VTŠ: Osnovi računarske tehnike

Kombinaciona logika

mr Veličković Zoran

Mart, 2010.

Realizacija I funkcije i Invertora

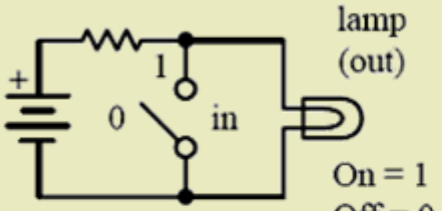
INVERT (NOT)

Truth table

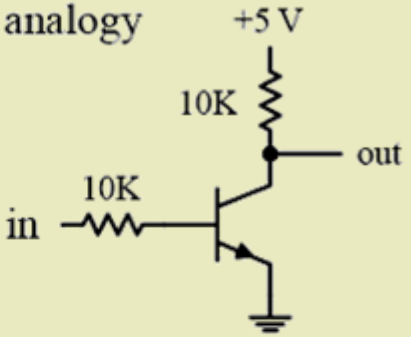
in	out
0	1
1	0

0 = LOW voltage level
1 = HIGH voltage level

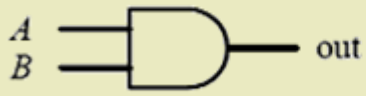
Switch analogy



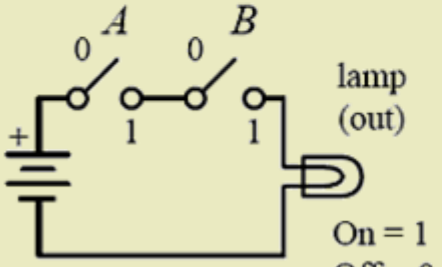
Transistor analogy

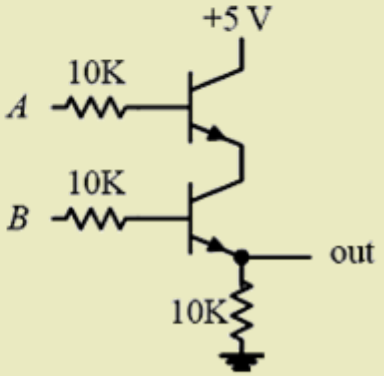


AND



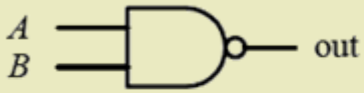
A	B	out
0	0	0
0	1	0
1	0	0
1	1	1



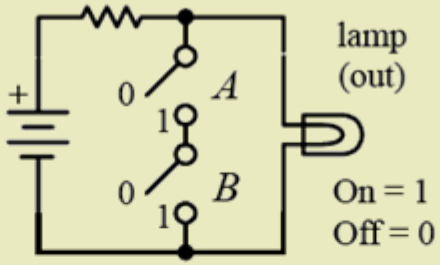


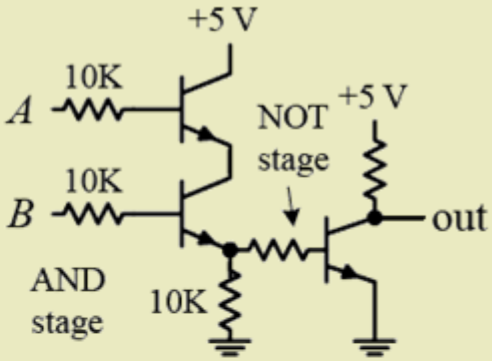
Realizacija NAND i OR funkcije

NAND



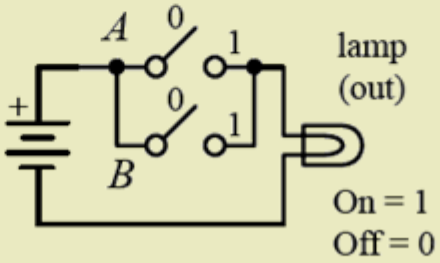
A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

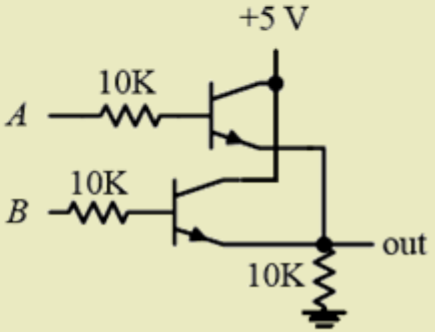




OR

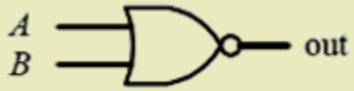
A	B	out
0	0	0
0	1	1
1	0	1
1	1	1



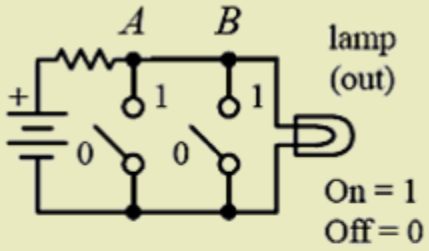


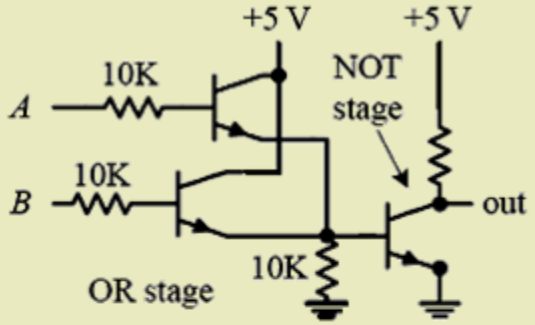
Realizacija NOR i XOR funkcije

NOR

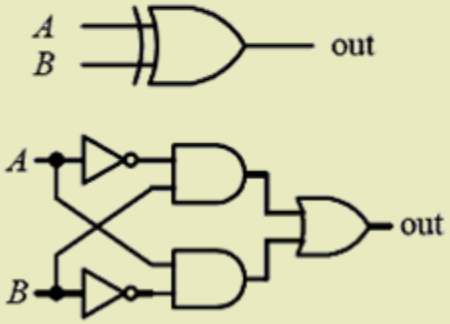


A	B	out
0	0	1
0	1	0
1	0	0
1	1	0





Exclusive OR (XOR)

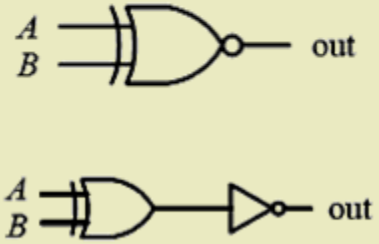


equivalent circuit

A	B	out
0	0	0
0	1	1
1	0	1
1	1	0

The output of an XOR gate goes HIGH if the inputs are different from each other. XOR gates only come with two inputs.

Exclusive NOR (XNOR)

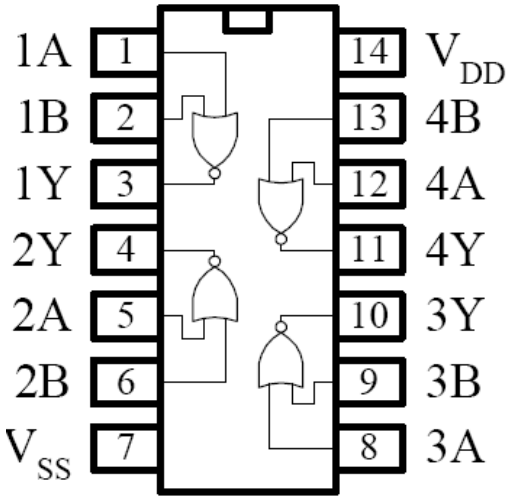


equivalent circuit

A	B	out
0	0	1
0	1	0
1	0	0
1	1	1

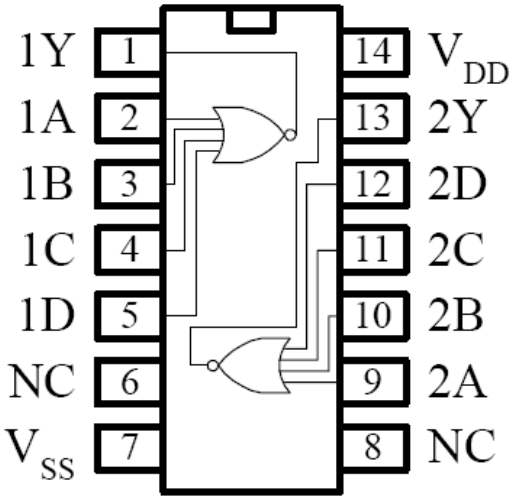
Standardna realizacija - DIP (1)

4x2-ulazno NILI kolo



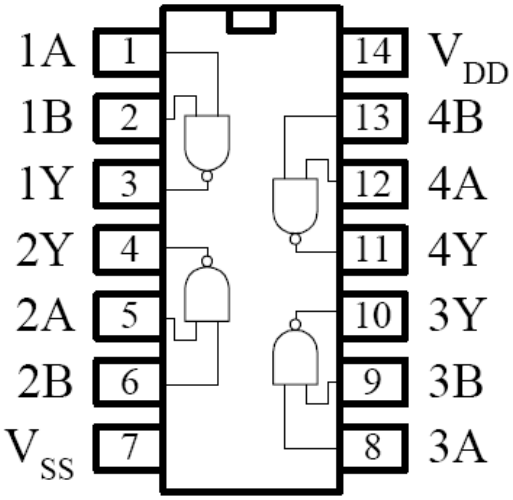
CMOS 4001(B)

2x4-ulazno NILI kolo



CMOS 4002(B)

4x2-ulazno NI kolo

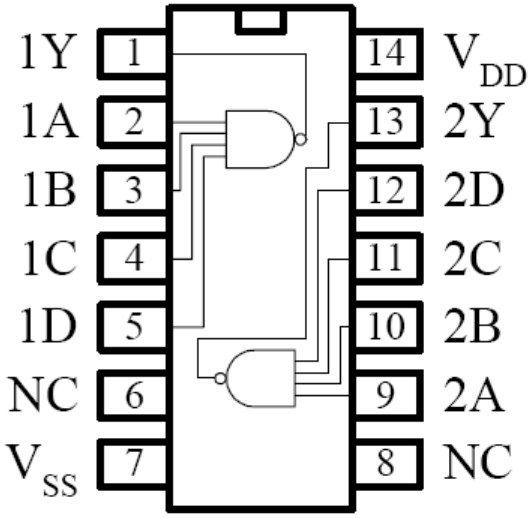


CMOS 4011(B)

Dual Inline Package - DIP

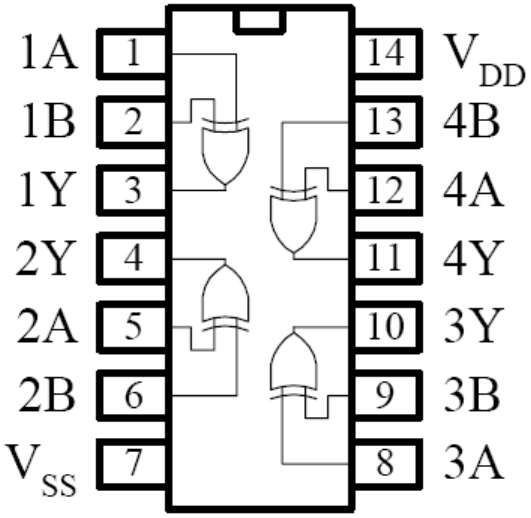
Standardna realizacija - DIP (2)

2x4-ulazno
NI kolo



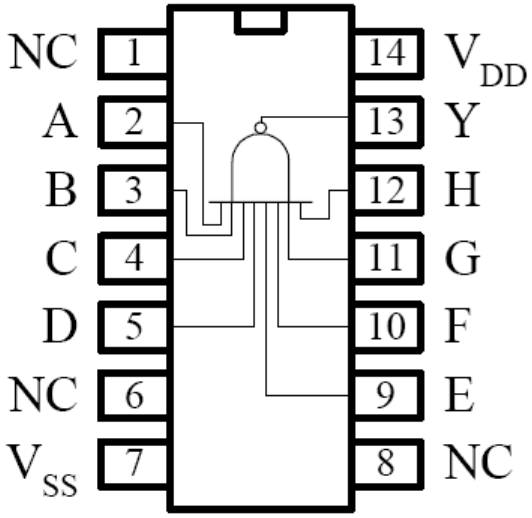
CMOS 4012(B)

4x2-ulazno
XOR kolo



CMOS 4030(B)

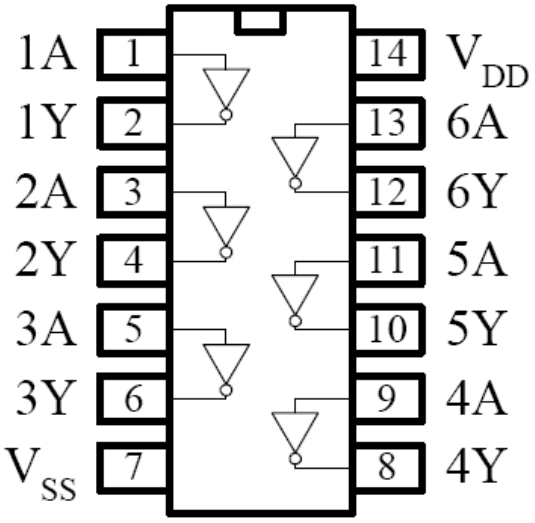
1x8-ulazno
NI kolo



CMOS 4068(B)

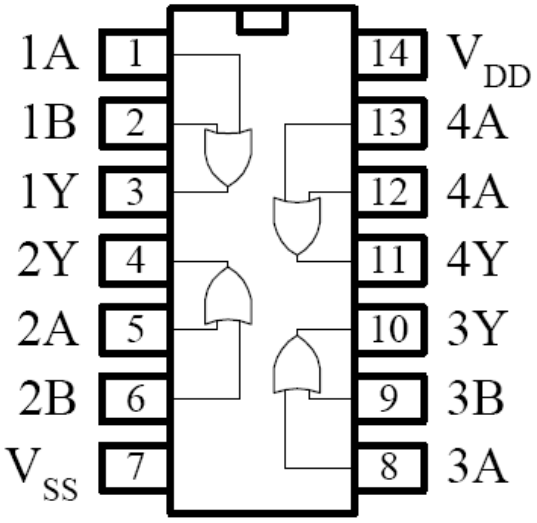
Standardna realizacija - DIP (3)

6x NE kolo



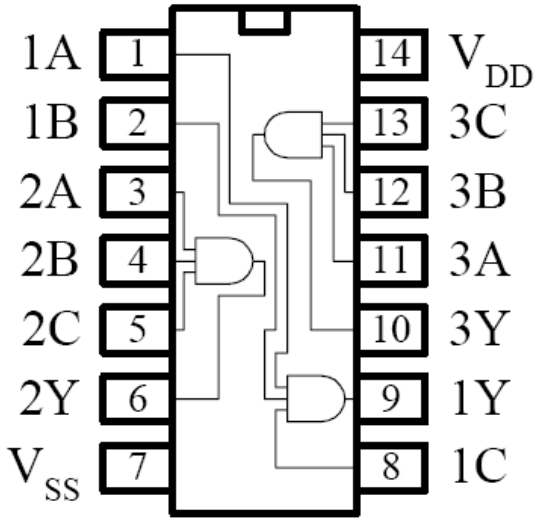
CMOS 4069(B)

4x2-ulazno ILI kolo



CMOS 4071(B)

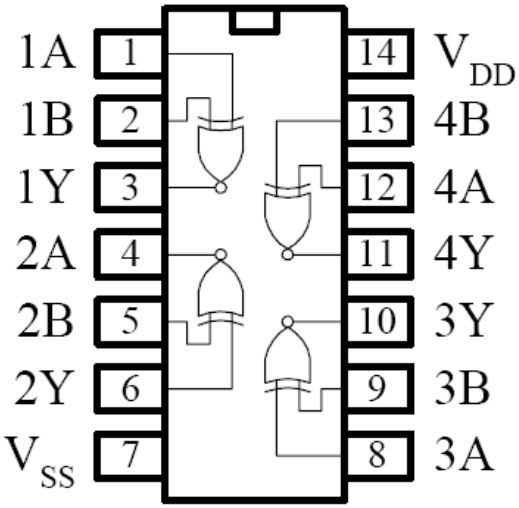
3x3-ulazno I kolo



CMOS 4073(B)

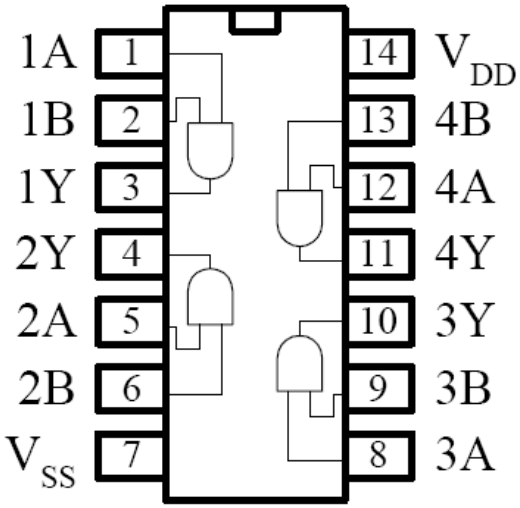
Standardna realizacija - DIP (4)

4x 2 ulazno NXOR kolo



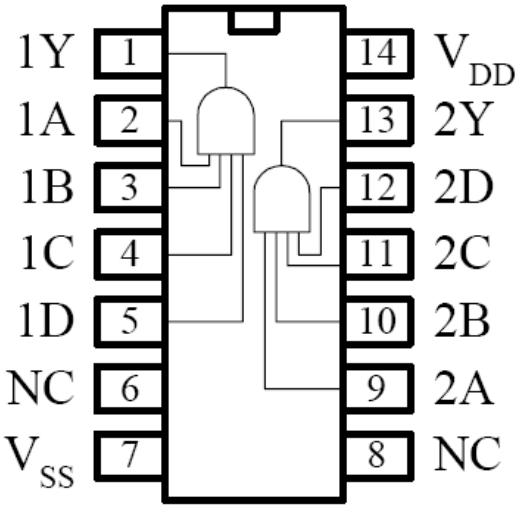
CMOS 4077(B)

4x2-ulazno I kolo



CMOS 4081(B)

2x4-ulazno I kolo

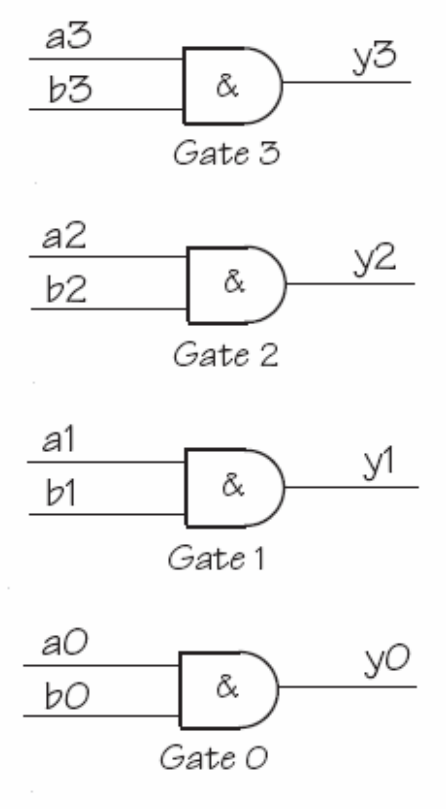


CMOS 4082(B)

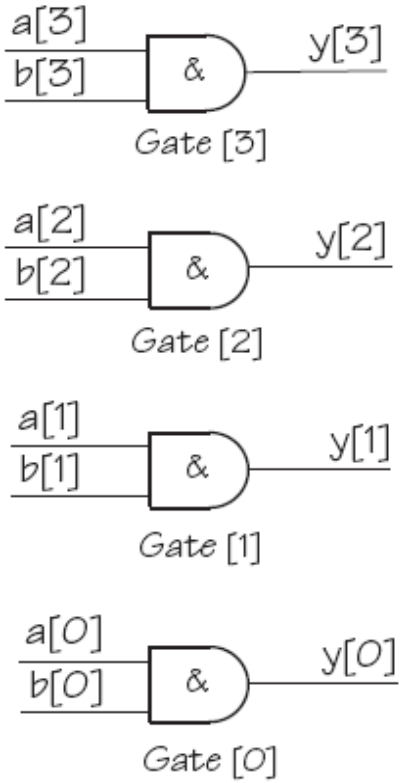
Kombinovanje logičkih modula

- Jednostavne logičke funkcije NOT, AND, OR, NAND, NOR i XOR se mogu međusobno povezati kako bi se sagradile **kompleksne funkcije**.
- Ovako dobijeni logički blokovi su zapravo gradivna materija za za još sofisticiranije sisteme.
- Primeri dati na ovom predavanju se često koriste kod **dizajniranja logičkih funkcija** i relativno su jednostavna za razumevanje i biće kasnije korišćene u ostalim predavanjima.
- Signal koji nosi samo jedan bit binarnih podataka naziva se **scalar**, dok se grupa sličnih podataka naziva **vektor**.

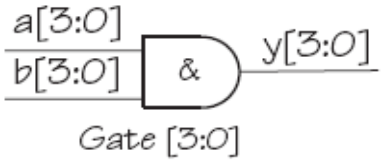
Kombinovanje logičkih modula



Skalarna notacija,
Svaki signal je posebno označen



Vektorska notacija,
isto ime za grupu signala



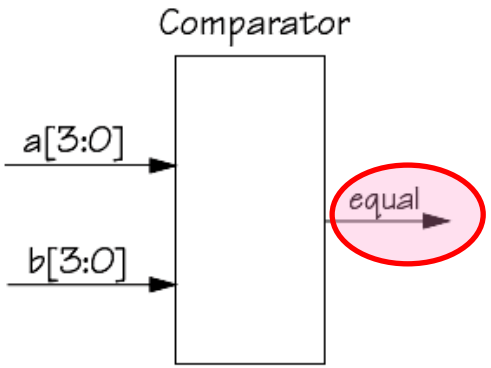
KOMPRIMOVANA
Vektorska notacija

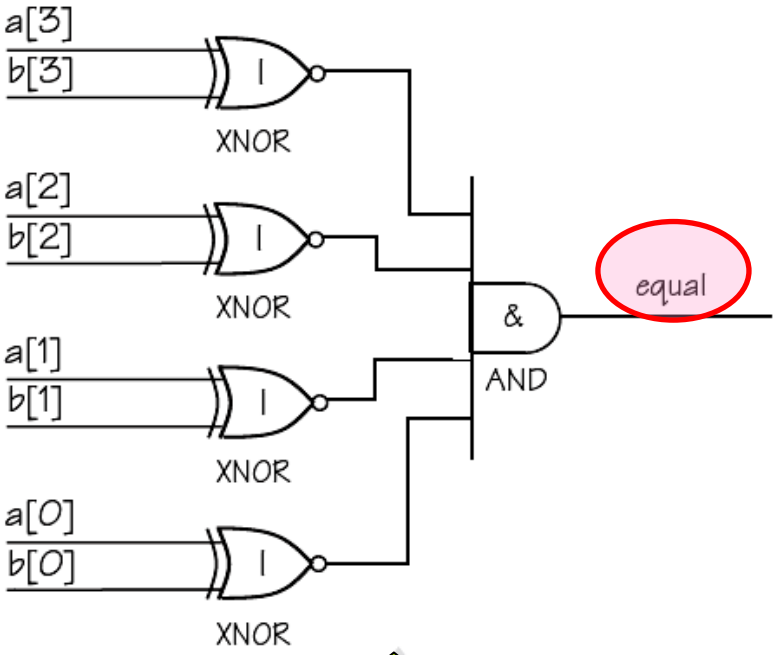
Komparator istovetnosti

- Često je potrebno komparirati dva skupa binarnih vrednosti kako bi se utvrdila njihova **jednakost**.
- Razmotrimo funkciju koja će komparirati dva **4-bitna** vektora **a[3:0]** i **b[3:0]**.
- Skalarni izlaz koji se naziva ***equal*** će biti postavljen na logičku jedinicu (1) samo ako je svaki bit u a[3:0] je jednak odgovarajućem bitu u b[3:0].
- Dakle vektori su istovetni-jednaki samo ako je

  a[3]=b[3], a[2]=b[2], a[1]=b[1], and a[0]= b[0]
- Vrednost u a[3] i b[3] se kompariraju pomoću **2-input XNOR** gejtom. Prema definicionoj tablici za ISKLJUČIVO ILI, na izlazu je logička jedinica samo ako su logičke vrednosti na ulazu iste.
- Slična komparacija treba da se izvrši za **sve ostale** bitove.
- Finalna kapija sakuplja rezultate **svih** pojedinačnih komparacija.

Realizacija komparatora





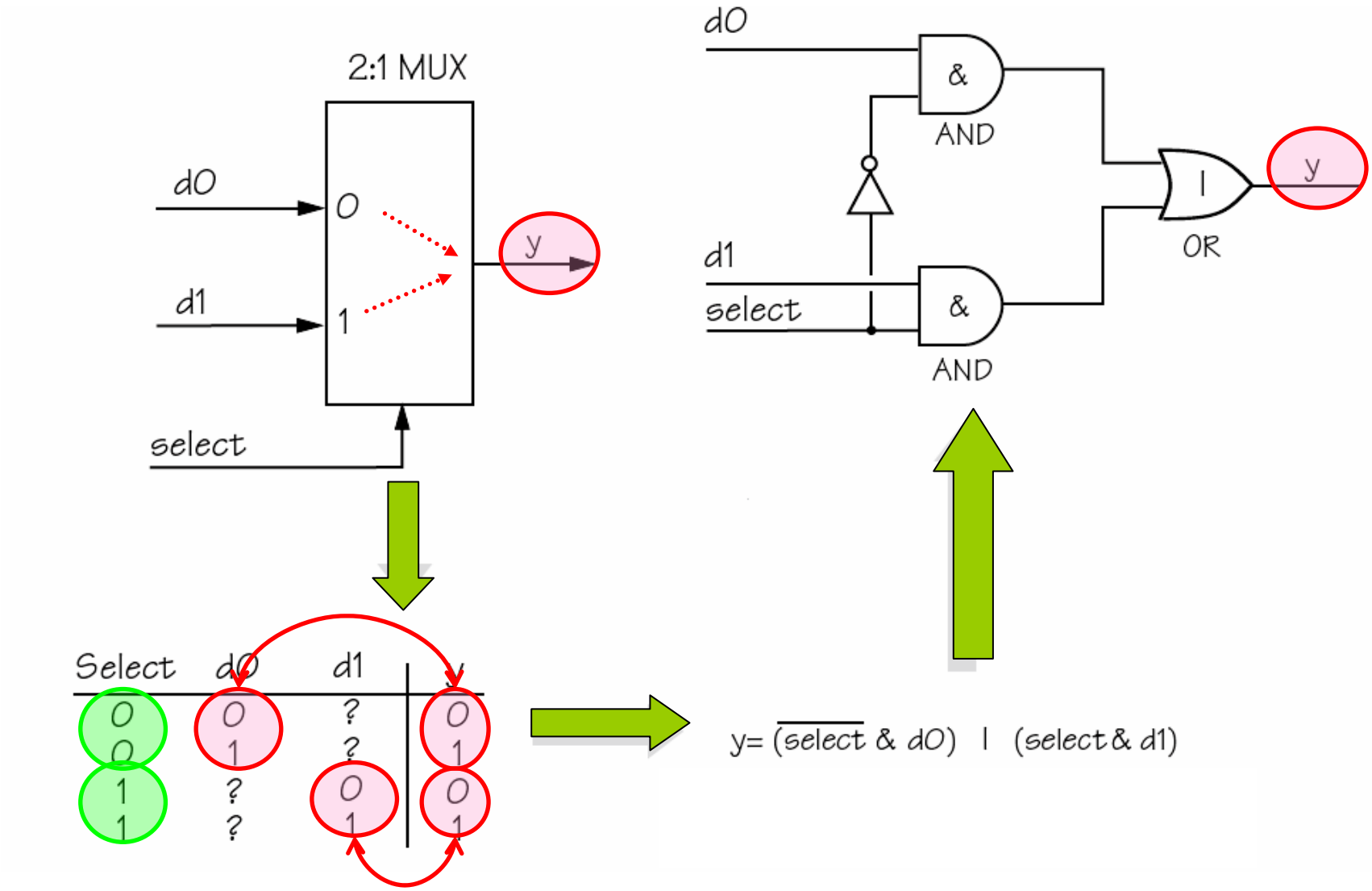
inputs	equal
a[3:0] ≠ b[3:0]	0
a[3:0] = b[3:0]	1

$$equal = \overline{(a[3] \wedge b[3])} \,\&\, \overline{(a[2] \wedge b[2])} \,\&\, \overline{(a[1] \wedge b[1])} \,\&\, \overline{(a[0] \wedge b[0])}$$

Multiplekser

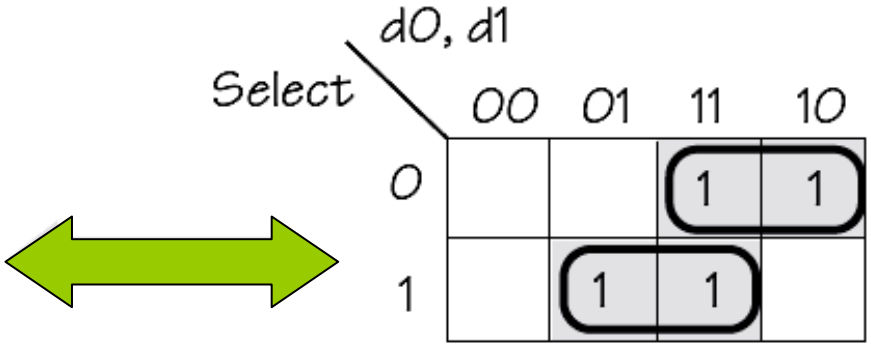
- Multiplekser koristi binarnu vrednost – adresu, da **selektuje jedan od ulaza** i da prenese podatak sa selektovanog **ulaza na izlaz**.
- Na primer pogledajte primer 2:1 (“dva u jedan”) multiplekser.
- Oznake **0** i **1** na multiplekseru pokazuju binarnu vrednost selektovanog ulaza koja se koristi za indikovanje koji će ulaz podataka biti selektovan.
- **Veći multiplekseri** su takođe u upotrebi, kao **4:1**, **8:1**, **16:1**.
- U zavisnosti od veličine multipleksera potrebno je obezbediti odgovarajući broj **linija za selektovanje**.
- Koliko linija je potrebno za multipleksere 8:1, i 16:1 ?

Multiplekser 2:1

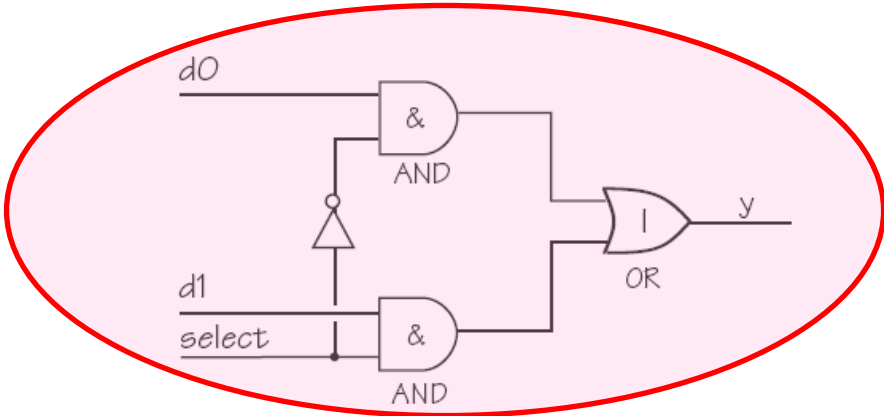


Multiplekser 2:1, Karnoova mape

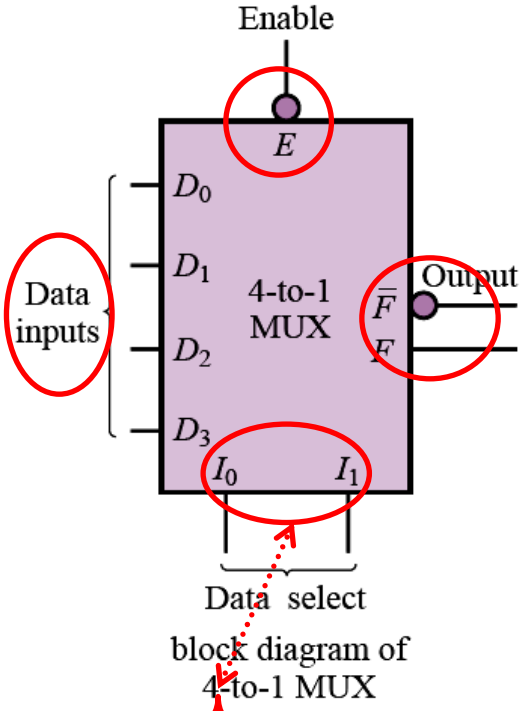
Select	d0	d1	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



$y = (\overline{select} \;\&\; d0) \mid (select \;\&\; d1)$



Multiplekser, 4:1



block diagram of 4-to-1 MUX

I_1	I_0	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Truth table of 4-to-1 MUX

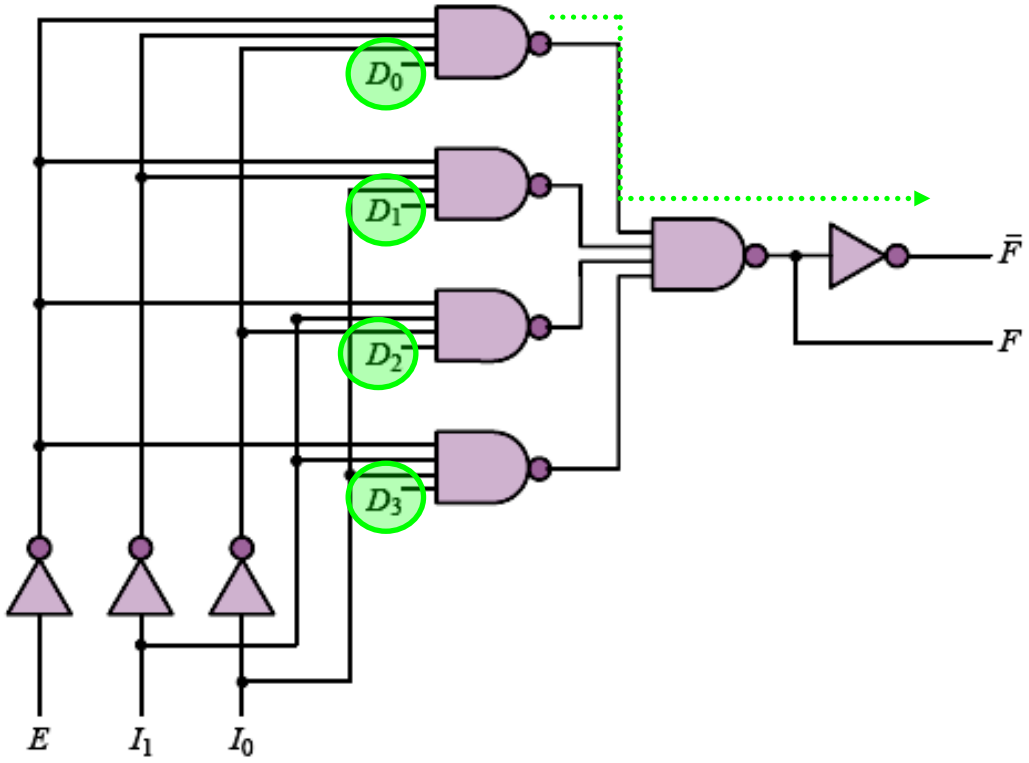
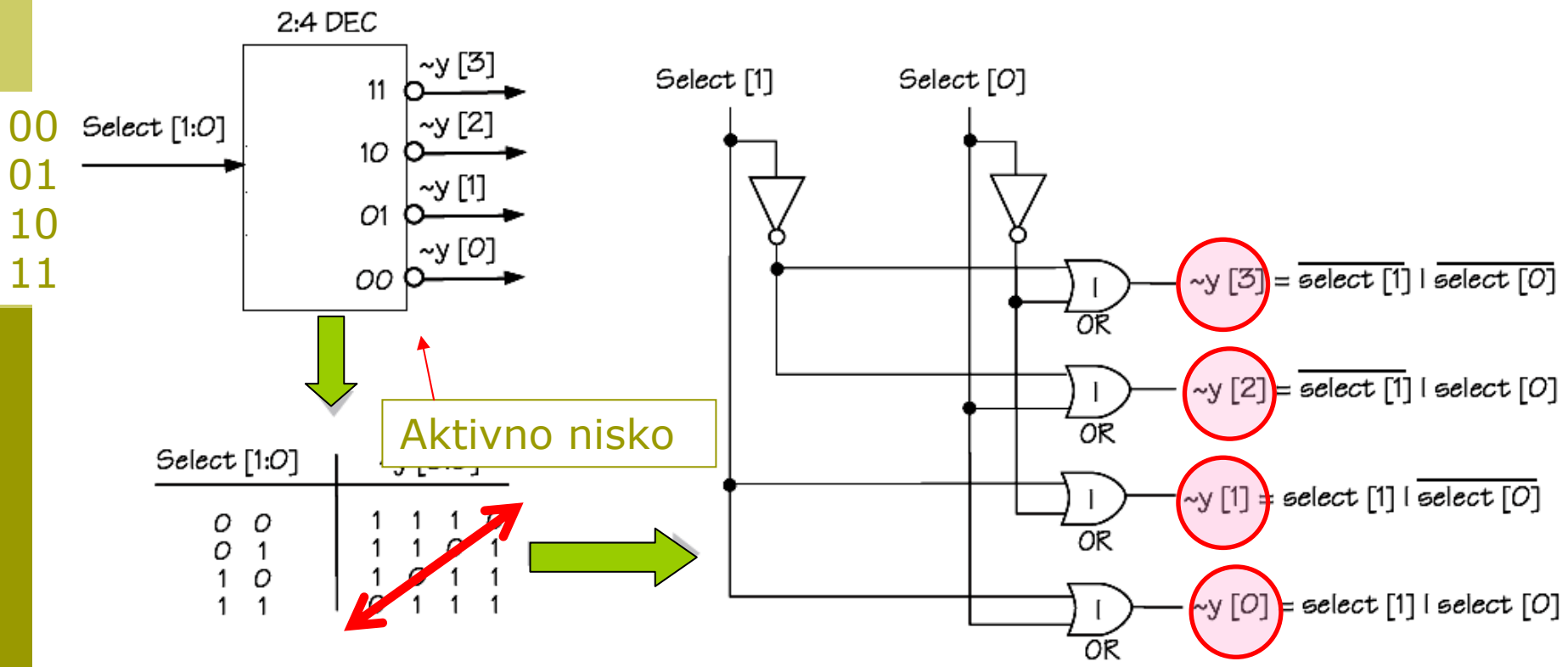


Figure 13.56 Internal structure of the 4-to-1 MUX

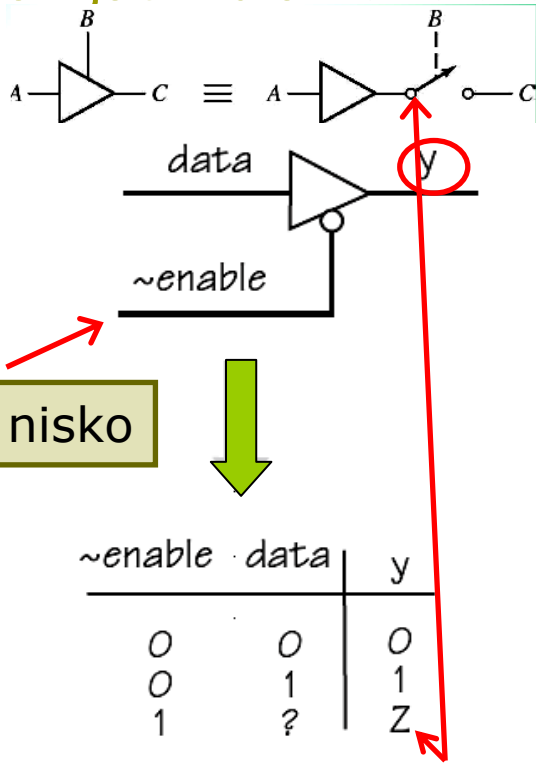
Dekoder, 2 u 4

- Dekoder koristi binarnu vrednost – adresu da selektuje jedan od izlaza i postavlja selektovani izlaz u njegovo aktivno stanje.
- Primer 2:4 ("dva u četiri") dekoder, aktivno nisko



Thre-state funkcije

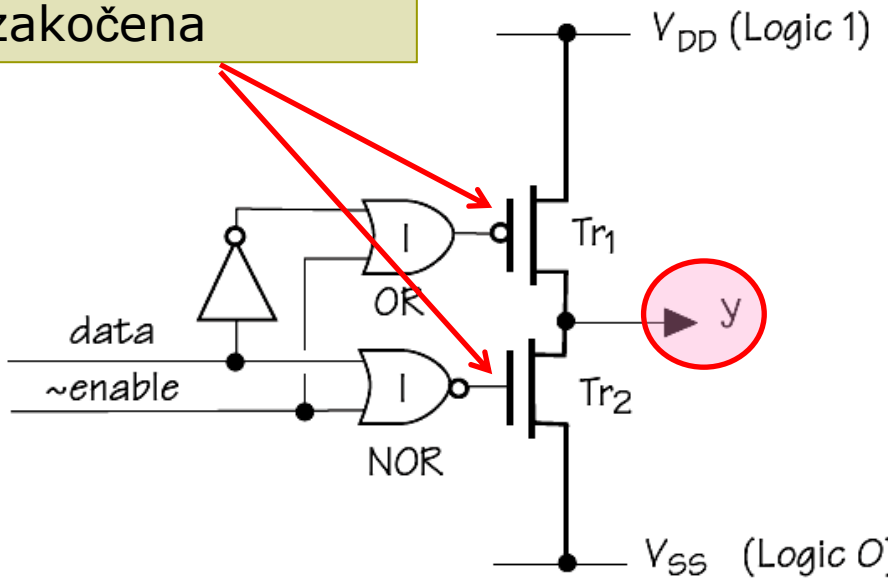
- **Specijalna kategorija** kapija čiji izlazi mogu imati **tri stanja**: **0**, **1**, and **Z**.
- Primer jednostavnog tri-state bafera. Obavezno poseduju dodatni upravljački ulaz ***enable*** *(dozvola).*



Aktivno nisko

~enable	data	y
0	0	0
0	1	1
1	?	Z

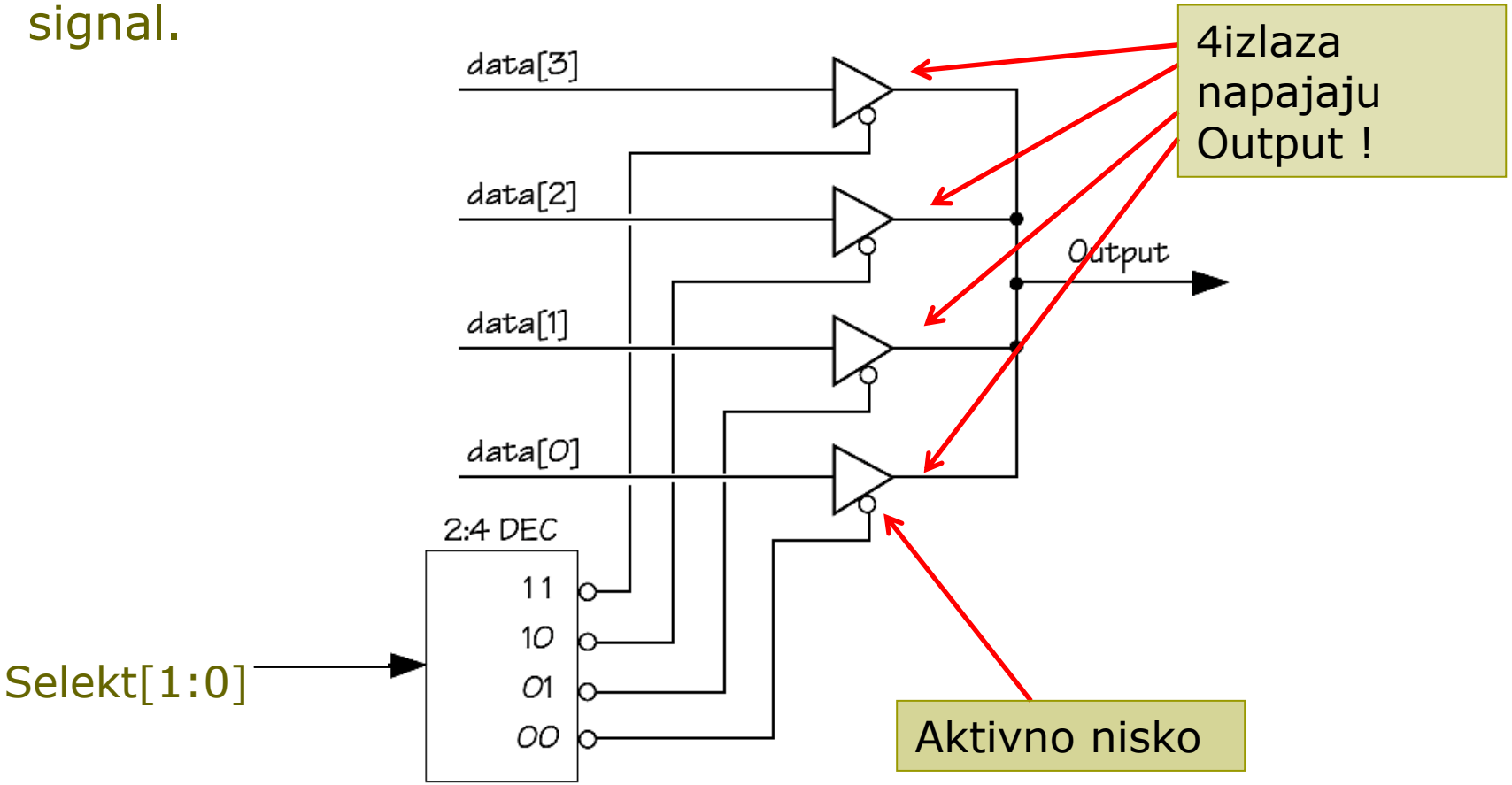
Za Hi-Z, oba tranzistora su zakočena



Visoka impedansa, Hi-Z, izlazi kao da su otkačeni !

Thre-state funkcije

- Tri-state buffers se koriste u kombinaciji sa dodatnom logikom da se omogući da više izlaza mogu napajati isti signal.



ROM-Read Only Memory

- **ROM** je još jedna tehnika za implementaciju logičkih funkcija.
- ROM je logičko kolo koje **zadržava logičko stanje** u formi binarnih brojeva i može biti pročitano logičkim kolima.
- ROM je **niz memoriskih ćelija** koje od kojih svaka može **zapamtiti** logičku nulu ili logičku jedinicu.
- Niz memorijskih ćelija se sastoji od ***m* ×*n*** *ćelija*, gde je n broj bitova u svakoj reči zapamćene u ROM-u.
- Za pristup informacijama smeštenim u ROM-u, ***m* adresnih linija** se zahteva.
- ROM se može smatrati **multiplekserom** čiji j izlaz **reč** umesto jediničnog bita!

ROM kao MUX

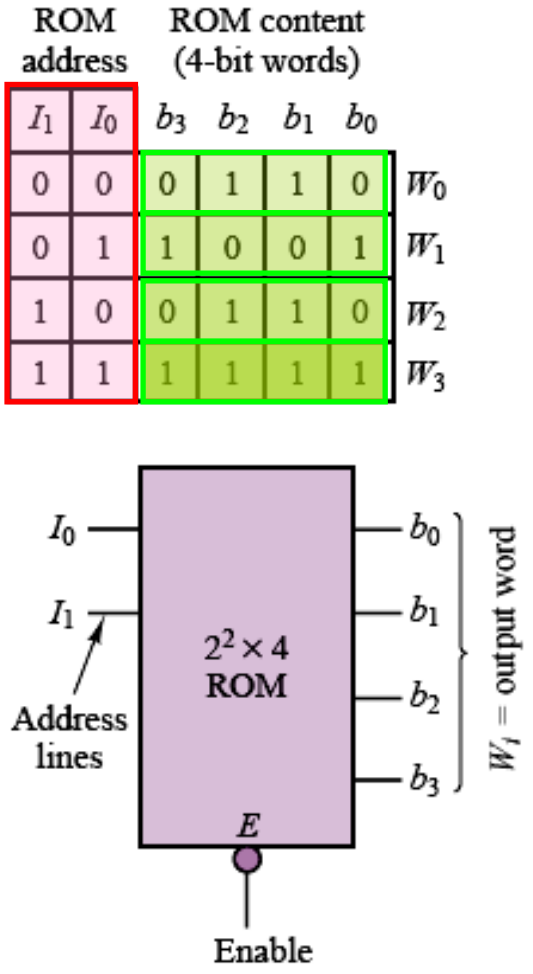


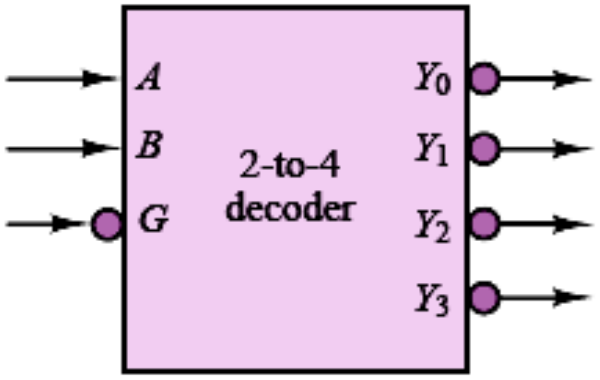
Figure 13.58 Read-only memory

- n=4, m=2
- ROM tabela može biti popunjena **proizvoljnim** sadržajem!
- Podaci smešteni ROM-u moraju biti smešteni u **vreme proizvodnje** i ne mogu se kasnije menjati.
- **EPROM** je vrsta ROM-a čiji se sadržaj može **menjati** ako je to potrebno.

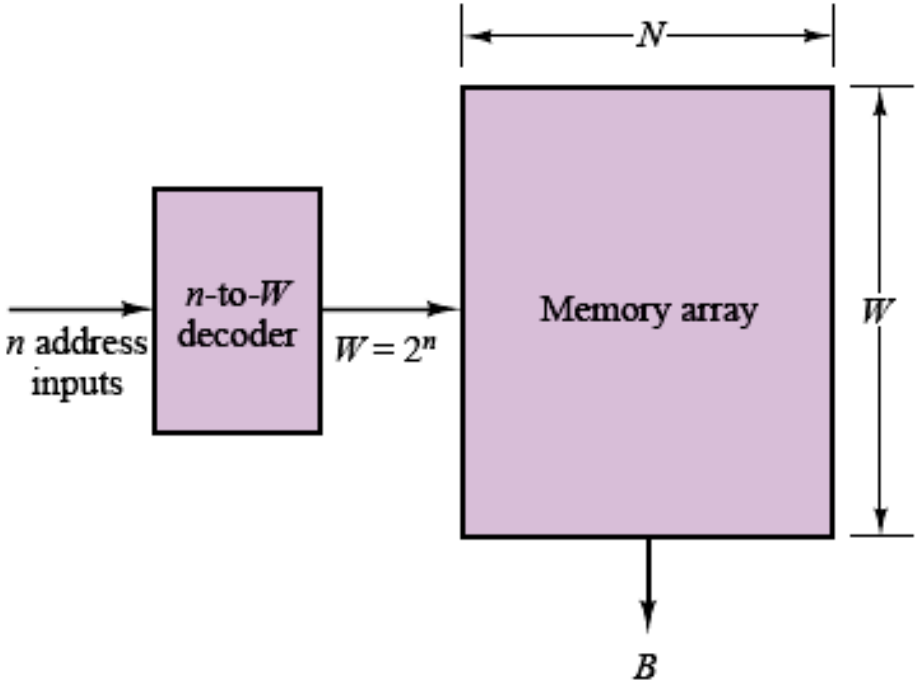
Dekoderi i SRAM

- Dekoderi su logička kola koja se upotrebljavaju za **dekodiranje (prepoznavanje) adresa** ili za proširivanje memorijskog prostora.
- Veoma je važna uloga dekodera kod organizacije **memoriskih sistema**.
- **SRAM** (Static Random-Sccess) je **memorijski element** koji je organizovan da omogući brz i jeftin pristup podacima.
- SRAM ima dužinu kolone jednaku **broju bitova u reči *W***, i vrsta dužine jednaka **broju bitova u reči *N***.
- Da bi se selektovala reč *n*-u-*W* dekoderu su potrebni adresni ulazi koji predstavljaju **ulaz u dekoder** koji selektuje samo **jedan** dekoderov izlaz.
- Dekoder selektuje samo **jednu reč** iz memoriskog niza.

Dekoderi i SRAM



Inputs			Outputs			
Enable	Select					
$\overline{G}$	A	B	Y_0	Y_1	Y_2	Y_3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



Struktura SRAM-a

Kombinaciona logika - DIP

4 x 1 od 2 data selctor

74LS157 quad 1-of-2 data selector