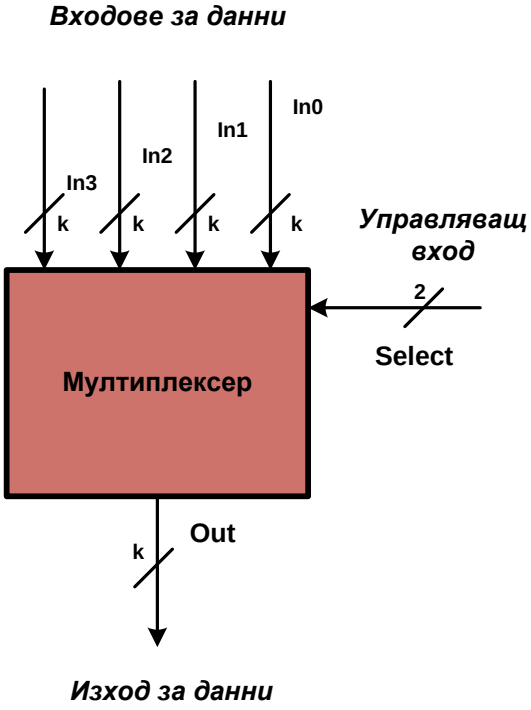
Използване комбинационни възли – мултиплексери, дешифратори, кодови преобразуватели в цифровите устройства

1. Мултиплексери (селектори на данни)

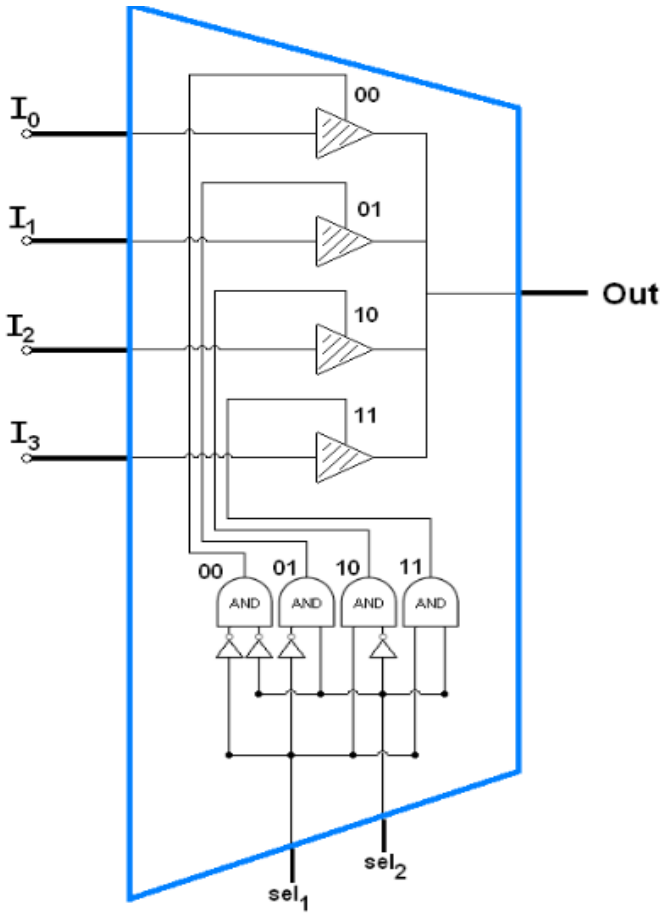
Блокова схема на мултиплексер (4:1) – с четири входа за данни:



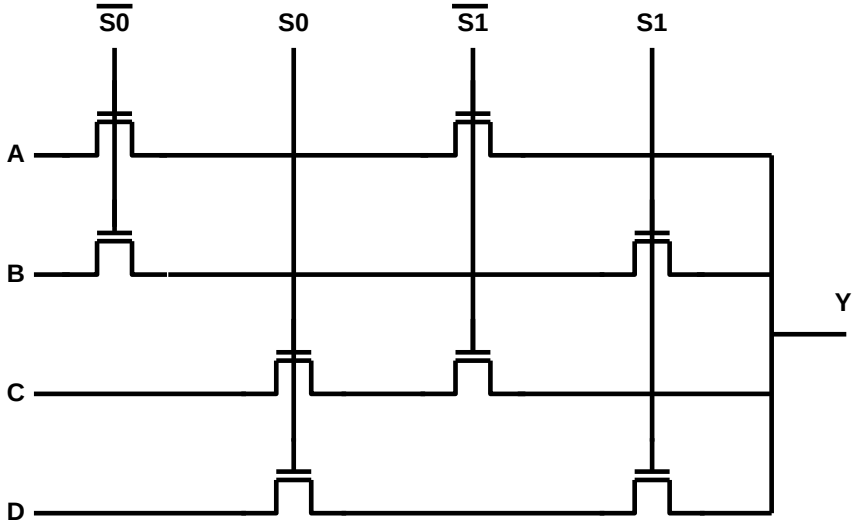
Мултиплексерът функционира по следния начин: във всеки момент изходът Out му приема стойността на сигнала от този от входовете за данни In0, In1, In2, In3, чиито номер е указан върху управляващия вход Select.
Следващата таблица на истинност показва функционирането на мултиплексера:

Select (управляващ вход)	Out (изход)
00	In0 (вход за данни No 0)
01	In1(вход за данни No 1)
10	In2(вход за данни No 2)
11	In3(вход за данни No 3)

Реализация на 4 – входов мултиплексер чрез логически елементи и буферни схеми е показана на следващата схема:

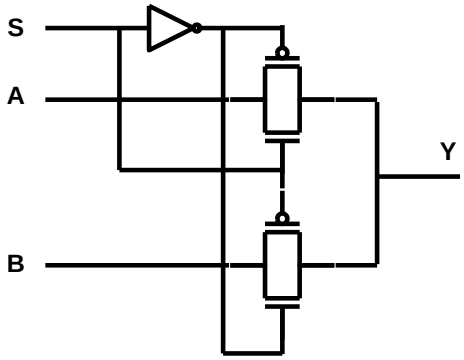


По-долу е показана реализация на мултиплексер с четири входа за данни (A,B,C,D) и управляващи входове S1, S0, използвайки Pass – логика (схемата е разглеждана и в по-предна лекция). Вдясно е показана и таблицата за истинност на мултиплексера.



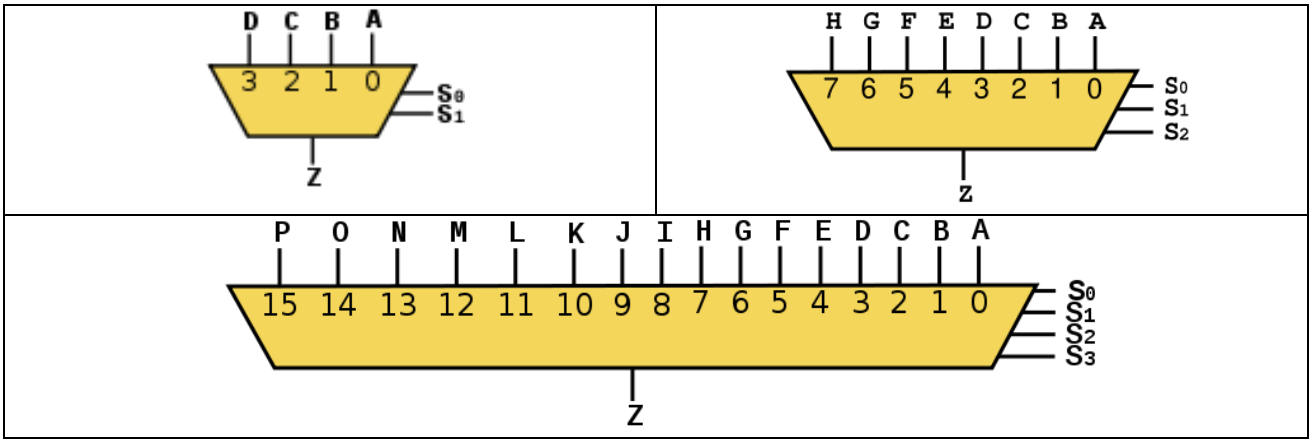
S1	S0	Y
0	0	A
0	1	C
1	0	B
1	1	D

На следващата схема е представен мултиплексер с два входа, реализиран с аналогови ключове (разглеждан също в предна лекция), който може да бъде използван за комутиране освен на цифрови,също и на аналогови сигнали:

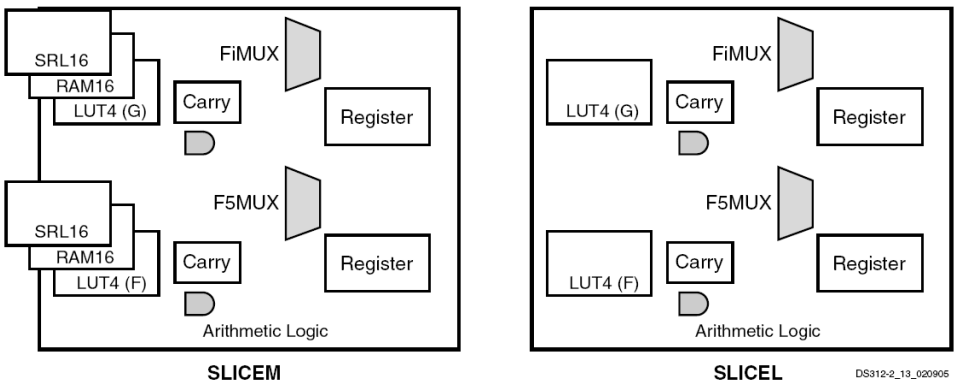


S	Y
0	B
1	A

Броят на битовете на управляващия вход на мултиплексер трябва да бъде $\lceil \log_2(n) \rceil$, където **n** е броят на входовете му за данни. В следващата таблица са показани мултиплексери с различен брой входове - 4, 8, 16 и управляващи сигнали с брой битове - съответно 2, 3, 4

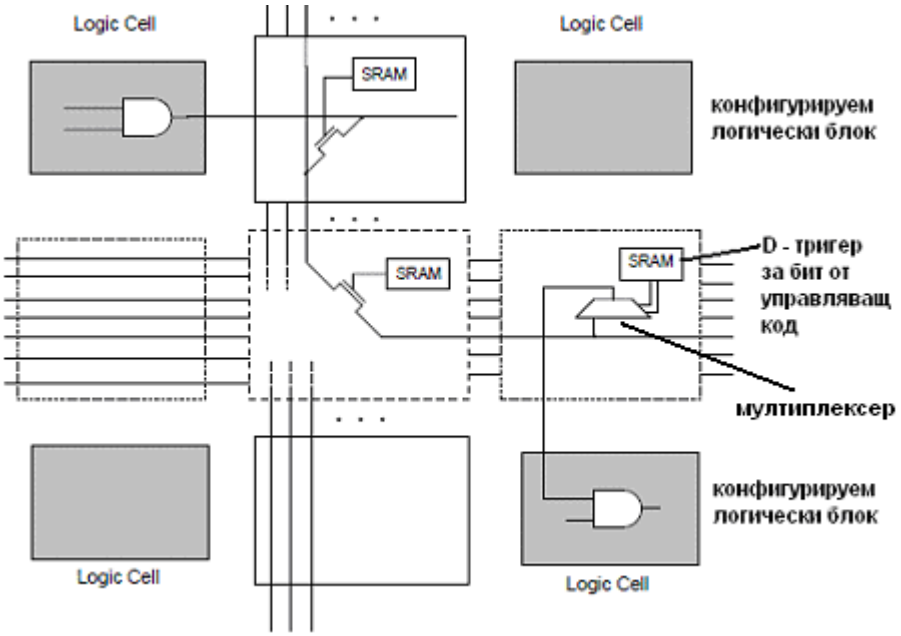


Мултиплексерите се използват много често в структурата на цифровите устройства – за селектиране от повече от една възможности на сигналите, които трябва да се разпространяват по линни или шини за данни. Следващите схеми показват използването на мултиплексери в структурата на конфигурируемите логически блокове на програмируемитв чипове със свръхголяма степен на интеграция (FPGA) – означени са с трапеци:



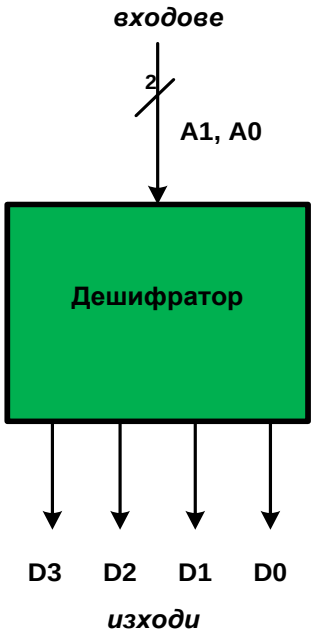
Както се вижда от горните схеми, в структурата на конфигурируемите логически блокове (най-масовите блокове на FPGA – чиповете) освен **мултиплексери**, се включват и блокове статична памет (означени с RAM16, разглеждани по-подробно в следваща лекция) и регистри за запомняне на резултатите от операциите в блоковете.

Мултиплексерите се използват много често и за реализация на комуникационните матрици, свързващи конфигурируемите логически блокове на FPGA – чиповете:



2. Дешифратори (декодери)

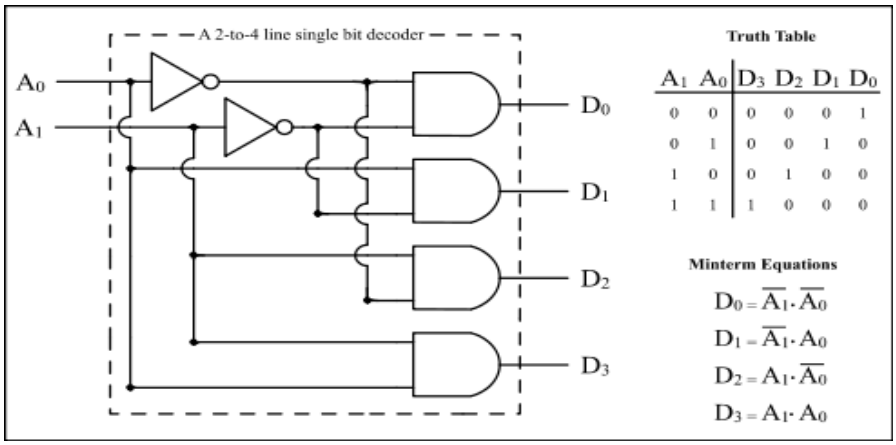
На следващата схема е показан дешифратор (2:4) - с два входа A1, A0 и четири изхода D3, D2, D1, D0. ***Всеки от входовете и изходите е еднобитов***.



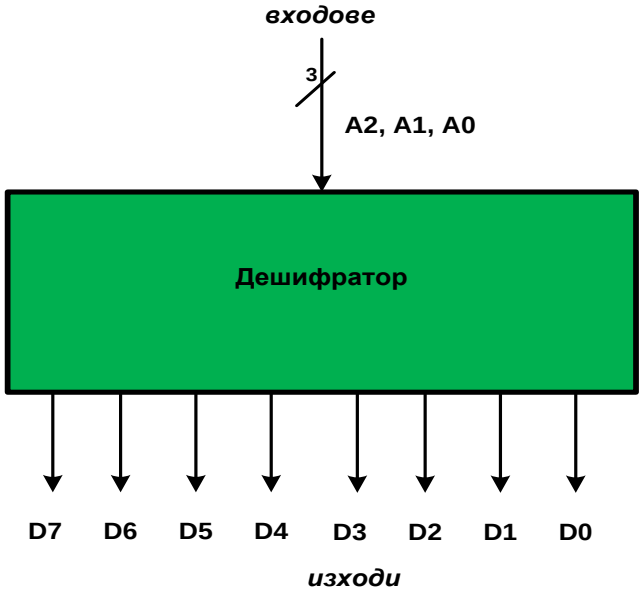
Дешифраторът функционира по следния начин: във всеки момент е активен (в ‘0’ или в ‘1’) само един от изходите – този, чиито номер е указан на входовете на дешифратора. Останалите изходи са неактивни (съответно в ‘1’ или в ‘0’). Следващата таблица за истинност показва функционирането на дешифратора:

A1 A0	D0	D1	D2	D3
00	0	1	1	1
01	1	0	1	1
10	1	1	0	1
11	1	1	1	0

Схемата по-долу показва реализация на дешифратор с логически елементи (логическата му схема). Таблицата за истинност е за дешифратор, при който активният изход е в състояние ‘1’ (а неактивните изходи са в ‘0’):



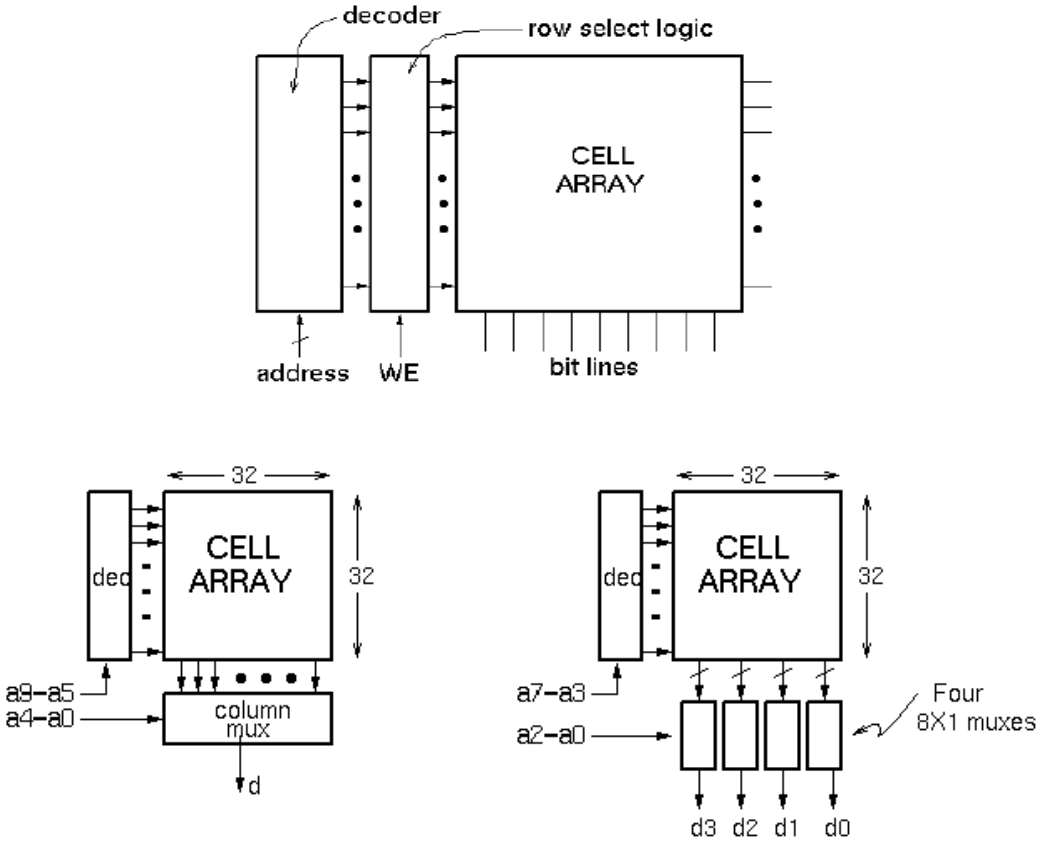
Всеки пълен дешифратор с **k** входа трябва да притежава $\mathbf{2^k}$ изхода, за да могат сигналите върху изходите му (във всеки момент само един изход активен) да отразяват всички възможни двоични кодове, които биха могли да бъдат формирани от сигналите върху входовете. По-долу е показан пълен дешифратор с три входа и съответно осем изхода (3:8):



Следва таблицата на истинност на дешифратора (всеки от изходите е активен в ‘0’):

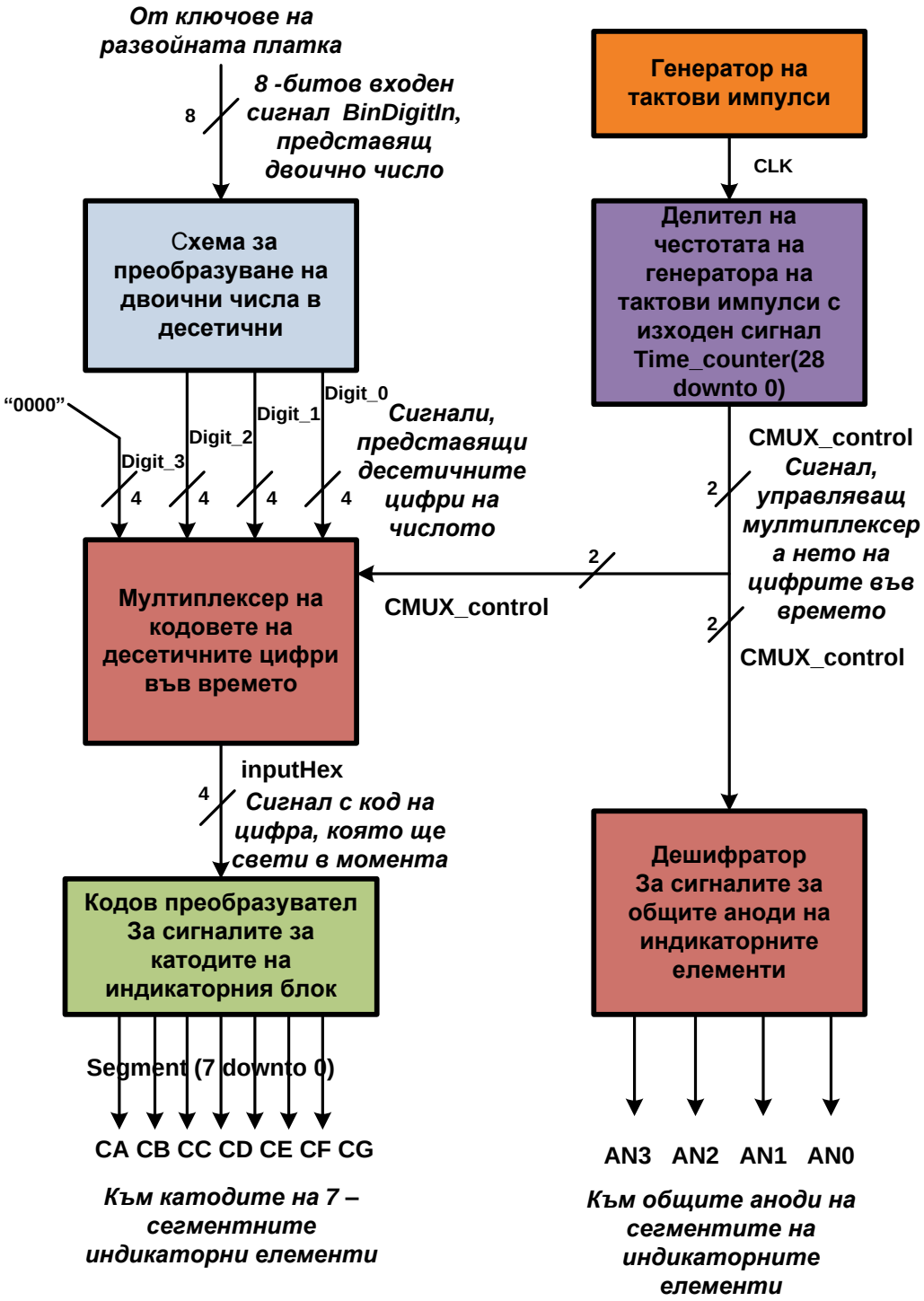
A2 A1 A0	D0	D1	D2	D3	D4	D5	D6	D7
000	0	1	1	1	1	1	1	1
001	1	0	1	1	1	1	1	1
010	1	1	0	1	1	1	1	1
011	1	1	1	0	1	1	1	1
100	1	1	1	1	0	1	1	1
101	1	1	1	1	1	0	1	1
110	1	1	1	1	1	1	0	1
111	1	1	1	1	1	1	1	0

Едно много разпространено приложение на мултиплексерите и дешифраторите е за осъществяване на адресирането и четенето на данните от блоковете с компютърни памети. По-долу са показани схеми на така нареченото едномерно и двумерно дешифриране (декодиране) на адресите в паметта, за да се управлява достъпът до думите на паметта. Дешифраторите избират реда от блока с памет, на който се намира думата, чиито адрес е зададен на входовете им.



3. Пример за използване на мултиплексер, дешифратор и кодов преобразувател в реално прилагано цифрово устройство. (Схемата се проектира, имплементира върху чип и се изследва на лабораторно упражнение по дисциплината).

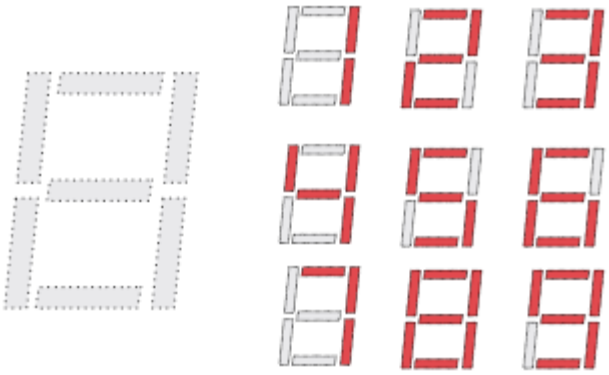
Коментираното по-долу цифрово устройство управлява извеждането на цифрова информация върху блок от 7 – сегментни индикаторни елементи на развойна платка. Следва блокова схема на устройството :



Устройството чете двоични числа от ключовете на развойната платка (чрез входен сигнал BinDigitIn), преобразува ги в десетични числа и извежда динамично десетичните им цифри върху 7 – сегментните индикаторни елементи на платката чрез управляващите сигнали за анодите AN3, AN2, ... и катодите CA, CB, CC, ... на индикаторните елементи, както е показано на горната схема.

Проектираното устройство изобразява десетичните цифри на резултата върху 7 – сегментните индикаторни елементи на развойната платка, като преобразува двоичните кодове на тези десетични цифри в кодове, необходими за управлението на съответните конфигурации от сегменти (светодиоди). ***Така например, двоичният код "0011", който представя десетичната цифра 3, трябва да бъде преобразуван в кода "0000110", за да укаже (чрез нулите в него) кои сегменти от 7- сегментния елемент ще трябва да светят, за да се изобрази образът на 3 чрез тях.***

***Горното преобразуване се извършва от цифров възел* – декодер.**

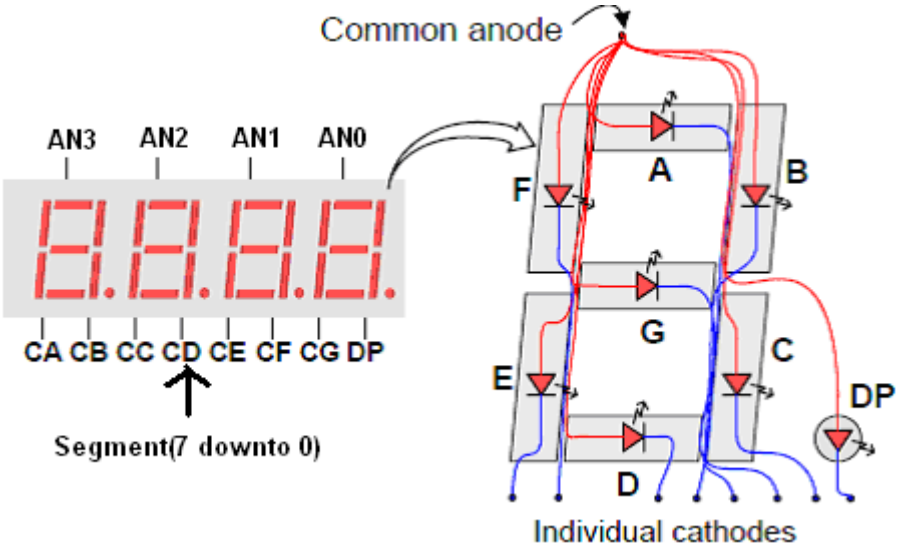


VHDL – текстът, който описва комбинаторния ***цифров възел - декодер***, извършващ това преобразуване е :

```
process (inputHex)
begin
case inputHex is
                    when "0001" => Segment <= "1001111";   --1
                    when "0010" => Segment <= "0010010";   --2
                    when "0011" => Segment <= "0000110";   --3
                    when "0100" => Segment <= "1001100";   --4
                    when "0101" => Segment <= "0100100";   --5
                    when "0110" => Segment <= "0100000";   --6
                    when "0111" => Segment <= "0001111";   --7
                    when "1000" => Segment <= "0000000";   --8
                    when "1001" => Segment <= "0000100";   --9
                    when others => Segment <= "0000001";   --0
            end case;

    end process;
CA <=  segment(6);
 CB <=  segment(5);
 CC <=  segment(4);
```

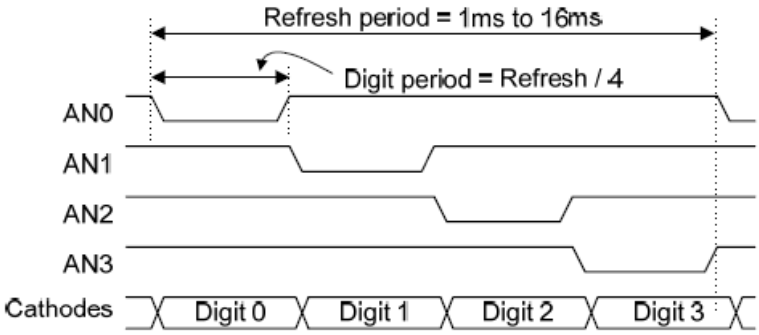
```
CD <= segment(3);
CE <= segment(2);
CF <= segment(1);
CG <= segment(0);
```

Четирите седемсегментни индикаторни елементи на развойната платка имат общи управляващи сигнали за катодите на сегментите: CA CB CC CD CE CF CG DP и индивидуални управляващи сигнали за анодите им: **AN3, AN2, AN1, AN0.**



Поради общите сигнали за катодите, налага се проектираното устройство във всеки момент да подава сигналите с кодовете на цифрите само към един индикаторен елемент и да мултиплексира (сменя) във времето тези сигнали за четирите елемента и то с такава честота, че да не се забелязва примигването им.

Устройството включва в структурата си комбинационни схеми – ***мултиплексер*** и ***дешифратор***, за да подава сигналите към управляващите входове на индикаторните елементи. На схемата по-горе и времедиаграмите по-долу сигналите ANi (към общите аноди) се генерират (изменят във времето) **от дешифратора**, а сигналите с кодовете, изобразяващи цифрите digit_i (към катодите)- CA, CB, CC, се генерират **от мултиплексера**. Новите стойности и на двата вида сигнали се подават синхронно – в едни и същи моменти от времето, траят еднакви интервали от време и ***определят цифрата, която ще свети в момента***.

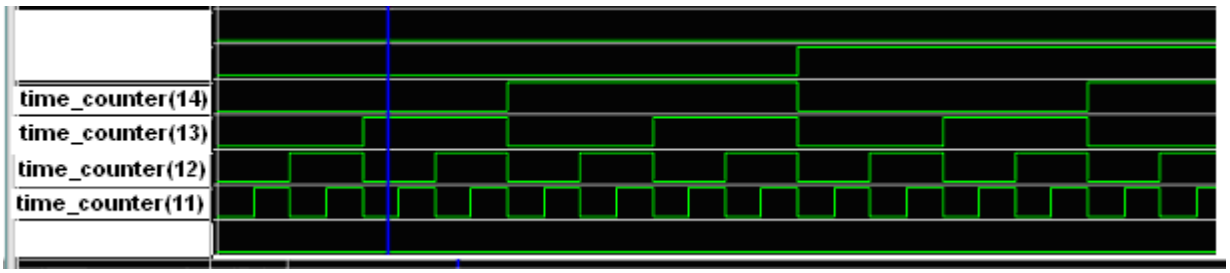


За да се осигури това, и мултиплексерът, и дешифраторът се управляват от един и същ сигнал – CMUX_control, генериран от делител на честотата на генератор на тактови импулси.

VHDL – описанието за това, как функционира **делителят на честота с изходен сигнал time_counter** и как **управляващият сигнал CMUX_control** приема от него стойностите си, е следното:

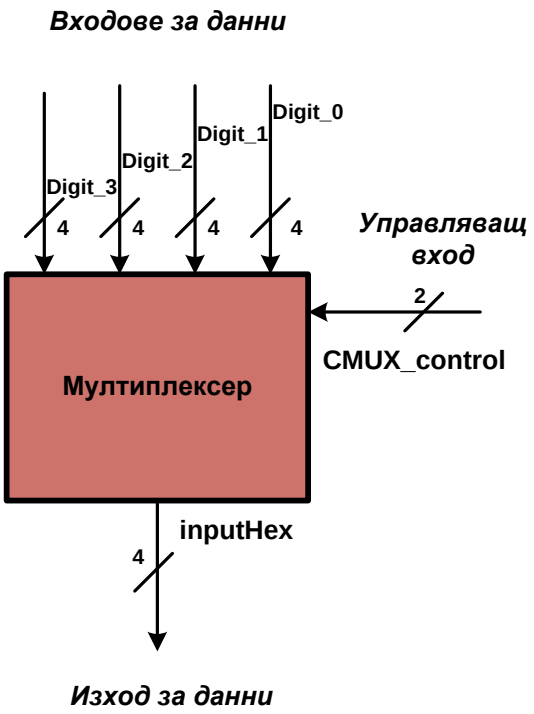
```
process (CLK)
 begin
        if CLK = '1' and CLK'event then
            time_counter <= time_counter +1;
        end if;
end process;

CMUX_control <= time_counter(13 downto 12);
```

Сигналът **CMUX_control (който е двубитов), приема от изходите на делителя на честота time_counter(13)и time_counter(12) последователно във времето стойностите: “00”, “01”, “10”, “11”.** Например на времедиаграмите за функционирането на time_counter, дадени по-долу, в момента на маркера **time_counter(13) = ‘1’, time_counter(12) = ‘0’ и следователно CMUX_control = “10”.**



Мултиплексерът в схемата на устройството осигурява последователното подаване във времето на кодовете (digit_i), изписващи образите на цифрите ***към катодите на 7 – сегментните индикаторни елементи.***

Мултиплексерът е с четири входа , един изход за данни (4:1) и един управляващ вход и както всеки стандартен мултиплексер функционира така: във всеки момент изходът му приема стойността на сигнала от този от входовете за данни, чиито номер е указан върху управляващия вход:



Следващата таблица на истинност показва функционирането на мултиплексера:

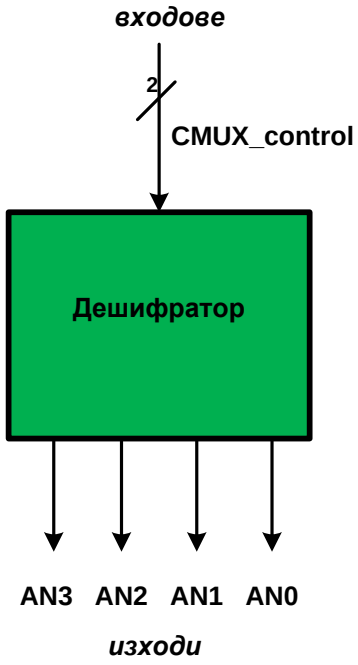
CMUX_control (управляващ вход)	inputHex (изход)
00	Digit_0 (вход за данни No 0)
01	Digit_1(вход за данни No 1)
10	Digit_2(вход за данни No 2)
11	Digit_3(вход за данни No 3)

VHDL – описанието на мултиплексера е:

```
process (CMUX_control, digit_3, digit_2, digit_1, digit_0)
begin
  case CMUX_control is
    when "00" => inputHex <= digit_0(3 downto 0);
    when "01" => inputHex <= digit_1(3 downto 0);
    when "10" => inputHex <= digit_2(3 downto 0);
    when "11" => inputHex <= digit_3(3 downto 0);
    when others => inputHex <= digit_0(3 downto 0);
  end case;
end process;
```

Дешифраторът активира (в стойност 0) в последователни интервали от време анодите на индикаторните елементи AN0, AN1, AN2, AN3. Той има два еднобитови

входа (CMUX_control(1), CMUX_control(0)) и следователно четири еднобитови изхода (AN0, AN1, AN2, AN3):



Като всеки стандартен дешифратор, той функционира така: във всеки момент е активен (в ‘0’) само един от изходите – този, чиито номер е указан на входовете на дешифратора. Останалите изходи са неактивни (в ‘1’). Следващата таблица за истинност показва функционирането на дешифратора:

CMUX_control (управляващ вход)	AN0	AN1	AN2	AN3
00	0	1	1	1
01	1	0	1	1
10	1	1	0	1
11	1	1	1	0

VHDL – описанието на дешифратора е:

```
process (CMUX_control)
begin
  case CMUX_control is
    when "00" => AN0 <= '0'; AN1 <= '1';AN2 <= '1';AN3 <= '1';
    when "01" => AN0 <= '1'; AN1 <= '0';AN2 <= '1';AN3 <= '1';
    when "10" => AN0 <= '1'; AN1 <= '1';AN2 <= '0';AN3 <= '1';
    when "11" => AN0 <= '1'; AN1 <= '1';AN2 <= '1';AN3 <= '0';
    when others => AN0 <= '0'; AN1 <= '1';AN2 <= '1';AN3 <= '1';
  end case;
end process;
```