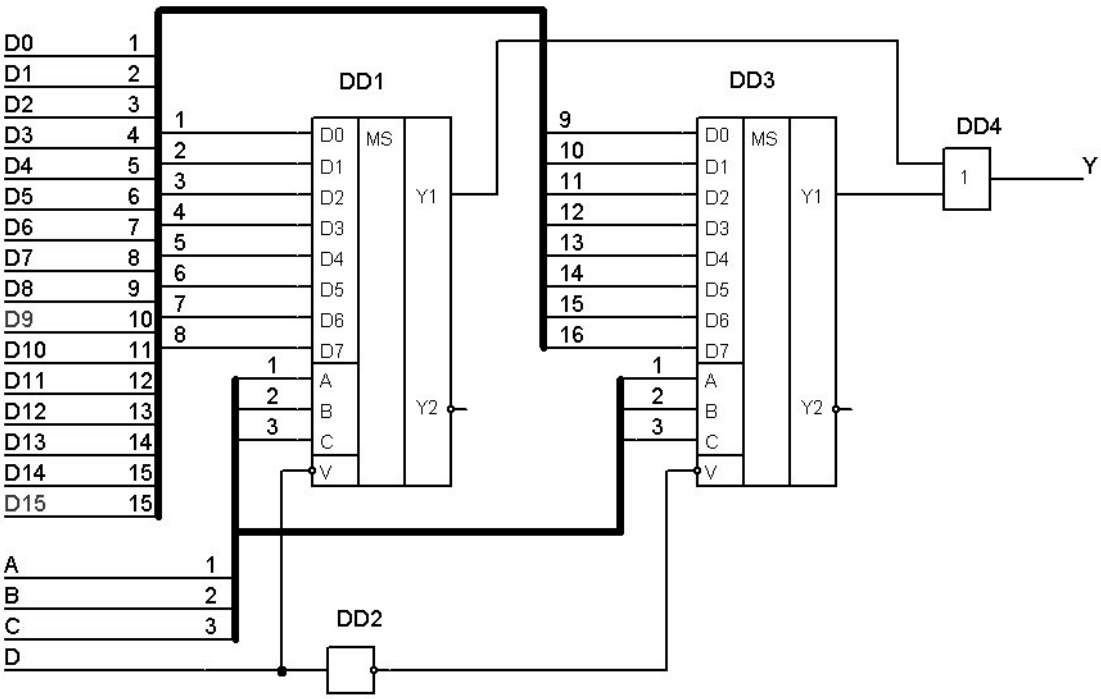


Рис. 3.12. Каскадирование мультиплексоров.

3.4.1 Описание мультиплексора 8-1 на VHDL.

```
library IEEE; use IEEE.STD_LOGIC_1164.all; [2][1]
entity MUX_8_1 is
        port( V_n : in STD_LOGIC;
              D : in STD_LOGIC_VECTOR(7 downto 0);
              X : in STD_LOGIC_VECTOR(2 downto 0);
              Y : out STD_LOGIC );
end MUX_8_1;

architecture MUX_8_1_ar of MUX_8_1 is
begin
              process (X,V_n)
              begin
      if (V_n = '0') then
              case (X) is
                    when "000" => Y <= D(0);
                    when "001" => Y <= D(1);
                    when "010" => Y <= D(2);
                    when "011" => Y <= D(3);
                    when "100" => Y <= D(4);
                    when "101" => Y <= D(5);
                    when "110" => Y <= D(6);
                    when "111" => Y <= D(7);
```

```
                when others => Y <= D(0);
                end case;
        else Y <= D(0);
        end if;
                end process;
end MUX_8_1_ar;
```

3.4.2. Реализация логических функций на мультиплексорах.

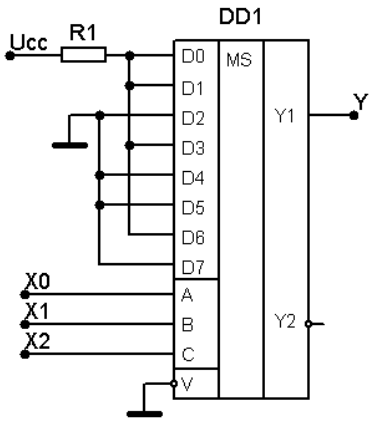
Существует множество вариантов схем, позволяющих реализовать произвольные логические функции на мультиплексорах. В качестве аргумента логической функции используются адресные входы мультиплексора {Xi}. Информационные входы мультиплексора {Dj} используются для настройки значений логической функции. Для этого на информационные входы подаются фиксированные логические уровни сигнала. Значение логической функции, зависящее от аргумента, получается на выходе Y мультиплексора.

Возможен еще один способ настройки, когда на часть информационных входов подается или один из аргументов функции или его инверсия.

Пример реализации логической функции, заданной таблицей, приведен на рис. 3.13.

№	X2	X1	X0	Y
1	0	0	0	1
2	0	0	1	1
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	1
8	1	1	1	0

а)



б)

а) – таблица истинности логической функции, б) – схема

Рис. 3.13. Реализация логической функции на мультиплексоре КР1533КП7.

3.5. Цифровой компаратор

Компаратор – это цифровой узел сравнения двух чисел.

Одноразрядный компаратор сравнивает два числа A и B и выдает однобитовый сигнал: $A = B \rightarrow 1$, $A \neq B \rightarrow 0$.

Полный компаратор сравнивает числа A и B и определяет соотношения:

$A = B$, $A > B$, $A < B$, $A \geq B$, $A \leq B$.

Сравнение чисел происходит на основе поразрядных операций над их одноименными разрядами. Очевидно, что числа равны, если равны все их одноименные разряды, т.е. в обоих числах в одноименных разрядах находятся либо 1, либо 0

Признаком равенства разрядов является следующее выражение:

$$R_i = A_i \cdot B_i \vee \overline{A}_i \cdot \overline{B}i = \overline{A_i \oplus B_i};$$

- Признаком неравенства разрядов чисел является:

$$\overline{R}_i = A_i \cdot \overline{B}_i \vee \overline{A}_i \cdot B_i = A_i \oplus B_i;$$

- Признаком равенства чисел A и B является конъюнкция приведенных выражений:

$$F_{A=B} = R = R_{n-1} \cdot R_{n-2} \cdot \ldots \cdot R_0$$

Схема цифрового компаратора, определяющего равенство двух чисел, приведена на рис. 3.14.

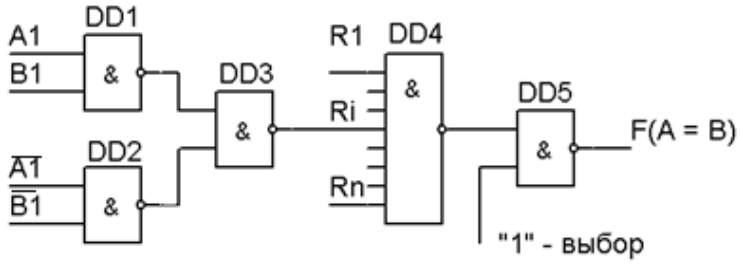


Рис. 3.14. Схема цифрового компаратора на равенство.

Сравнение чисел на неравенство $F_{A>B}$ проводится, начиная со старших разрядов. Например, если двоичное число трехразрядное, то:

если $A_2 = 1$ и $B_2 = 0$, то числа $A > B$ при любых значениях в младших разрядов чисел;

если $A_2 = 0$ и $B_2 = 1$, то числа $A < B$ при любых значениях в младших разрядов чисел;

если $A_2 = B_2$, то результат сравнения не определен, и необходимо провести анализ следующего младшего разряда числа.

Анализ трехразрядных чисел проводится на основе следующих соотношений:

$$F_{A>B} = A_2 \cdot \overline{B}_2 \vee R_2 \cdot A_1 \cdot \overline{B}_1 \vee R_1 \cdot R_2 \cdot A_0 \cdot \overline{B}_0;$$
$$F_{A<B} = \overline{A}_2 \cdot B_2 \vee R_2 \cdot \overline{A}_1 \cdot B_1 \vee R_1 \cdot R_2 \cdot \overline{A}_0 \cdot B_0;$$

для n-разрядов:

$$F_{A>B} = A_{n-1} \cdot \overline{B}_{n-1} \vee R_{n-1} \cdot A_{n-2} \cdot \overline{B}_{n-2} \vee \ldots \vee R_{n-1} \cdot R_{n-2} \cdot \ldots \cdot R_1 \cdot A_0 \cdot \overline{B}_0;$$

$$F_{A<B} = \overline{A}_{n-1} \cdot B_{n-1} \vee R_{n-1} \cdot \overline{A}_{n-2} \cdot B_{n-2} \vee \ldots \vee R_{n-1} \cdot R_{n-2} \cdot \ldots \cdot R_1 \cdot \overline{A}_0 \cdot B_0;$$

Схема полного двухразрядного компаратора приведена на рис. 3.15. Для каждого из разрядов числа вырабатываются сигналы "больше", "меньше" и "равно". Затем эти сигналы обрабатываются логической схемой, связанной с входными управляющими сигналами.

3.5.1. Описание цифрового компаратора на VHDL.

```
entity Comp_1 is [1][2]
        port(  A,B : in STD_LOGIC_VECTOR(7 downto 0);
                EQ,NE,GT,GE,LT,LE : out STD_LOGIC
              );
end Comp_1;
architecture Comp_1_ar of Comp_1 is
begin
        process (A,B)
        begin
        EQ <= '0'; NE <= '0'; GT <= '0'; GE <= '0'; LT <= '0'; LE <= '0';
        if A = B then EQ <= '1'; end if;
        if A /= B then NE <= '1'; end if;
        if A > B then GT <= '1'; end if;
        if A >= B then GE <= '1'; end if;
        if A < B then LT <= '1'; end if;
        if A =< B then LE <= '1'; end if;
        end process;
end Comp_1_ar;
```

Цифровой компаратор, описание которого на VHDL приведено выше, выполняет все операции сравнения двух 8-ми разрядных двоичных чисел. Результатом сравнения двух чисел является появление '1' в списке переменных EQ, NE, GT, GE, LT и LE.

Цифровой компаратор, схема которого представлена на рис. 3.15, выполняет три операции сравнения двух 8-ми разрядных двоичных чисел. Результат сравнения двух чисел появляется на выходах микросхем DD13, DD14 и DD15

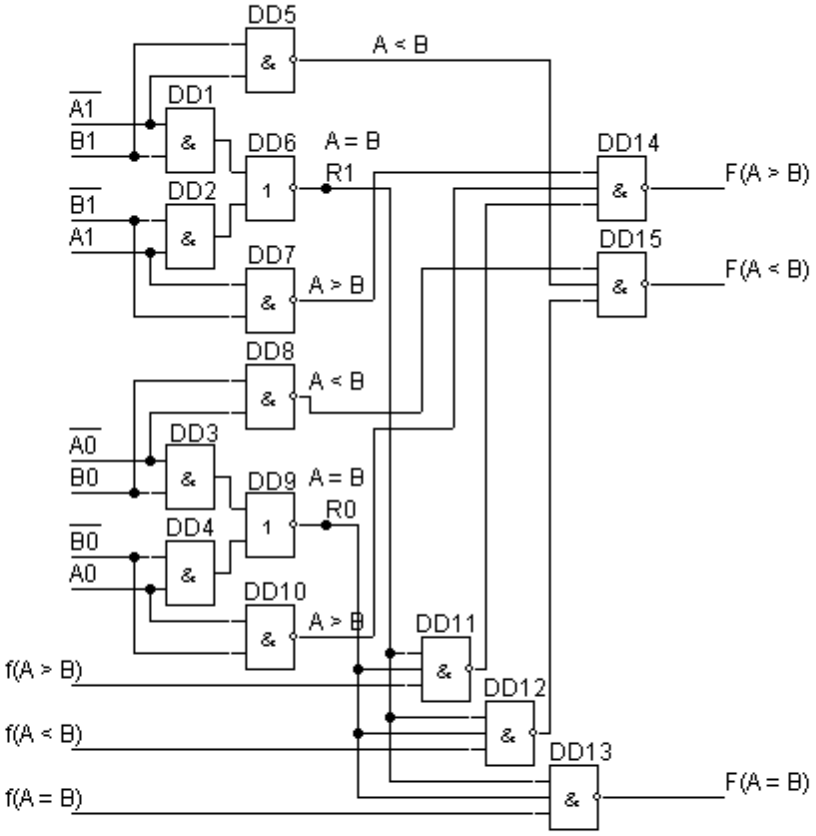


Рис. 3.15. Схема полного цифрового компаратора.