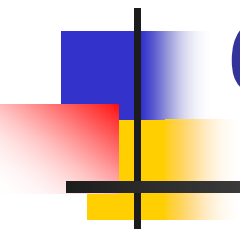


6. Standardni kombinacijski moduli (2)

Sadržaj predavanja

- **multipleksor**
  - **ostvarivanje Booleovih funkcija multipleksorom**
- prioritetni koder
- pretvornik koda
- komparator

Ostvarivanje funkcija multipleksorom

- ostvarivanje logičkih funkcija multipleksorom:
  - funkcija multipleksiranja:
    m_i: minterm predstavlja adresu

$$Z = \sum_{i=0}^{2^n-1} I_i \cdot m_i$$

  - definicija funkcije od n varijabli u kanonskom disjunktivnom obliku:

$$f(x_{n-1},\ldots,x_0) = \sum_{i=0}^{2^n-1} \alpha_i \cdot m_i$$

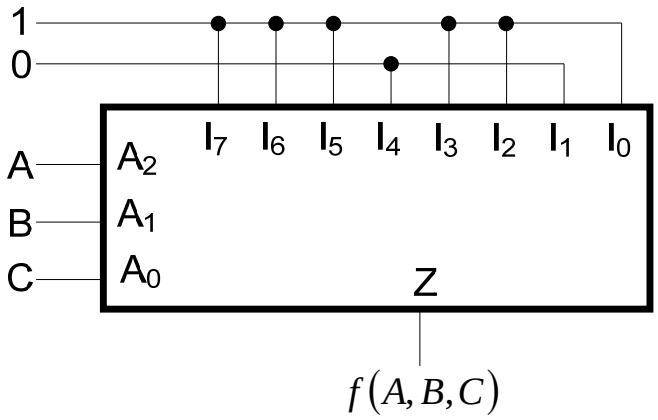
  - pridruživanje: $\forall A_i = x_i, I_i = \alpha_i \Rightarrow Z = f(x_{n-1},\ldots,x_0)$

Ostvarivanje funkcija multipleksorom

Primjer: ostvarivanje funkcije tri varijable

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

- "simulacija rada permanentne memorije (ROM)"
- neefikasno! ($\forall$ m_i $\exists$ I_i)



Ostvarivanje funkcija multipleksorom

- rješenje *trivijalnim rezidualnim funkcijama*
  ~ efikasnije rješenje:
  - za $f(x_{n-1},\ldots,x_0)$ MUX s n-1 adresa
    ~ 2^{n-1} "informacijskih" ulaza
  - na ulaze MUX dovoditi funkcije varijable *najmanje* težine:

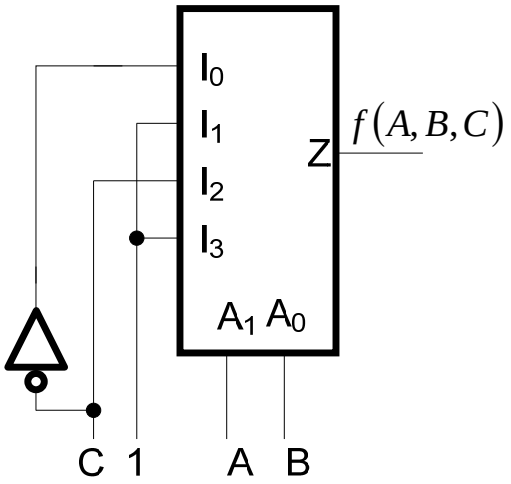
    $$\varphi(x_0)=\{0,1,x_0,\bar{x}_0\}$$

  - 2^{n-1} informacijskih ulaza
    ~ 2^{n-1} funkcija ostatka, *rezidualnih funkcija*
  - rezidualne funkcije od jedne varijable
    ~ *trivijalne* rezidualne funkcije

Ostvarivanje funkcija multipleksorom

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

A_1 A	A_0 B	C	ADRESIRANI ULAZ	f	
0	0	0	I_0	1	$\overline{C}$
0	0	1		0	
0	1	0	I_1	1	1
0	1	1		1	
1	0	0	I_2	0	C
1	0	1		1	
1	1	0	I_3	1	1
1	1	1		1	



Ostvarivanje funkcija multipleksorom

- rješenje *netrivijalnim* rezidualnim funkcijama:
  - "netrivijalne" rezidualne funkcije (>1 varijable)
    ~ obično presloženo rješenje
  - (također) kanonski oblik funkcije
    ~ *nema* minimizacije
  - pojednostavljivanje rješenja
    ~ odabir prikladnog pridruživanja varijabli adresnim ulazima

Ostvarivanje funkcija multipleksorom

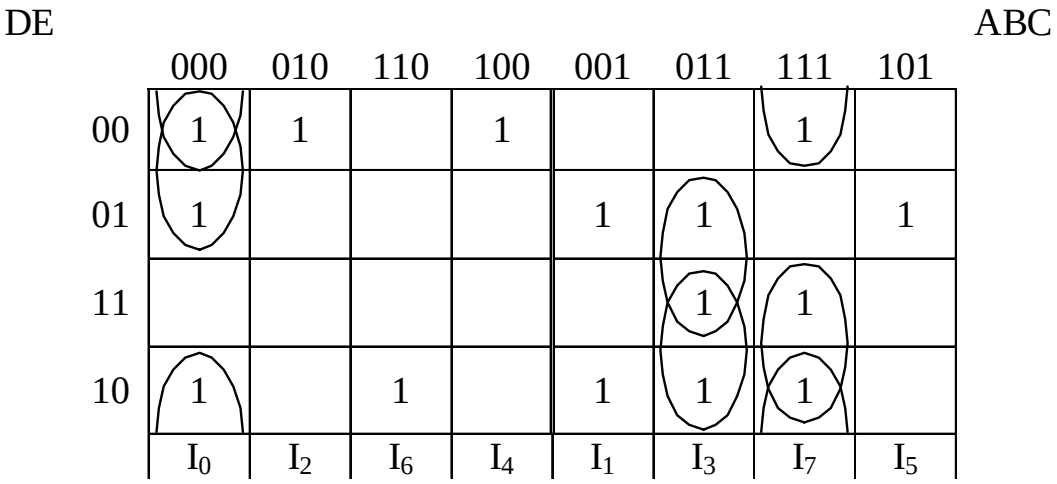
Primjer: $f(A,B,C,D,E) = \sum m(0,1,2,5,6,8,13,14,15,16,21,26,28,30,31)$

- ostvarenje 16-ulaznim MUX
  ~ standardno rješenje
  trivijalnim rezidualnim funkcijama
- ostvarenje 8-ulaznim MUX
  ~ rezidualne funkcije od 2 varijable

Ostvarivanje funkcija multipleksorom

- ostvarenje 8-ulaznim MUX
  ~ rezidualne funkcije od 2 varijable

$$f(A,B,C,D,E) = \sum m(0,1,2,5,6,8,13,14,15,16,21,26,28,30,31)$$

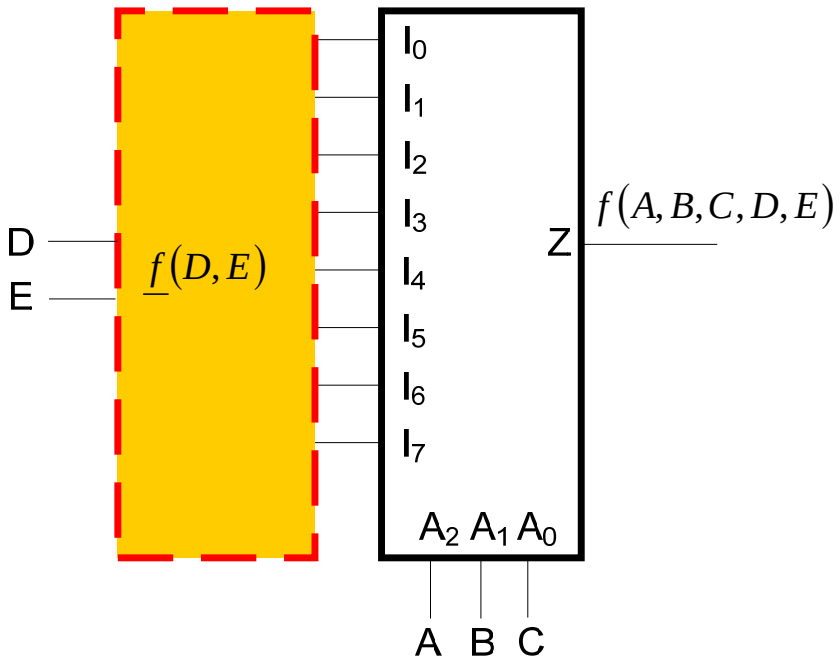


Ostvarivanje funkcija multipleksorom

- ostvarenje 8-ulaznim MUX
  ~ rezidualne funkcije od 2 varijable: $ABC \rightarrow A_2 A_1 A_0$

$$f(A,B,C,D,E) = \sum m(0,1,2,5,6,8,13,14,15,16,21,26,28,30,31)$$

i	A	B	C	$f_{res}=I_i$
	A_2	A_1	A_0	
0	0	0	0	$\overline{DE}$
1	0	0	1	$D \oplus E$
2	0	1	0	$\overline{D}\overline{E}$
3	0	1	1	$D + E$
4	1	0	0	$\overline{D}\overline{E}$
5	1	0	1	$\overline{D}E$
6	1	1	0	$D\overline{E}$
7	1	1	1	$D + \overline{E}$



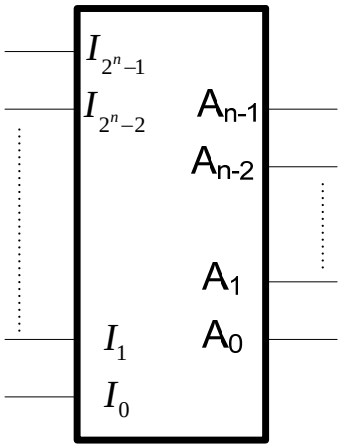
Sadržaj predavanja

- multipleksor
- **prioritetni koder**
- pretvornik koda
- komparator

Prioritetni koder

- funkcija *kodiranja*
  ~ generiranje *binarne* kodne riječi nekog koda; potrebno osigurati da je *samo jedan* ulaz aktivan!
  - koder
    ~ aktivan samo jedan ulaz (npr. $I_i = 1$)

    2^n ulaza $\rightarrow$ n izlaza
  - *prioritetni* koder
    ~ aktivno više ulaza, samo jedan djeluje!



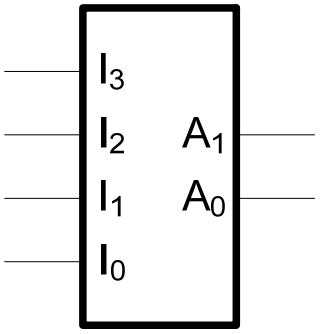
- tipična oznaka:

  2<broj adresa>/<broj adresa>

Prioritetni koder

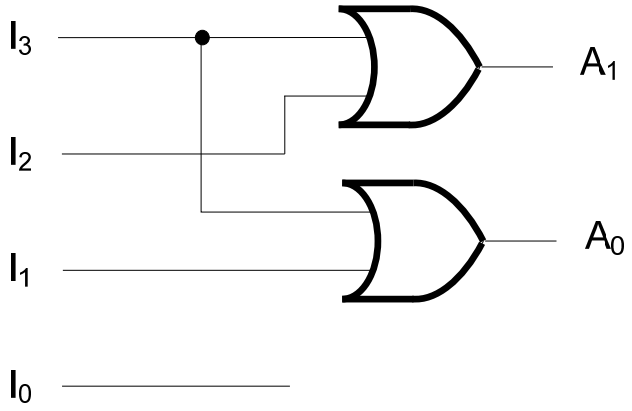
Primjer: koder 4/2

- ograničenje: uzorci ulaza s više 1 *ne mogu* se pojaviti
- simbol "0" (~ I_0) daje $A_1A_0 = 00$ ~ ne utječe!



I_3	I_2	I_1	I_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$$A_1 = I_3 + I_2$$
$$A_0 = I_3 + I_1$$

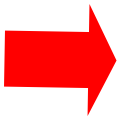


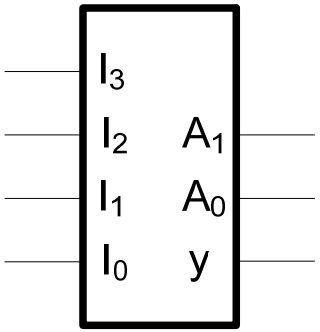
Prioritetni koder

- *prioritetni koder* (engl. priority encoder)
  ~ rješenje problema više aktivnih ulaza:
  - djeluje ulaz najvišeg prioriteta
  - svi ulazi = 0 ?
    ~ poseban izlaz:

    y = **if** I_3 **or** I_2 **or** I_1 **or** I_0 **then** 1 **else** 0

I_3	I_2	I_1	I_0	A_1	A_0	y
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

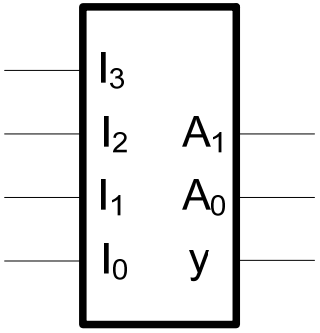




Prioritetni koder

- VHDL *ponašajni* model prioritetnog kodera 2/4

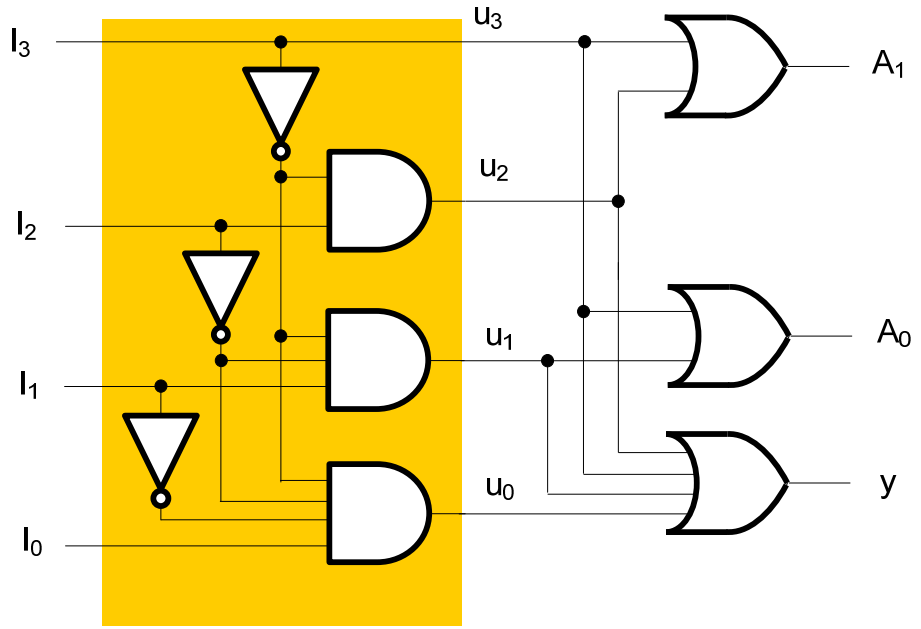
```
library ieee;
use ieee.std_logic_1164.all;

entity priorityEncoder is
  port (I: in std_logic_vector(3 downto 0);
        A: out std_logic_vector(1 downto 0);
        y: out std_logic);
end priorityEncoder;

architecture ponasajna of priorityEncoder is
  begin
    A <= "11" when I(3)='1' else
         "10" when I(2)='1' else
         "01" when I(1)='1' else
         "00";
    y <= '0'  when I="0000" else '1';
end ponasajna;
```



Prioritetni koder

Primjer: prioritetni koder 4/2 [Brown i Vranešić, 2000]

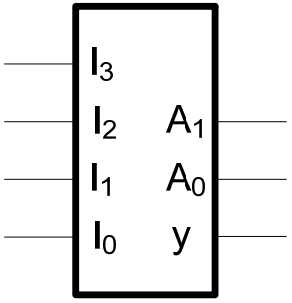
I_3	I_2	I_1	I_0	A_1	A_0	y
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1



$$u_3 = I_3$$
$$u_2 = \overline{I_3} \cdot I_2$$
$$u_1 = \overline{I_3} \cdot \overline{I_2} \cdot I_1$$
$$u_0 = \overline{I_3} \cdot \overline{I_2} \cdot \overline{I_1} \cdot I_0$$
$$y = u_3 + u_2 + u_1 + u_0$$

$$\Rightarrow$$

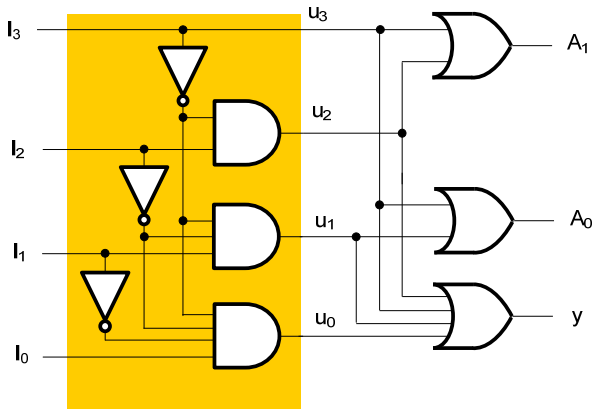
$$A_1 = u_3 + u_2$$
$$A_0 = u_3 + u_1$$



Prioritetni koder

- VHDL *strukturni* model prioritetnog kodera 2/4

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

entity priorityEncoder is
  port (I: in  std_logic_vector(3 downto 0);
        A: out std_logic_vector(1 downto 0);
        y: out std_logic );
  end priorityEncoder;

architecture strukturna of priorityEncoder is
  signal n3,n2,n1,u3,u2,u1,u0: std_logic;
  begin
    sklop1: entity work.sklopNOT  port map (I(3),n3);
    sklop2: entity work.sklopNOT  port map (I(2),n2);
    sklop3: entity work.sklopNOT  port map (I(1),n1);
    u3 <= I(3);
    sklop4: entity work.sklopAND2 port map (n3,I(2),u2);
    sklop5: entity work.sklopAND3 port map (n3,n2,I(1),u1);
    sklop6: entity work.sklopAND4 port map (n3,n2,n1,I(0),u0);
    sklop7: entity work.sklopOR2  port map (u3,u2,A(1));
    sklop8: entity work.sklopOR3  port map (u3,u2,u1,A(0));
    sklop9: entity work.sklopOR4  port map (u3,u2,u1,u0,y);
end strukturna;
```



Sadržaj predavanja

- multipleksor
- prioritetni koder
- **pretvornik koda**
- komparator

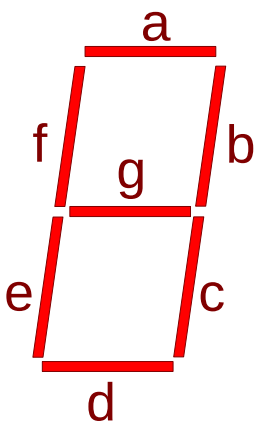
Pretvornik koda

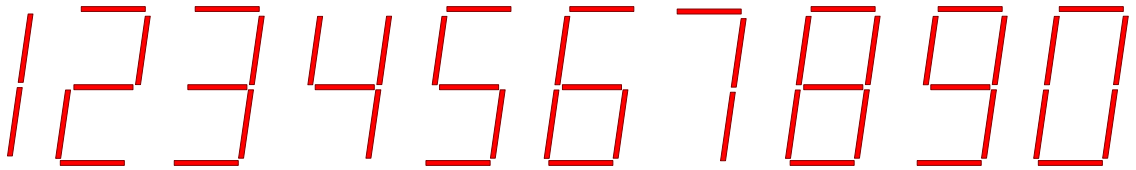
- *pretvornik koda* (engl. code converter):
  - pretvorba kodnih riječi *dvaju različitih* kodova
  - isti princip kao kod dekodera i kodera:
    - dekoder
      ~ kodna riječ $\rightarrow$ 1 aktivni izlaz
    - koder
      ~ 1 aktivni ulaz $\rightarrow$ kodna riječ
  - različiti tipovi MSI modula
    + mogućnost *kaskadiranja* (2-dimenzijske strukture)

Pretvornik koda

Primjer: pretvornik BCD koda u 7-segmentni

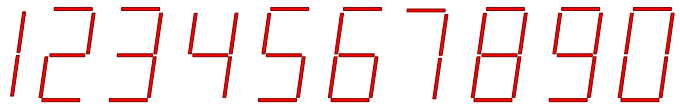
- vrlo raširena primjena
  ~ prikaz BCD znamenki
- element za prikaz
  ~ *7-segmentni prikaz*
  (engl. 7-segment display)

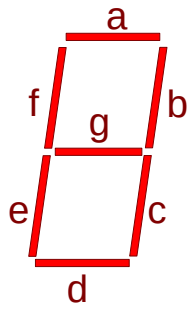




Pretvornik koda

- tablica pretvorbe BCD u 7-segmentni kod



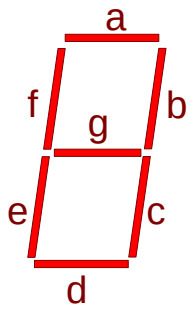


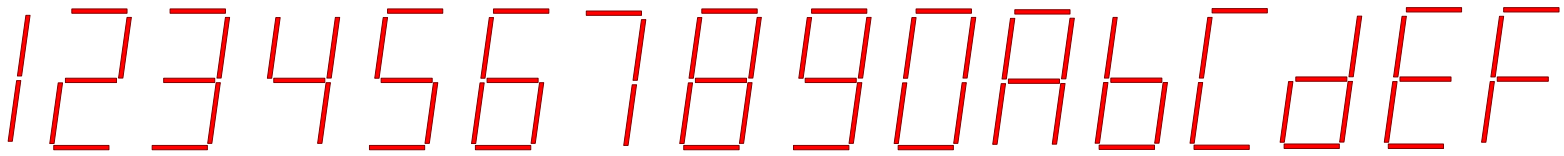
	D_3 2^3 8	D_2 2^2 4	D_1 2^1 2	D_0 2^0 1	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Pretvornik koda

Zadatak:

- napisati minimalne izraze za a, b, …, g
- nacrtati sklop
- napisati VHDL ponašajni *i* strukturni model
- ponoviti sve za pretvornik *heksadekadskih brojeva* u 7-segmentni kod



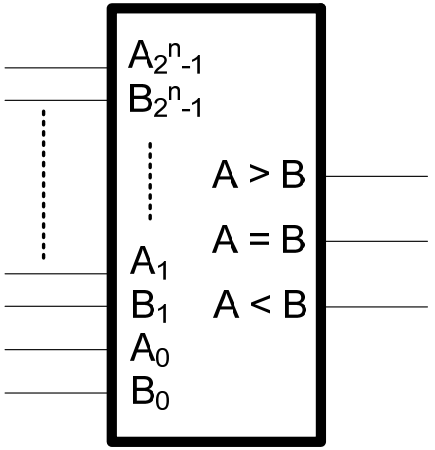


Sadržaj predavanja

- multipleksor
- prioritetni koder
- pretvornik koda
- **komparator**

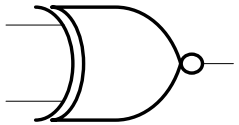
Komparator

- *komparator*
  ~ sklop za usporedbu dva *n*-bitna broja (npr. A i B)
  - obično cijeli brojevi *bez* predznaka
  - mogućnosti:
    - A = B
    - A > B
    - A < B
  - MSI modul
    ~ 4-bitni
    + mogućnost *kaskadiranja*



Komparator

- izgradnja komparatora (jedna mogućnost):
  - usporedba po bitovima
    ~ sklop EX-NILI



  - izlaz A = B
    ~ I funkcija usporedbi po bitovima
  - izlaz A > B
    ~ dominira prvi bit sa svojstvom $A_i > B_i$
    (počev od bita najviše težine)
  - izlaz A < B
    ~ **not** (($A_i > B_i$) **or** (A = B))

Komparator

Primjer: 4-bitni komparator

- usporedba po bitovima: $u_i = \overline{a_i \oplus b_i}, \quad i = 0,...,3$
- izlaz A = B: $"A = B" = u_3 \cdot u_2 \cdot u_1 \cdot u_0$
- izlaz A > B

  ~ *rekurzivno* utvrđivanje $a_i > b_i$, od bita najviše težine:

$$"a_3 > b_3" = a_3 \cdot \overline{b_3}$$
$$"a_2 > b_2" = a_2 \cdot \overline{b_2} \cdot u_3$$
$$"a_1 > b_1" = a_1 \cdot \overline{b_1} \cdot u_3 \cdot u_2$$
$$"a_0 > b_0" = a_0 \cdot \overline{b_0} \cdot u_3 \cdot u_2 \cdot u_1$$

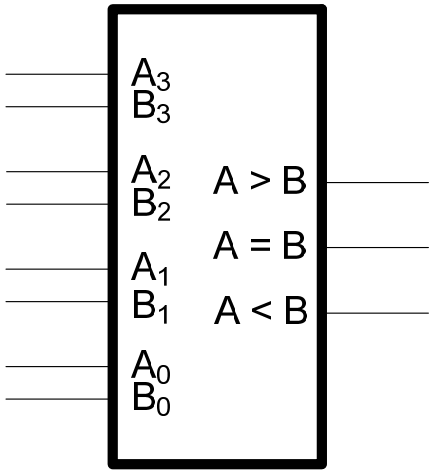
$$\Rightarrow \quad "A > B" = "a_3 > b_3" + "a_2 > b_2" + "a_1 > b_1" + "a_0 > b_0"$$

- izlaz A < B: $"A < B" = \overline{"A = B" + "A > B"}$

Komparator

Zadatak:

- nacrtati sklop 4-bitnog komparatora
- napisati tablicu kombinacija
- napisati VHDL ponašajni model sklopa
- napisati VHDL strukturni model sklopa



Literatura

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

- ostvarivanje Booleovih funkcija multipleksorom: str. 264-266

Zadaci za vježbu (1)

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

- ostvarivanje Booleovih funkcija multipleksorom: 7.7-7.26
- pretvornik koda: 7.47

Zadaci za vježbu (2)

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

- ostvarivanje Booleovih funkcija multipleksorom:
  - riješeni zadaci: 5.5-5.8, 5.11b, 5.16 (VHDL), 5.17
  - zadaci za vježbu: 16-18, 20-29, 32
- komparator
  - zadaci za vježbu: 30, 31