# 6. Standardni kombinacijski moduli (1)

# Sadržaj predavanja

- **kombinacijski moduli**
- dekoder
- multipleksor

# Kombinacijski moduli

- dekompozicija sustava:
  - identifikacija češće korištenih podsustava/sklopova ~ *moduli*
  - *kombinacijski* moduli:
    - izlazi = f(ulazi)
    - ostvarivanje *složenije* (od I, ILI, NE) funkcije
  - tipične izvedbe:
    - MSI i LSI
    - čipovi/dijelovi čipa

# Kombinacijski moduli

- općenita podjela kombinacijskih modula:
  - specijalni:
    - ciljano projektirani za zadani sustav
    - *optimalna* izvedba
  - standardni:
    - "opće namjene" (engl. general purpose)
    - proizvodnja u velikim serijama
      ~ niska cijena
    - široko korištene funkcije
  - univerzalni
    ~ ostvarivanje proizvoljne Booleove funkcije

# Kombinacijski moduli

- ostvarivanje složenijih kombinacijskih funkcija:
  - dekoderi i pretvornici koda
  - sklopovi za odabir podataka
  - koderi
  - komparatori
  - aritmetički moduli

# Sadržaj predavanja

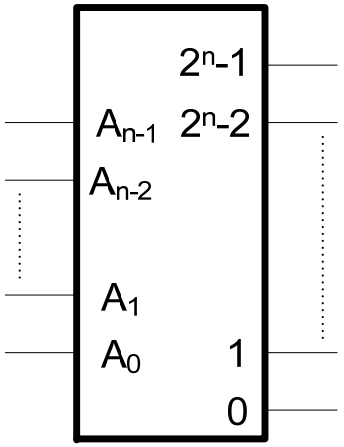
- kombinacijski moduli
- **dekoder**
  - **binarni dekoder**
  - **dekodersko stablo**
  - **ostvarivanje Booleovih funkcija dekoderom**
  - **dekadski dekoder**
  - **demultipleksor**
- multipleksor

# Dekoder

- funkcija *dekodiranja*
  ~ identificiranje kodne riječi nekog koda
- *dekoder*
  ~ aktivan *samo jedan* izlaz,
  onaj koji "odgovara" narinutoj kodnoj riječi

  *n* ulaza $\rightarrow$ $2^n$ izlaza

$$Z_i = \begin{cases} 1 & \text{za} \quad i = A_{n-1} \ldots A_0 \\ 0 & \text{za} \quad i \neq A_{n-1} \ldots A_0 \end{cases}$$




- tipična oznaka:

  <broj adresa>/$2^{\text{<broj adresa>}}$

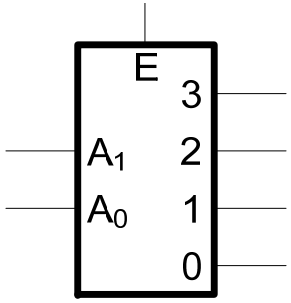
# Dekoder

- podjela dekodera:
  - *binarni* dekoderi:
    - n = 2, 3, 4, ... ulaza $\rightarrow$ "1-od-$2^n$" izlaza
    - usmjeravanje informacije na jedan od izlaza
  - *dekadski* dekoderi:
    - n = 4 ulaza $\rightarrow$ "1-od-10" izlaza
    - dekodiranje binarnih kodova za prikaz dekadskih znamenki npr. BCD, XS-3

# Binarni dekoder

*Primjer*: binarni dekoder 2/4

$$\overline{A}_1\overline{A}_0 \equiv 00_2 = 0_{10}$$
$$\overline{A}_1 A_0 \equiv 01_2 = 1_{10}$$
$$A_1\overline{A}_0 \equiv 10_2 = 2_{10}$$
$$A_1 A_0 \equiv 11_2 = 3_{10}$$



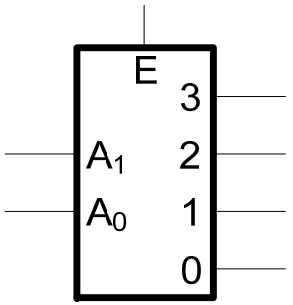

- upravljanje sklopom
  ~ *ulaz za omogućavanje*
  E (engl. enable)

| E | $A_1$ | $A_0$ | "0" | "1" | "2" | "3" |
|---|---|---|---|---|---|---|
| 0 | X | X | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

# Binarni dekoder

- VHDL *ponašajni* model binarnog dekodera 2/4

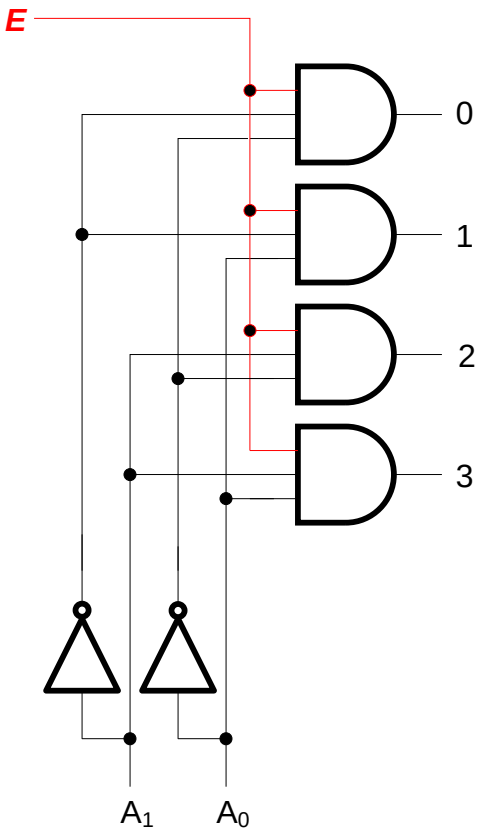
```
library ieee;
use ieee.std_logic_1164.all;

entity dekoder24e is
  port (d: out std_logic_vector(0 to 3);
        a: in std_logic_vector(1 downto 0);
        e: in std_logic);
end dekoder24e;

architecture ponasajna of dekoder24e is
begin
  process (a,e)
  begin
    if(e = '0')
      then d <= "0000";
      else case a is
             when "00"   => d <= "1000";
             when "01"   => d <= "0100";
             when "10"   => d <= "0010";
             when "11"   => d <= "0001";
             when others => d <= "0000";
           end case;
    end if;
  end process;
end ponasajna;
```



# Binarni dekoder

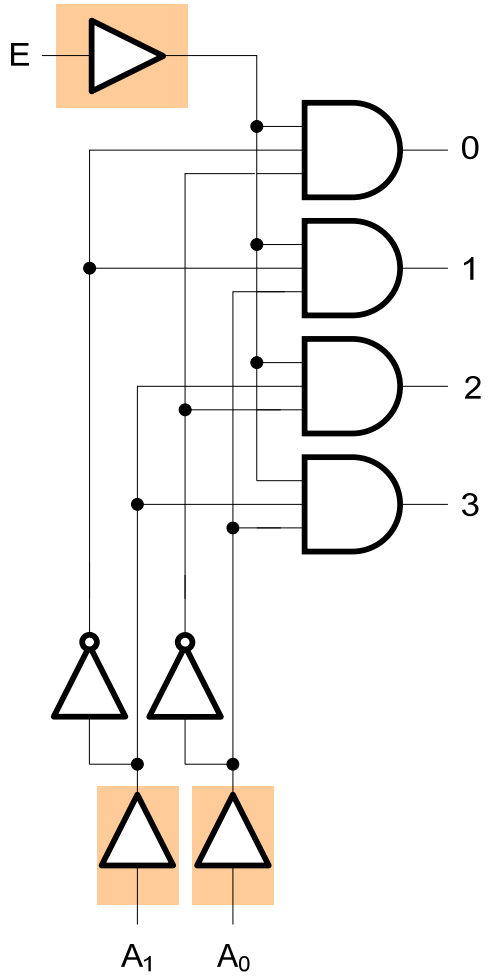
- *struktura* dekodera 2/4:

$$\overline{A}_1\overline{A}_0 \equiv 00_2 = 0_{10}$$
$$\overline{A}_1 A_0 \equiv 01_2 = 1_{10}$$
$$A_1\overline{A}_0 \equiv 10_2 = 2_{10}$$
$$A_1 A_0 \equiv 11_2 = 3_{10}$$



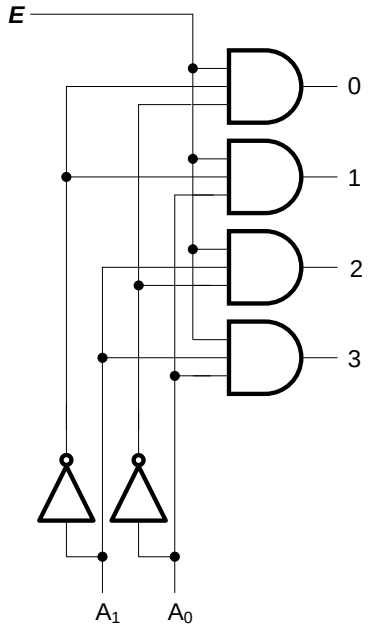
# Binarni dekoder

- *stvarna* izvedba:
  - ograničiti opterećenje ulazima
  - odvojni sklopovi ~ *jedinično* opterećenje
  - funkcijski ekvivalentna izvedba



# Binarni dekoder

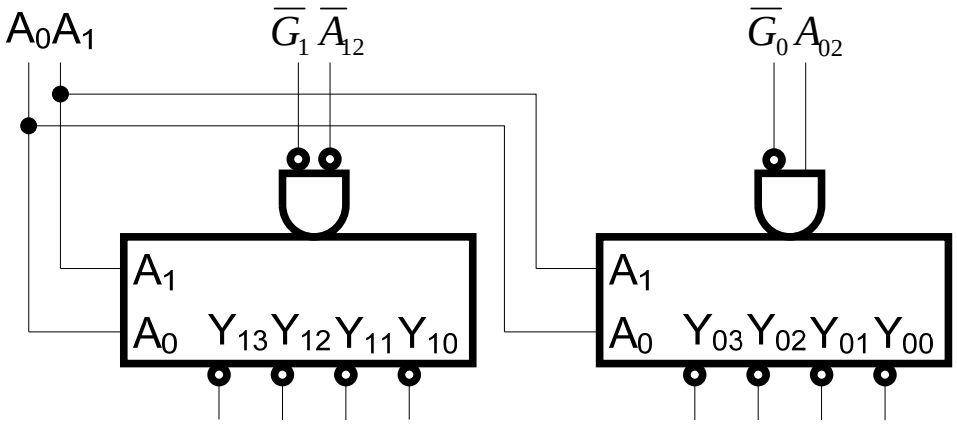
- VHDL *strukturni* model binarnog dekodera 2/4




```
library ieee;
use ieee.std_logic_1164.all;

entity dekoder24e is
  port (d: out std_logic_vector(0 to 3);
        a: in std_logic_vector(1 downto 0);
        e: in std_logic);
end dekoder24e;

architecture strukturna of dekoder24e is
  signal a1_komplement, a0_komplement: std_logic;
  begin
    sklop1: entity work.sklopNOT  port map (a(1),a1_komplement);
    sklop2: entity work.sklopNOT  port map (a(0),a0_komplement);
    sklop3: entity work.sklopAND3 port map (e,a1_komplement,a0_komplement,d(0));
    sklop4: entity work.sklopAND3 port map (e,a1_komplement,a(0),d(1));
    sklop5: entity work.sklopAND3 port map (e,a(1),a0_komplement,d(2));
    sklop6: entity work.sklopAND3 port map (e,a(1),a(0),d(3));
end strukturna;
```

# Binarni dekoder

- izvedbe dekodera
  ~ MSI modul; npr. 74155 (dvostruki četveroizlazni)
  - zajedničke adrese $A_1$, $A_0$
  - oznake:
    $\overline{G}_1, \overline{G}_0$: ulazi za omogućavanje
  - dekoder 3-bitnih riječi: $A_{02} = \overline{A}_{12} = A_2$



# Binarni dekoder

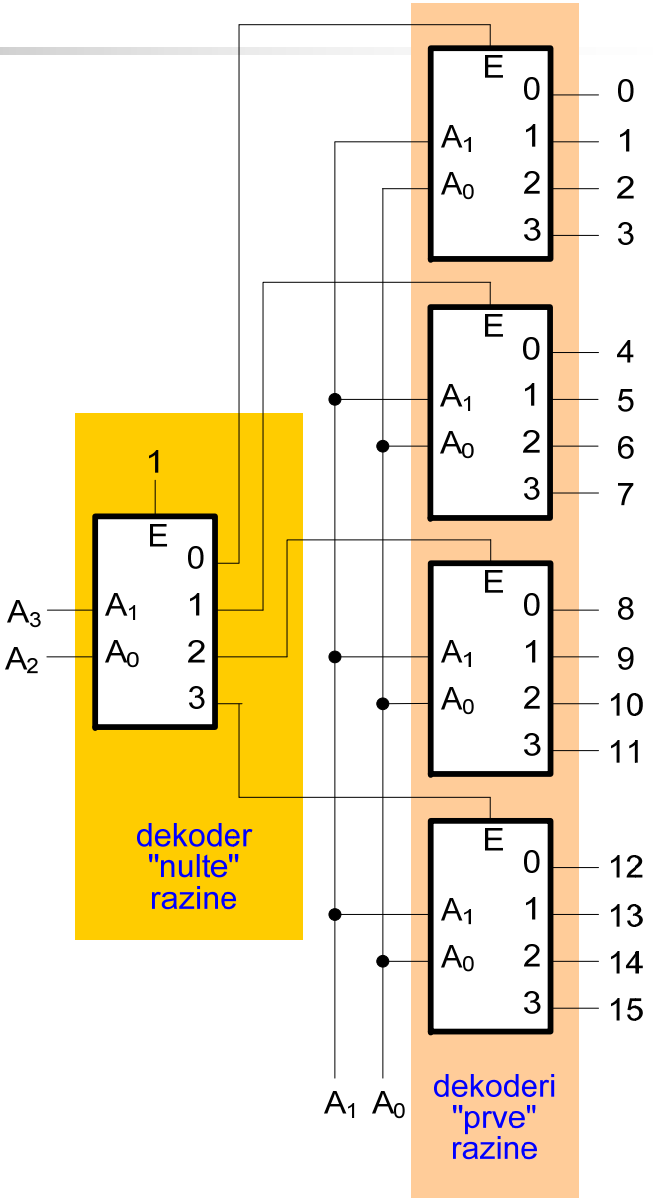
- dekoderi s većim (n > 16) brojem izlaza:
  - izvedba jednim modulom *nepraktična*
    ~ presloženi MSI modul
  - radije *kaskadiranje*
    ~ *dekodersko stablo* (engl. decoder tree):
    - općenita metoda
    - vrijedi za proizvoljno složeni modul;
      npr. izvedba dekodera sklopovima I

# Binarni dekoder

*Primjer*: dekoder 4/16
kao dekodersko stablo

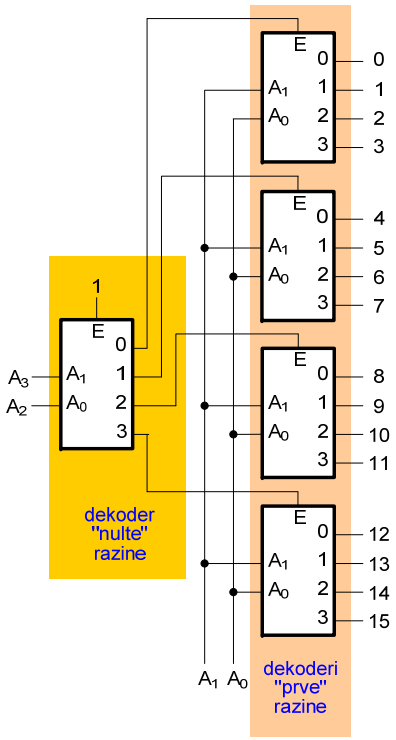
| $A_3$ | $A_2$ | $A_1$ | $A_0$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $D_0^1$ |
| | | 0 | 1 | |
| | | 1 | 0 | |
| | | 1 | 1 | |
| 0 | 1 | 0 | 0 | $D_1^1$ |
| | | 0 | 1 | |
| | | 1 | 0 | |
| | | 1 | 1 | |
| 1 | 0 | 0 | 0 | $D_2^1$ |
| | | 0 | 1 | |
| | | 1 | 0 | |
| | | 1 | 1 | |
| 1 | 1 | 0 | 0 | $D_3^1$ |
| | | 0 | 1 | |
| | | 1 | 0 | |
| | | 1 | 1 | |

$D^0$



# Binarni dekoder

- VHDL *strukturni* model dekoderskog stabla 4/16




```
library ieee;
use ieee.std_logic_1164.all;

entity dekoder416e is
  port (d: out std_logic_vector(0 to 15);
        a: in  std_logic_vector(3 downto 0);
        e: in  std_logic);
end dekoder416e;

architecture strukturna of dekoder416e is
  signal e_tmp: std_logic_vector(0 to 3);
  begin
    sklop1: entity work.dekoder24e port map (e_tmp, a(3 downto 2), e);
    sklop2: entity work.dekoder24e port map (d(0 to 3), a(1 downto 0), e_tmp(0));
    sklop3: entity work.dekoder24e port map (d(4 to 7), a(1 downto 0), e_tmp(1));
    sklop4: entity work.dekoder24e port map (d(8 to 11), a(1 downto 0), e_tmp(2));
    sklop5: entity work.dekoder24e port map (d(12 to 15), a(1 downto 0), e_tmp(3));
end strukturna;
```

# Ostvarivanje funkcija dekoderom

- zapažanje:
  - izlazi dekodera = *potpuno dekodirane* kodne riječi ~ *mintermi*

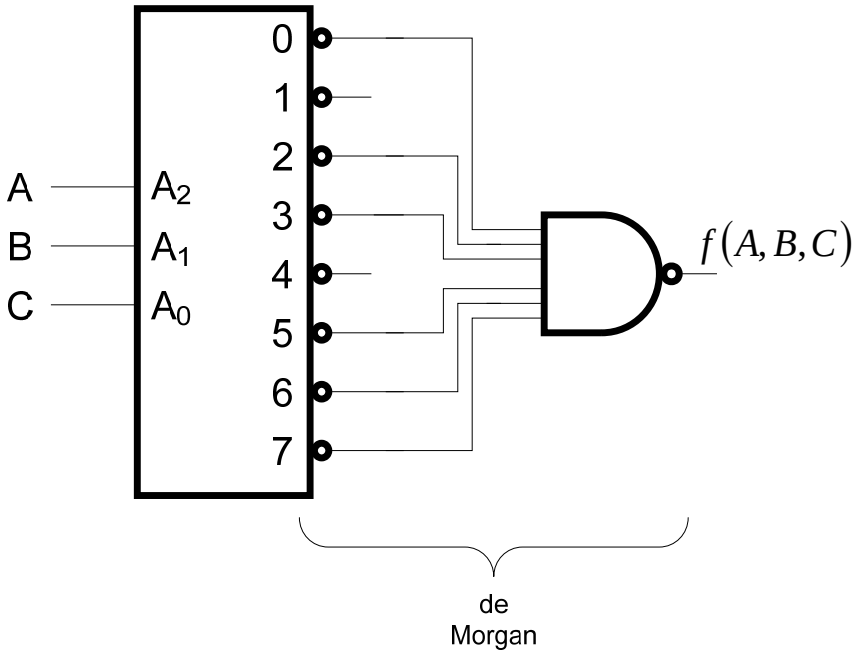
$$\overline{A}_1\overline{A}_0 \equiv 00_2 = 0_{10}$$
$$\overline{A}_1 A_0 \equiv 01_2 = 1_{10}$$
$$A_1\overline{A}_0 \equiv 10_2 = 2_{10}$$
$$A_1 A_0 \equiv 11_2 = 3_{10}$$

- *ostvarivanje logičkih funkcija* dekoderom:
  - funkcija u kanonskom *disjunktivnom* obliku
  - "pokupiti" (funkcija ILI) izlaze koji odgovaraju mintermima zastupljenim u definiciji funkcije

# Ostvarivanje funkcija dekoderom

*Primjer*: $f(A,B,C)=\sum m(0,2,3,5,6,7)$

- f(A, B, C) $\rightarrow$ dekoder 3/8
- uočiti: ILI°I = (NI°NE)°I



# Dekadski dekoder

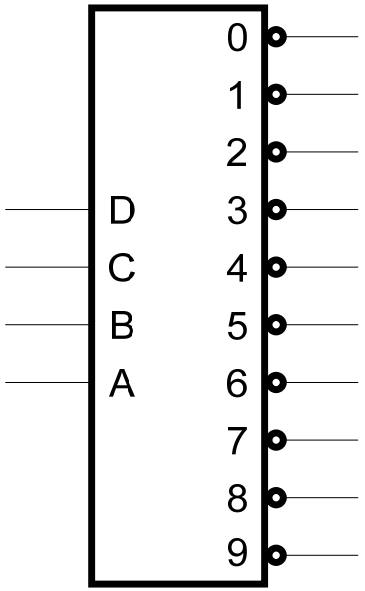
- dekodiranje koda s $N < 2^n$
  ~ "nepotpuno" dekodiranje
  - broj izlaza < $2^{\text{broj ulaza}}$
  - tipični slučaj
    ~ dekodiranje binarno kodiranih dekadskih znamenki
    - BCD
    - XS-3
    - 2421
    - itd.

# Dekadski dekoder

*Primjer*: BCD-dekadski dekoder 7442

- oznake: D ~ najviša težina ($2^3$)
- izlazi invertirani:
  zgodno kod kombiniranja sklopova
- uzorci 0 i 1 koji nisu kodne riječi:
  - ne prepoznaju se!
  - minimizacija sklopa
- varijante:

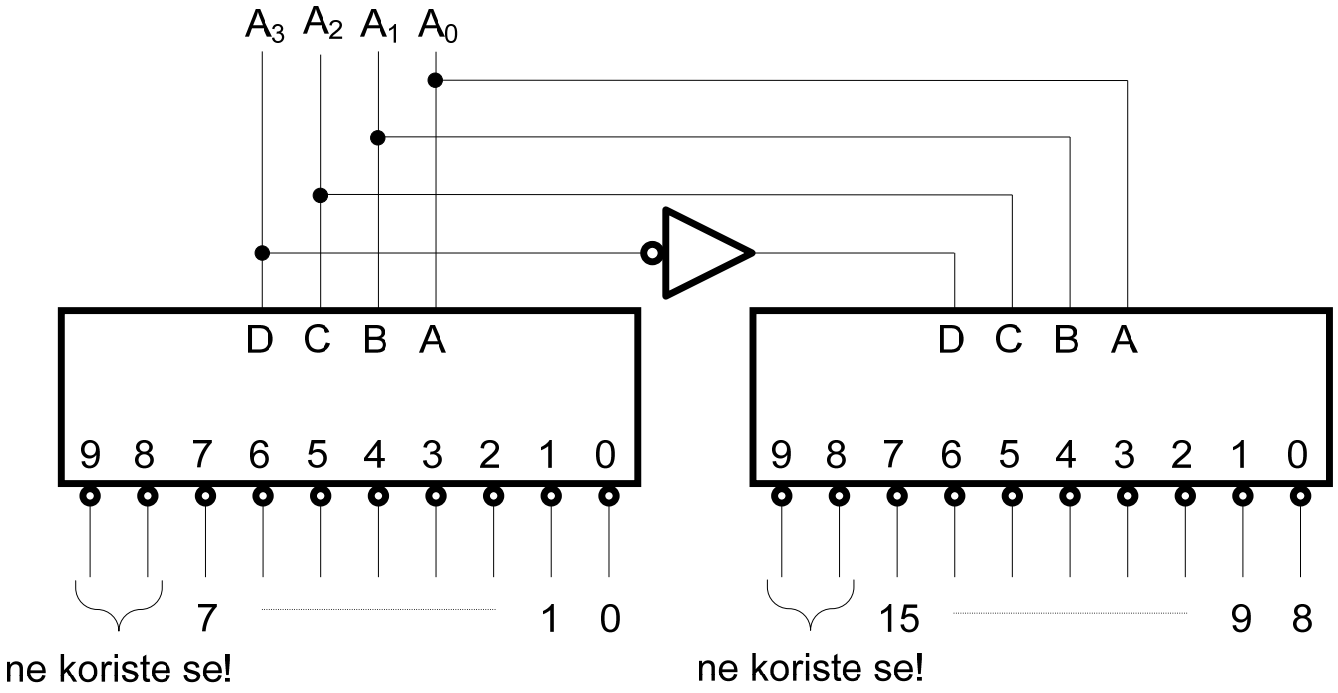
  7443: XS-3
  7444: XS-3 Gray



# Dekadski dekoder

*Primjer*: izvedba dekodera "1-od-16" od dva 7442 :

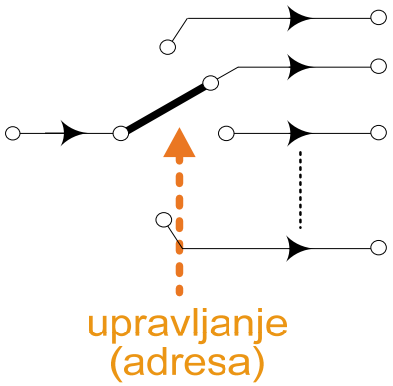
- D : odabir jednog od dva dekodera 1/8

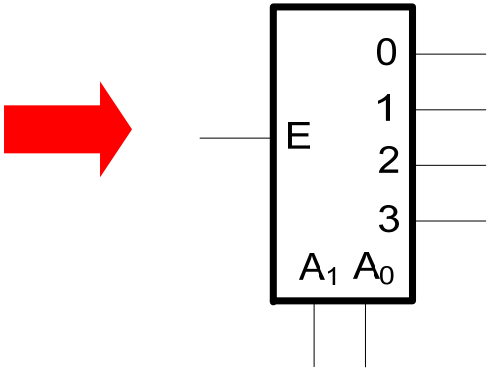


# Demultipleksor

- demultipleksor:
  - ulaz za omogućavanje dekodera
    ~ funkcija *ulaza za podatke*
  - "usmjeravanje"/"raspodjela" ulaza na odabrani izlaz
    ~ "demultipleksiranje"

| $A_1$ | $A_0$ | "0" | "1" | "2" | "3" |
|---|---|---|---|---|---|
| 0 | 0 | E | 0 | 0 | 0 |
| 0 | 1 | 0 | E | 0 | 0 |
| 1 | 0 | 0 | 0 | E | 0 |
| 1 | 1 | 0 | 0 | 0 | E |



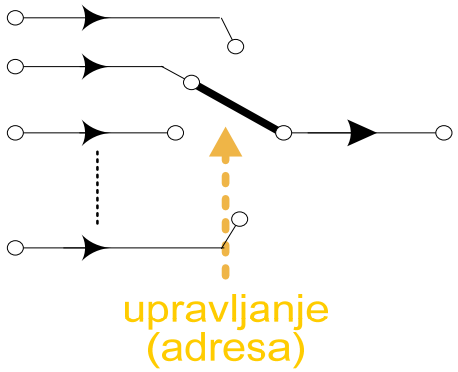
# Sadržaj predavanja

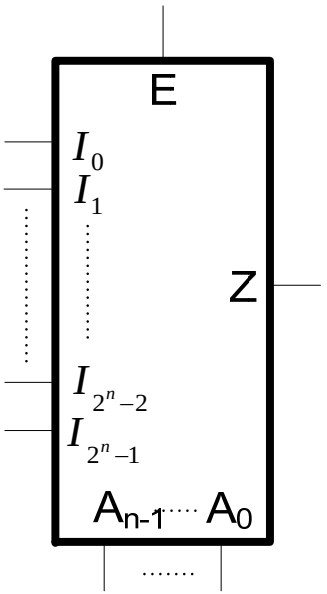
- kombinacijski moduli
- dekoder
- **multipleksor**
  - **funkcionalnost multipleksora**
  - **multipleksorsko stablo**

# Multipleksor

- multipleksor:
  - odabir podataka
    ~ "multipleksiranje"
  - funkcija *upravljane preklopke*:

$$Z = \begin{cases} I_i & \text{za} \quad i = A_{n-1} \ldots A_0 \quad (\wedge E = 1) \\ 0 & \text{za} \quad E = 0 \end{cases}$$



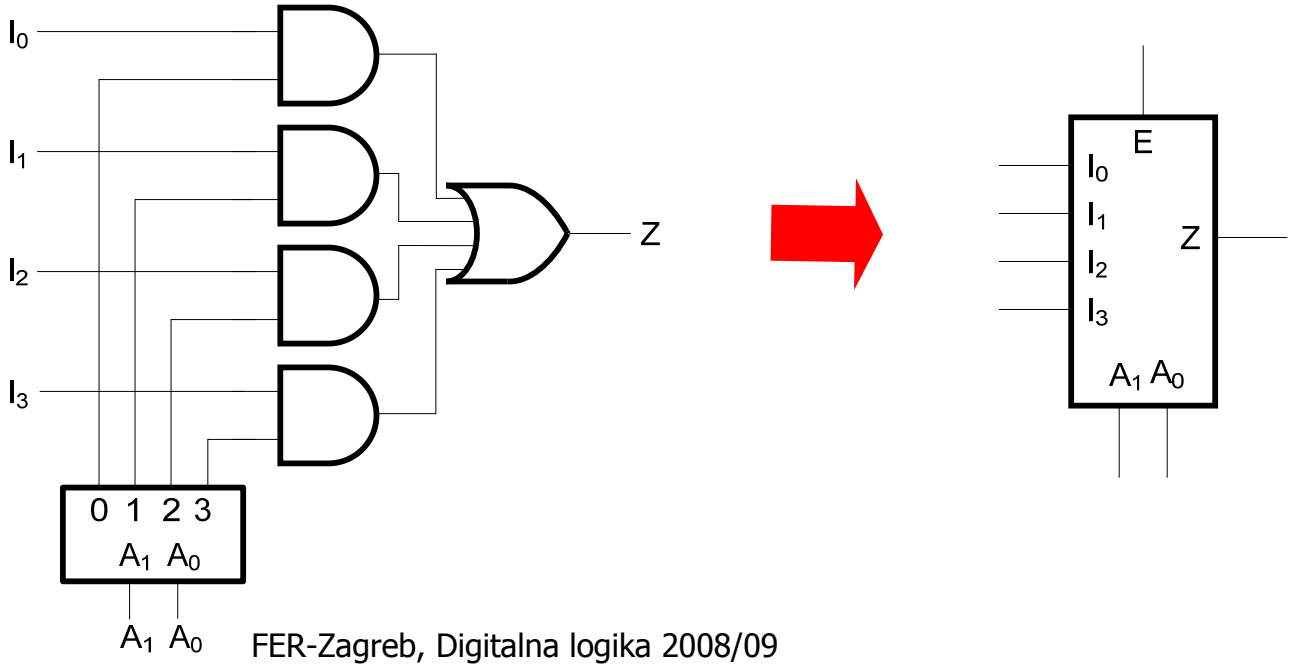



# Multipleksor

*Primjer*: multipleksor 4/1

- adrese se dekodiraju
- izlazi iz dekodera *koincidiraju* s ulazima $I_i$
  ~ propuštaju *samo jedan* od ulaza
  na izlazni ILI sklop

| E | $A_1$ | $A_0$ | Z |
|---|---|---|---|
| 0 | X | X | 0 |
| 1 | 0 | 0 | $I_0$ |
| 1 | 0 | 1 | $I_1$ |
| 1 | 1 | 0 | $I_2$ |
| 1 | 1 | 1 | $I_3$ |



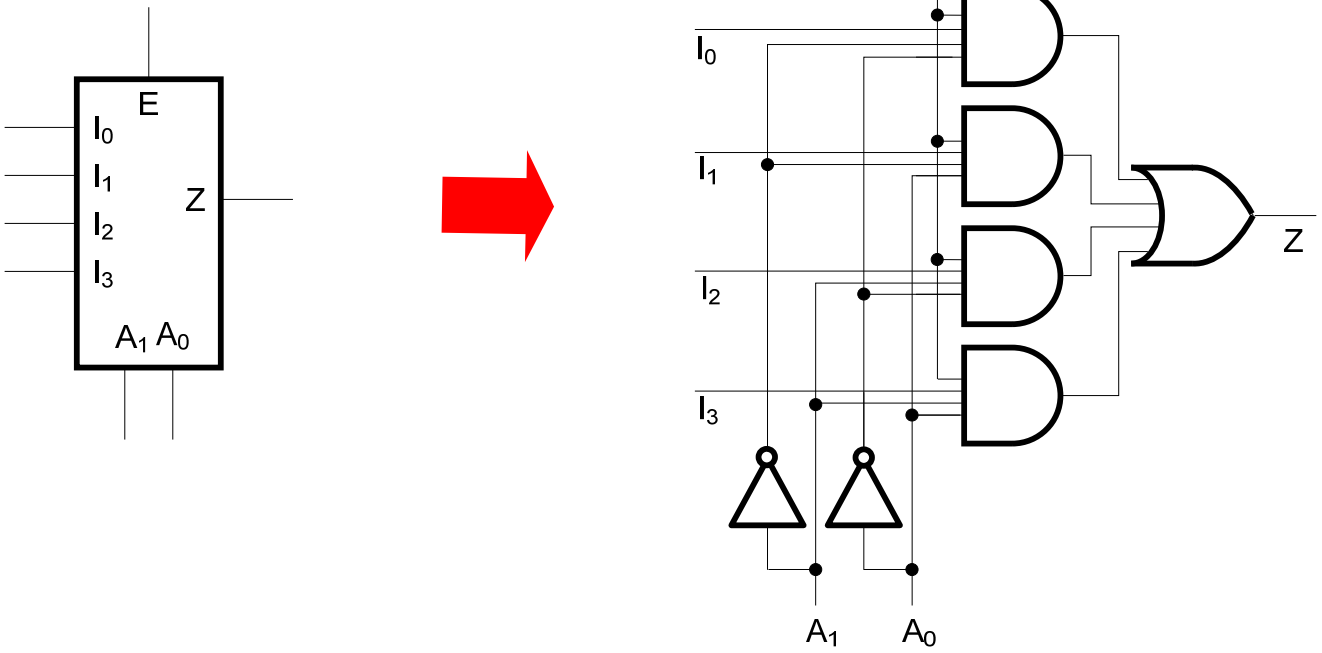
# Multipleksor

- *VHDL ponašajni model* multipleksora 4/1

```
entity mux41e is
  port (i: in std_logic_vector(0 to 3);
        a: in std_logic_vector(1 downto 0);
        e: in std_logic;
        z: out std_logic);
end mux41e;

architecture ponasajna of mux41e is
begin
  process (i,a,e)
  begin
    if(e = '0')
      then z <= '0';
      else
        case a is
          when "00" =>   z <= i(0);
          when "01" =>   z <= i(1);
          when "10" =>   z <= i(2);
          when "11" =>   z <= i(3);
          when others => z <= '0';
        end case;
    end if;
  end process;
end ponasajna;
```

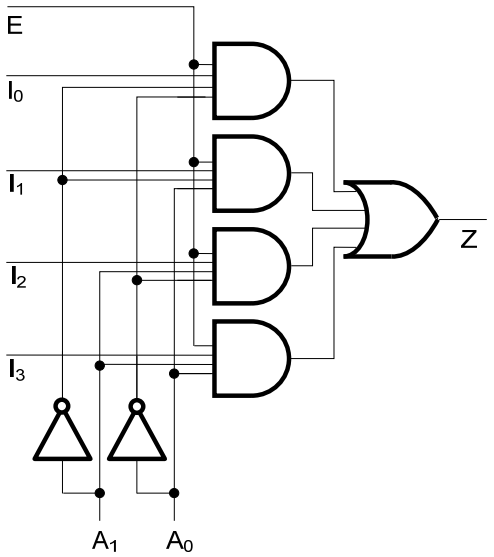
# Multipleksor

- funkcija I je *asocijativna*
  ~ "grupirati" I sklopove iz dekodera i koncidenciju



# Multipleksor

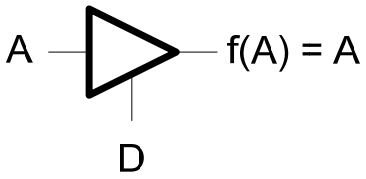
- *VHDL strukturni model* multipleksora 4/1



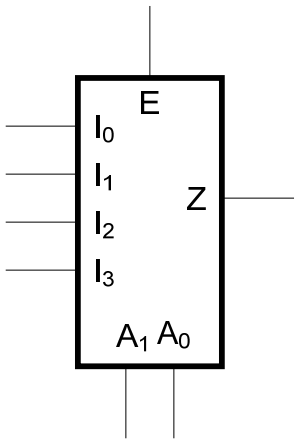

```
entity mux41e is
  port (i: in std_logic_vector(0 to 3);
        a: in std_logic_vector(1 downto 0);
        e: in std_logic;
        z: out std_logic);
end mux41e;

architecture strukturna of mux41e is
  signal a1_komplement, a0_komplement: std_logic;
  signal rez: std_logic_vector(0 to 3);
  begin
   sklop1: entity work.sklopNOT  port map (a(1),a1_komplement);
   sklop2: entity work.sklopNOT  port map (a(0),a0_komplement);
   sklop3: entity work.sklopAND4 port map (e,a1_komplement,a0_komplement,i(0),rez(0));
   sklop4: entity work.sklopAND4 port map (e,a1_komplement,a(0),i(1),rez(1));
   sklop5: entity work.sklopAND4 port map (e,a(1),a0_komplement,i(2),rez(2));
   sklop6: entity work.sklopAND4 port map (e,a(1),a(0),i(3),rez(3));
   sklop7: entity work.sklopOR4  port map (rez(0),rez(1),rez(2),rez(3),z);
end strukturna;
```

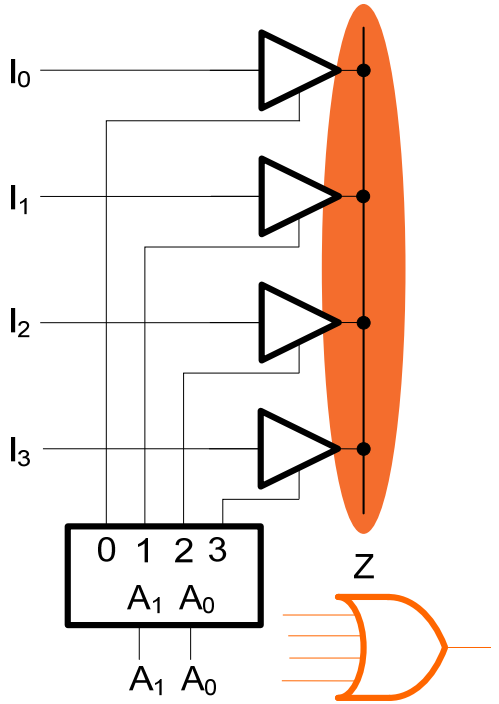
# Multipleksor

- izvedba funkcije multipleksiranja *na sabirničkoj liniji*:
  - izlazni ILI → "upravljani spojeni ILI"
    ~ *samo jedan* sklop definira vrijednost V/N
  - *sklopovi s tri stanja* upravljani izlazima iz dekodera
  - (upravljani) *odvojni sklop*



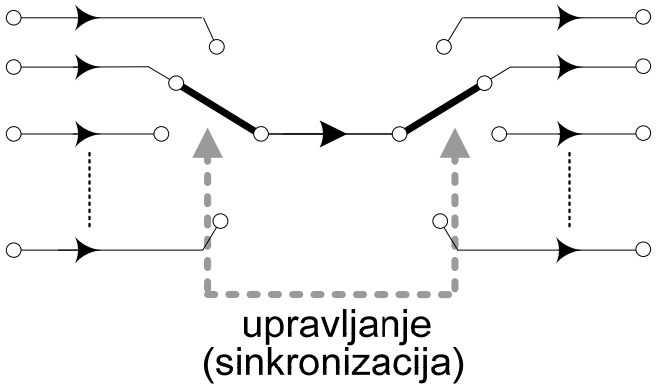

| D | A | f |
|---|---|---|
| 0 | X | Z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

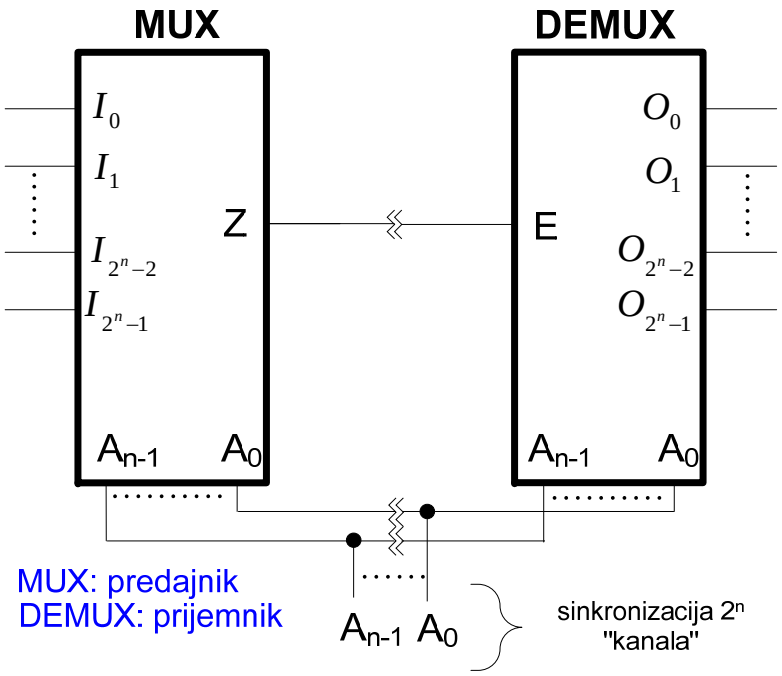


# Multipleksor

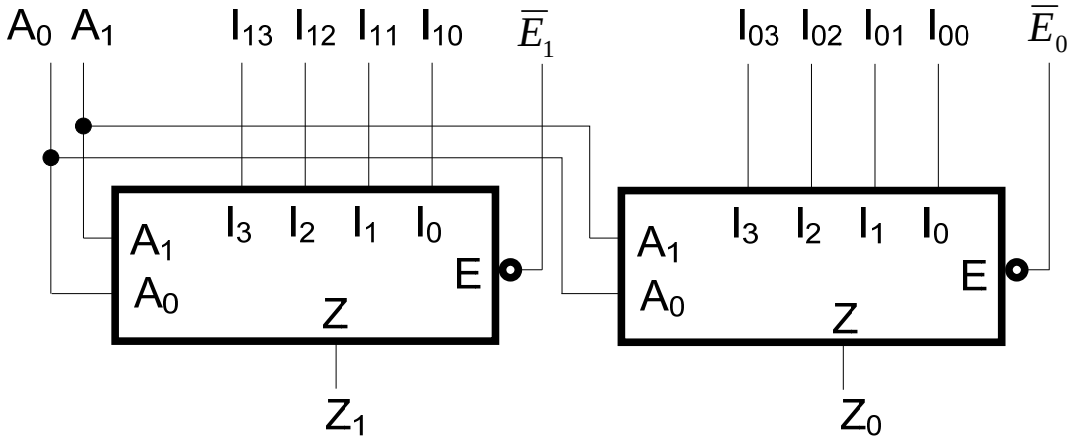
- funkcija multipleksiranja:
  ~ višestruko iskorištenje spojnih puteva
  - prijenos različitih podataka *istim* fizičkim spojnim putem
    ~ "više logičkih kanala preko jedne fizičke linije"
  - vremenska podjela (vremenski multipleks)



# Multipleksor

- izvedbe multipleksora
  ~ MSI modul; npr. 74153 (dvostruki četveroulazni)
  - zajedničke adrese $A_1$, $A_0$
  - *izdvojeni* ulazi za omogućavanje: $\overline{E_1}, \overline{E_0}$

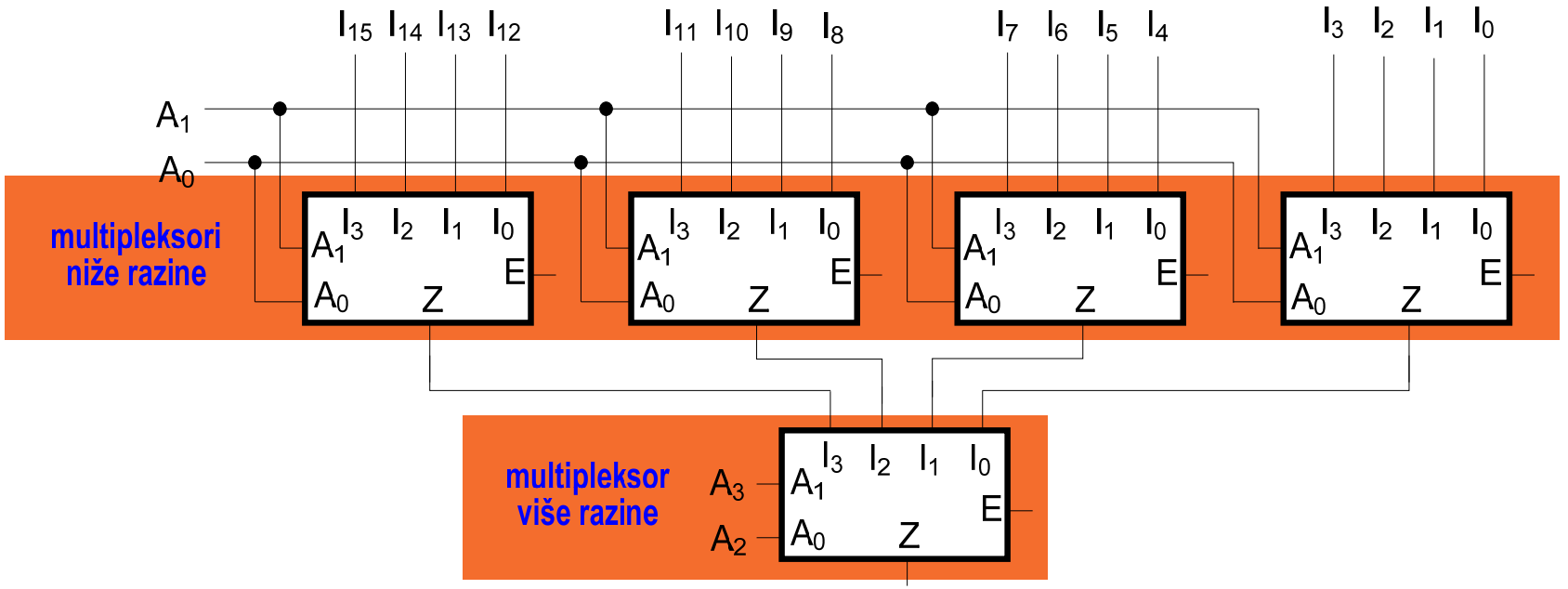


# Multipleksor

- multipleksori s većim (n > 16) brojem ulaza:
  - izvedba jednim modulom *nepraktična*
    ~ presloženi MSI modul
  - radije *kaskadiranje*
    ~ *multipleksorsko stablo* (engl. multiplexer tree)
  - izgradnja multipleksorskog stabla:
    - podjela tablice definicije funkcija u podtablice
      ~ ulazi u MUX više razine
    - varijable viših težina na MUX "više razine"

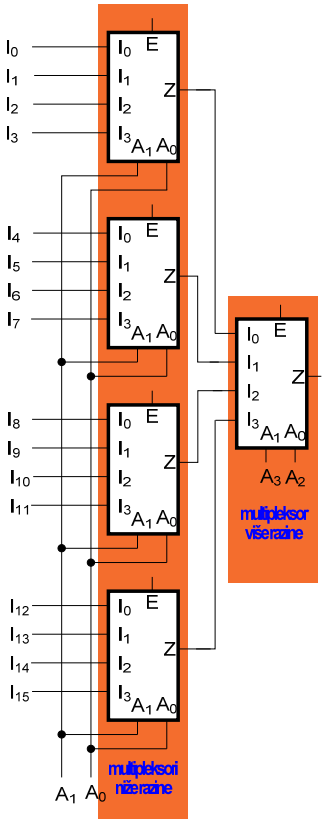
# Multipleksor

*Primjer*: multipleksor 16/1
kao multipleksorsko stablo



# Multipleksor

- VHDL *strukturni* model multipleksorskog stabla 16/1:




```
library ieee;
use ieee.std_logic_1164.all;

entity mux161e is
  port (i: in std_logic_vector(0 to 16);
        a: in std_logic_vector(3 downto 0);
        e: in std_logic;
        z: out std_logic);
end mux161e;

architecture strukturna of mux161e is
  signal rez: std_logic_vector(0 to 3);
  begin
    sklop1: entity work.mux41e port map (i(0 to 3),a(1 to 0),e,rez(0));
    sklop2: entity work.mux41e port map (i(4 to 7),a(1 to 0),e,rez(1));
    sklop3: entity work.mux41e port map (i(8 to 11),a(1 to 0),e,rez(2));
    sklop4: entity work.mux41e port map (i(12 to 15),a(1 to 0),e,rez(3));
    sklop5: entity work.mux41e port map (rez(0 to 3),a(3 to 2),e,z);
end strukturna;
```

# Literatura

U. Peruško, V. Glavinić: *Digitalni sustavi*, Poglavlje 7: Standardni kombinacijski moduli.

- kombinacijski moduli: str. 253-254
- dekoder: str. 254-259
- multipleksor (funcionalnost, multipleksorsko stablo): str. 260-264, 266-267

# Zadaci za vježbu (1)

U. Peruško, V. Glavinić: *Digitalni sustavi*, Poglavlje 7: Standardni kombinacijski moduli.

- dekoder: 7.1, 7.3-7.6

# Zadaci za vježbu (2)

M. Čupić: *Digitalna elektronika i digitalna logika. Zbirka riješenih zadataka*, Cjelina 5: Standardni kombinacijski moduli.

- dekoder:
  - riješeni zadaci: 5.2, 5.3, 5.9, 5.10, 5.11a, 5.12a
  - zadaci za vježbu: 6-12, 15, 19 (VHDL)
- multipleksor (funcionalnost, multipleksorsko stablo):
  - riješeni zadaci: 5.1
  - zadaci za vježbu: 1-5, 13-14