



11. Programirljivi moduli



Sadržaj predavanja

- **koncept programirljivih modula**
- permanentna memorija
- programirljivo logičko polje
- poluprogramirljivo logičko polje
- složeni programirljivi moduli
- programirljivo polje logičkih blokova

Koncept programirljivih modula

- *programirljivi* moduli
 - ~ "programirljive naprave", PLD
(engl. Programmable Logic Devices):
 - ostvarivanje složenije funkcije koja *nije* unaprijed određena
 - ~ moduli opće namjene
 - mogućnost *naknadnog* "programiranja"
 - ~ konfiguriranje sklopa u smislu određivanja "izvana opazivog ponašanja":
 - u posebnim uređajima
 - unutar uređaja u kojem se modul koristi





Koncept programirljivih modula

- struktura složenija (puno "logike"), ali *nije* fiksirana:
 - logički sklopovi ("vrata", engl. gates)
ili skupovi logičkih sklopova (\sim "logički blokovi")
 - tvornički izvedeni kontakti/veze ili *programirljive sklopke*:
 - različita povezivanja logičkih sklopova
 - konfiguriranje skupova logičkih sklopova
unutar modula
 - osnovna struktura
 \sim dvodimenzijsko polje dekodekoder:
permanentna memorija



Koncept programirljivih modula

- podjela programirljivih modula:
 - jednostavni PLD (engl. Simple PLD, SPLD):
 - programirljivo logičko polje, PLA
 - poluprogramirljivo logičko polje, PAL
 - složeni PLD (engl. Complex PLD, CPLD)
~ *više programirljivo povezanih SPLD u modulu*
 - programirljiva polja logičkih blokova
(engl. Field Programmable Gate Arrays, FPGA)
~ *veliki broj programirljivo povezanih programirljivih logičkih blokova*



Sadržaj predavanja

- koncept programirljivih modula
- **permanentna memorija**
 - **funkcionalnost i struktura**
 - **tehnologija izvedbe**
 - **karakteristične primjene**
 - **generiranje Booleovih funkcija**
- programirljivo logičko polje
- poluprogramirljivo logičko polje
- složeni programirljivi moduli
- programirljivo polje logičkih blokova



Permanentna memorija

- funkcijski pogled
 - ~ sklop s *permanentno* upisanim "sadržajem"
= *memorija*:
 - jedan upis (obično pri proizvodnji), ostalo čitanje
~ ispisna memorija:
"samo-se-čita", ROM (engl. Read Only Memory)
 - može i više upisa, ali *zanemarivo malo* u odnosu na broj čitanja
- izvedba
 - ~ *kombinacijski* sklop:
 - podatak upisan nekom vrstom "ožičenja"
 - mogućnost "programiranja"

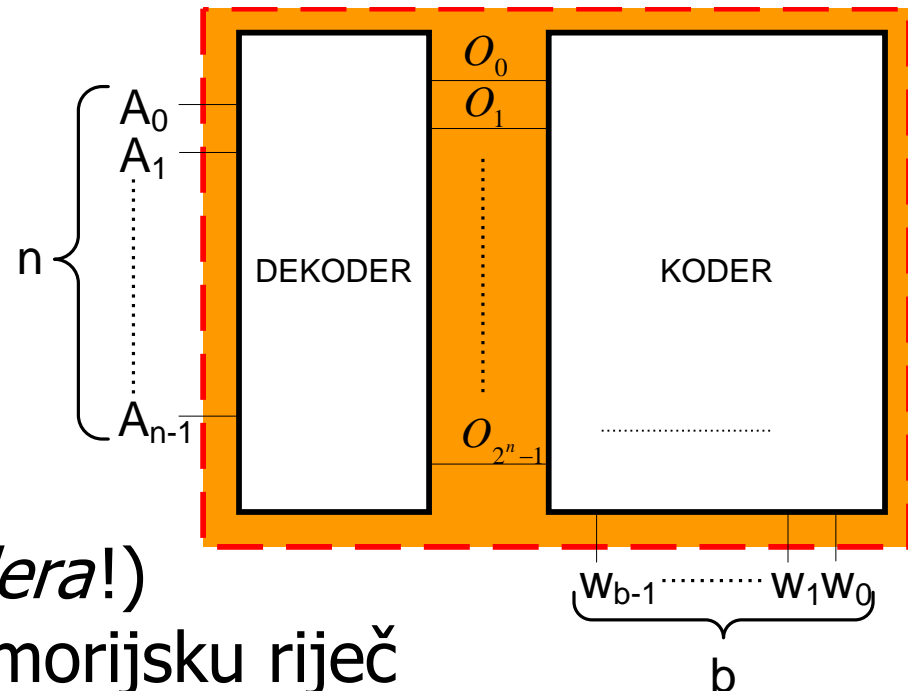


Permanentna memorija

- karakteristična struktura
~ *dva* polja:
 - ulazno ili *dekodersko* polje:
 - generiranje potrebnog broja internih adresnih linija
 - *potpuno* (binarno) adresiranje: "1-od- 2^n "
 - dekodeer
~ I sklopovi na izlazima → *I polje*
 - izlazno ili *kodersko* polje:
 - generiranje bitova adresirane "riječi"
~ aktiviranje željenih izlaza:
"kodiranje" pojedinih simbola (~ izlaza iz dekodera)
 - koder
~ ILI sklopovi na izlazima → *ILI polje*
(izlaz = podatak1 ILI podatak2 ILI ...)

Permanentna memorija

- karakteristična struktura:
 - dva polja
 - n (adresnih) ulaza
~ broj "memorijskih riječi" = 2^n
 - broj bitova/riječ = b
 - "kapacitet": $W = 2^n \times b$
- programiranje (samo *kodera*!)
~ upis uzorka 1 i 0 \forall memorijsku riječ



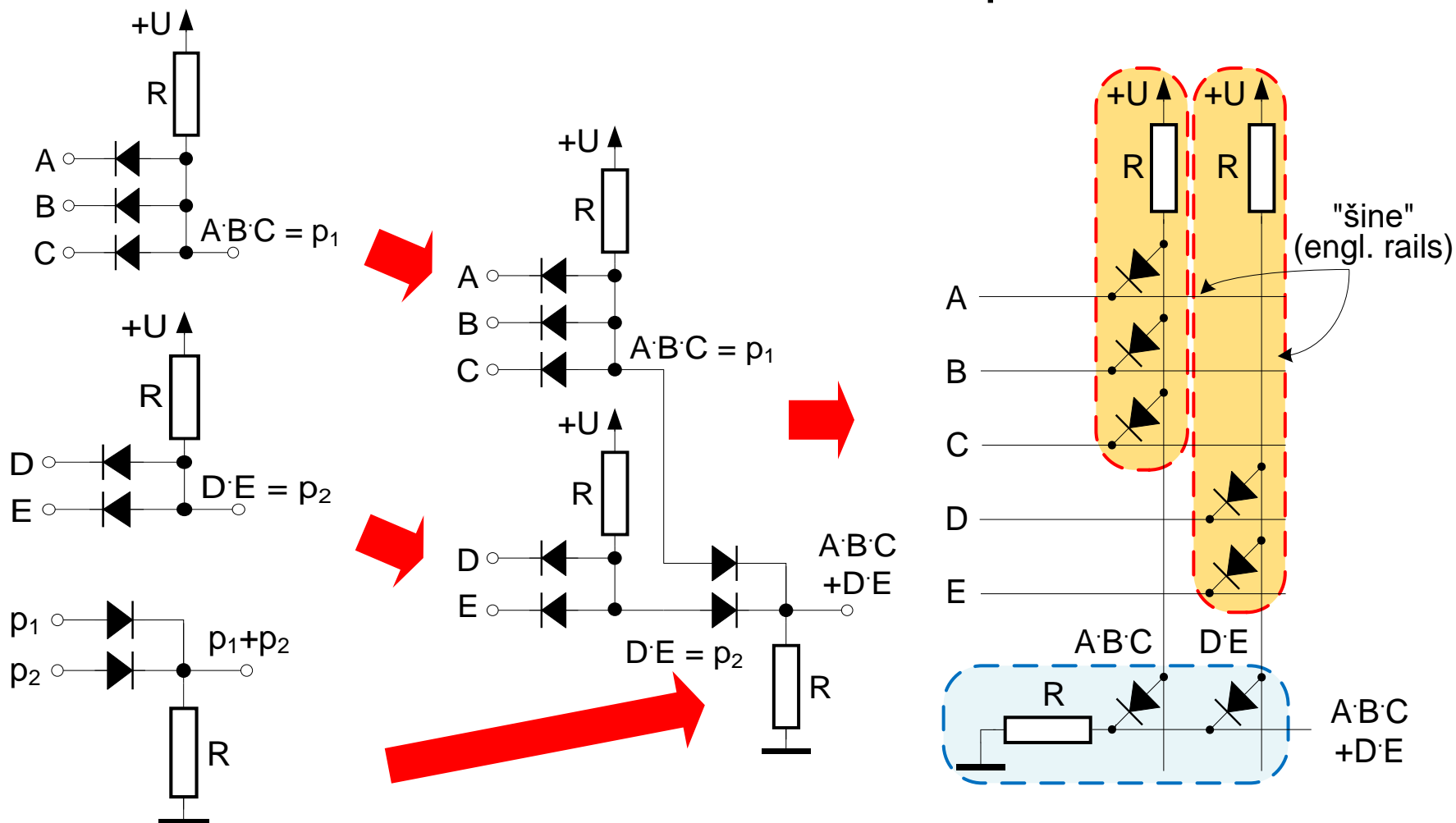


Izvedbe permanentne memorije

- karakteristična struktura s *dva* polja
~ izvorno *diodna matrica*:
 - osnovni logički sklopovi ostvareni *diodnim mrežama*
 - struktura tipa funkcije drugog reda (suma minterma)
~ oblik ILI-I
 - električka funkcija dioda
~ onemogućiti *povratno* djelovanje s izlaza sklopova I, ILI
koji se vode preko "šina" (engl. rails)
 - suvremene strukture
~ *poopćenja* diodne matrice
 - povijesno
~ *upravljačka memorija* (engl. control store)
računala *MIT Whirlwind* (Forrester i Everett, 1947)

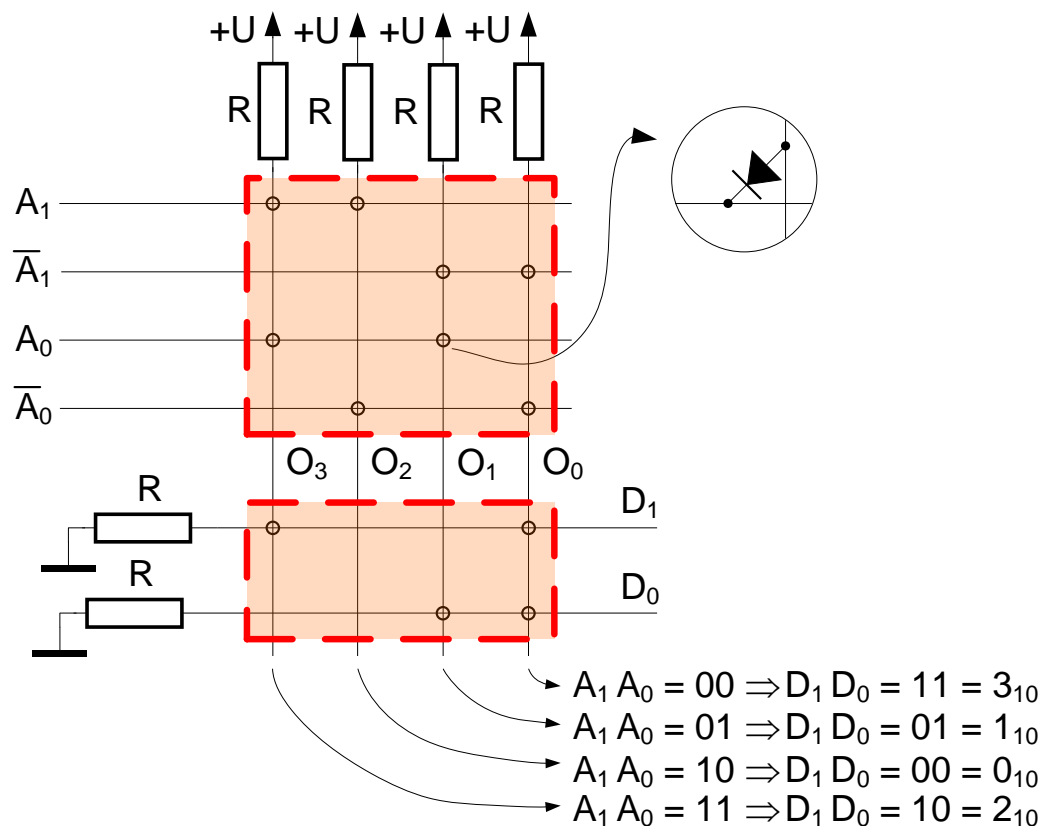
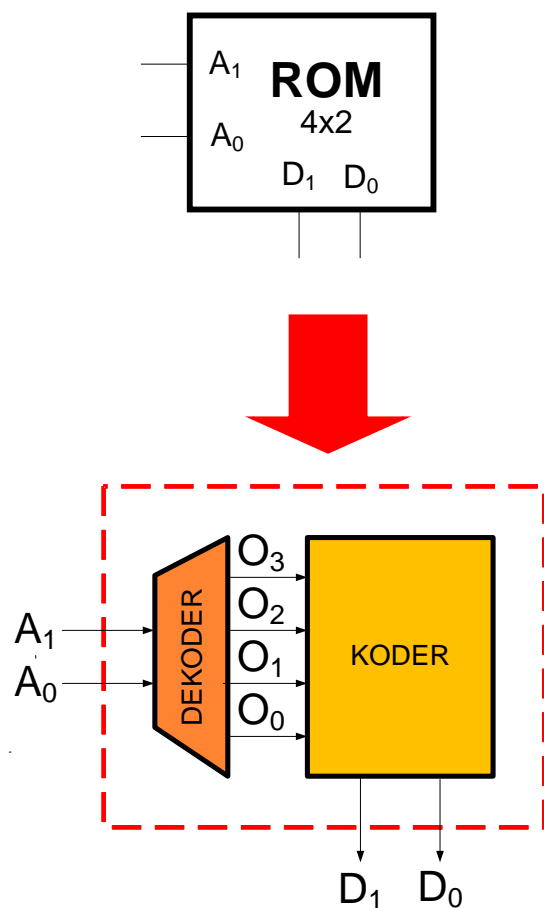
Izvedbe permanentne memorije

- izvedba diodne matrice iz diodnih sklopova I i ILI



Izvedbe permanentne memorije

Primjer: Izvedba ROMa s 4 2-bitne riječi (ROM 4x2)
diodnom matricom





Izvedbe permanentne memorije

Zadatak: temeljeći sa na izvedbi permanentne memorije diodnom mrežom nacrtati:

- sklop 1-bitnog potpunog zbrajala
- sklop 1-bitnog potpunog odbijala
- sklop 1-bitnog potpunog zbrajala/odbijala
(uputa: predvidjeti upravljačku varijablu K za odabir zbrajanja ($K = 0$), odnosno oduzimanja ($K = 1$))
- na raspolaganju su varijable i komplementi



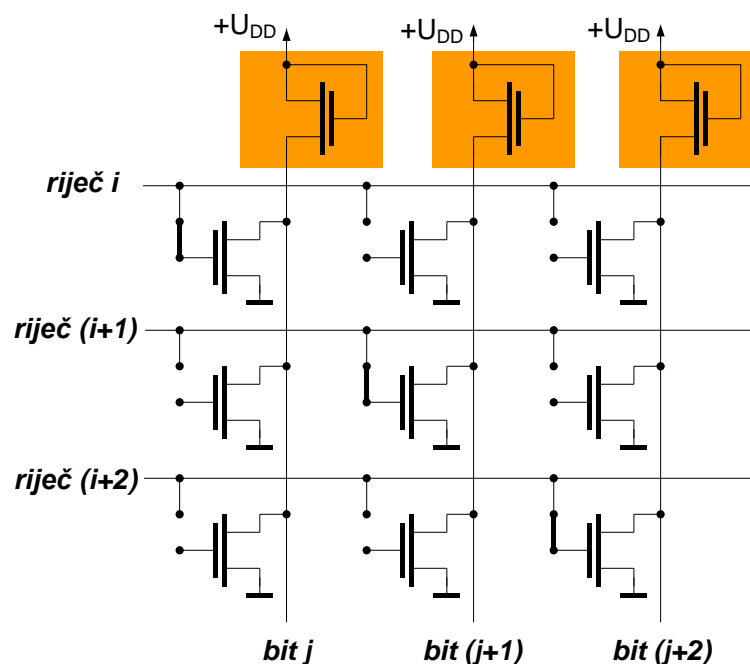
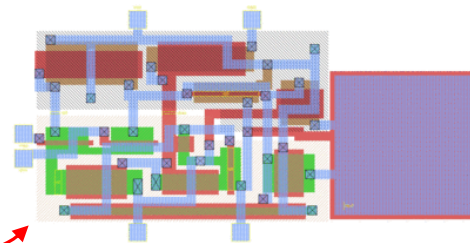
Izvedbe permanentne memorije

- izvedbe ~ tehnologija:
 - bez mogućnosti programiranja, ROM
 - s mogućnošću (jednokratnog) programiranja, PROM (engl. Programmable ROM)
 - s mogućnošću *višekratnog* programiranja i brisanja UV svjetlom, EPROM (engl. Erasable PROM) ~ kućište sa staklenim prozorčićem
 - s mogućnošću višekratnog programiranja i brisanja *električkim* putem, EAROM (engl. Electrically Alterable ROM), EEPROM (engl. Electrically EPROM)

Izvedbe permanentne memorije

- "klasična" permanentna memorija, ROM:

- uobičajena tehnologija
~ MOSFET
- programiranje u proizvodnji
~ zadnja se *maska* izrađuje po narudžbi
i sadrži potrebne veze
- postoji veza s linijom riječi
~ "programirana" je 0,
inače 1
- $t_a \sim 100 \text{ ns}$





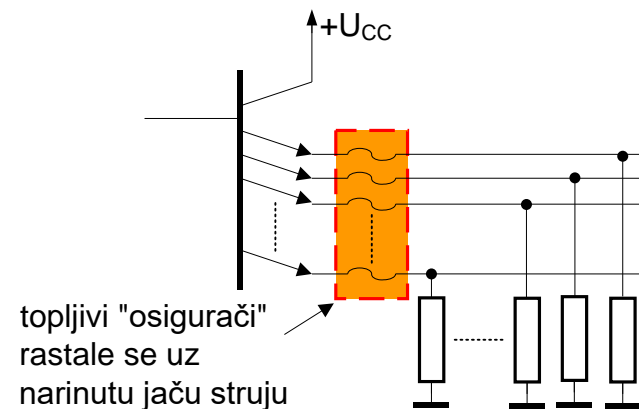
Izvedbe permanentne memorije

Zadatak: temeljeći sa na izvedbi permanentne memorije MOSFET tranzistorima nacrtati:

- sklop 1-bitnog potpunog zbrajala
- sklop 1-bitnog potpunog oduzimala
- sklop 1-bitnog potpunog zbrajala/oduzimala
(uputa: predvidjeti upravljačku varijablu K za odabir zbrajanja ($K = 0$), odnosno oduzimanja ($K = 1$))

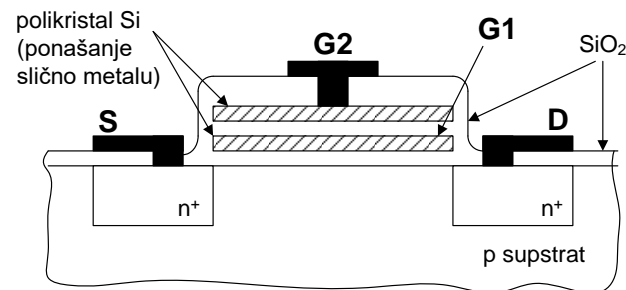
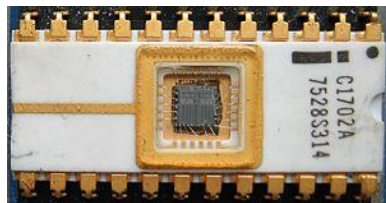
Izvedbe permanentne memorije

- s mogućnošću *jednokratnog* programiranja, PROM:
 - bipolarna tehnologija, tipično TTL
~ višeemitorski izlazni tranzistor,
priteže prema masi;
odgovara memorijskoj "riječi"
- za male serije:
 - programiranje "na licu mjesta"
(engl. in-the-field),
u posebnim uređajima
 - programiranje taljenjem
"osigurača" jačom strujom ($U_B \gg$)
- $t_a \sim 30 \div 50$ ns



Izvedbe permanentne memorije

- s mogućnošću *višekratnog* programiranja i brisanja *UV svjetlom*, EPROM:
 - tehnologija MOSFET
 - ~ posebna izvedba NMOS tranzistora "s lebdećom elektrodom", FAMOS (engl. Floating-gate Avalanche Injection MOS)



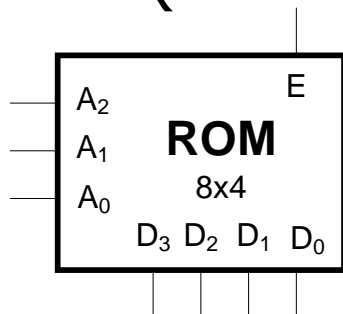
- programiranje u posebnim uređajima uz prethodno brisanje UV svjetlom
 - ~ $U_{G2D} \sim 25 \text{ V}$, prodor elektrona u G_1 *lavinskim probojem*
- $t_a \sim 200 \text{ ns}$

Izvedbe permanentne memorije

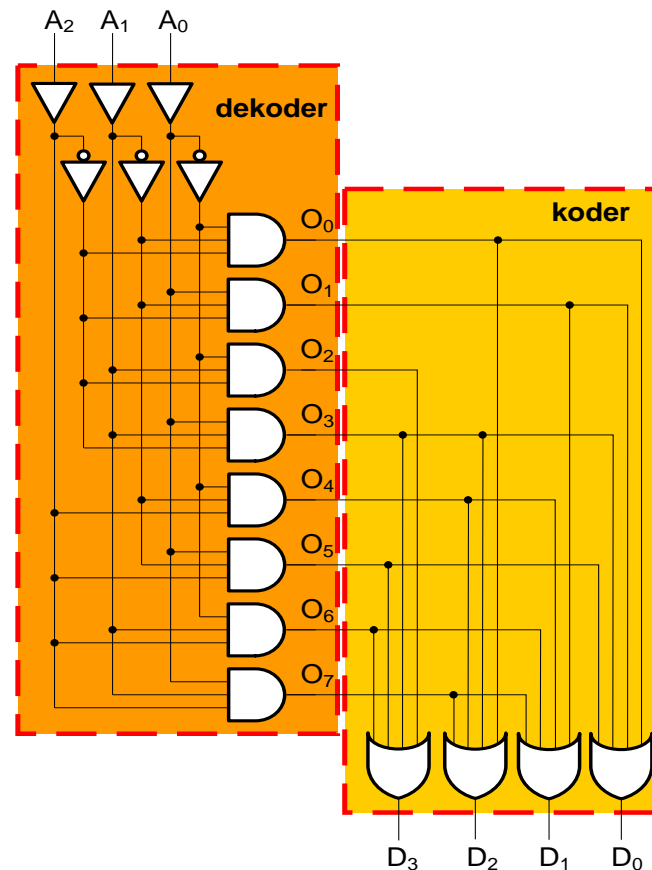
- s mogućnošću *višekratnog* programiranja i brisanja *električkim putem*, EEPROM, E²PROM:
 - izbjeći probleme EPROM:
 - *dugo* brisanje cijelog sadržaja u posebnom uređaju
 - programiranje *unutar* uređaja u kojem se koristi
 - EAROM: smanjen razmak G_1 i D :
 - upisivanje i brisanje podatka *tuneliranjem* (upis: $U_{G2D} \sim 10 \text{ V}$, brisanje: $U_{G2D} \sim -10 \text{ V}$)
 - $t_a \sim 250 \text{ ns}$
 - "brza memorija" (engl. flash memory, flash), >1984: brže "preprogramiranje" i pristup, veći kapacitet, veća trajnost upisa podataka; npr. USB stick/dongle

Izvedbe permanentne memorije

Primjer: ROM s 8 4-bitnih riječi
(ROM 8x4)

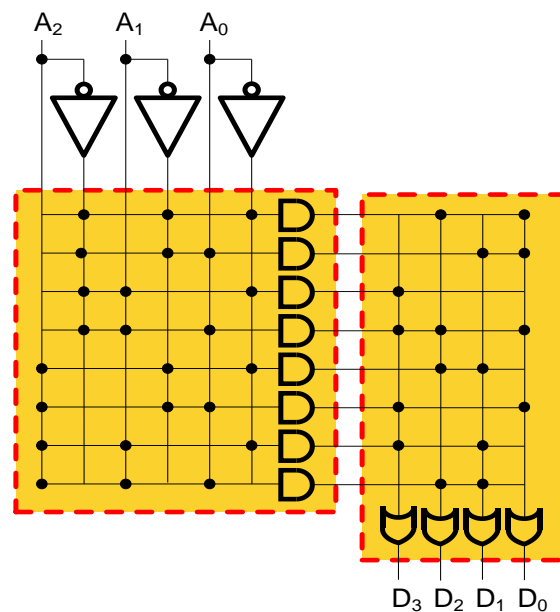
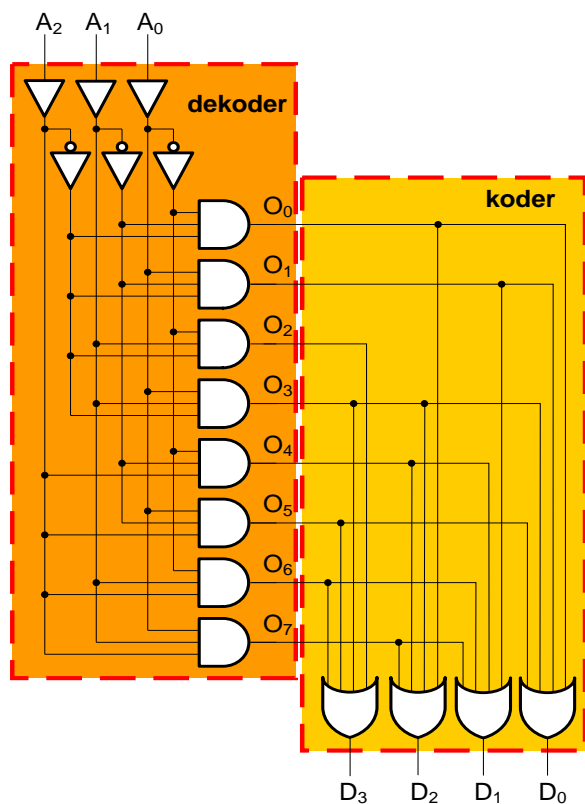


riječ	A ₂	A ₁	A ₀	D ₃	D ₂	D ₁	D ₀
0	0	0	0	0	1	0	1
1	0	0	1	0	0	1	1
2	0	1	0	1	0	0	0
3	0	1	1	1	1	0	1
4	1	0	0	0	1	1	0
5	1	0	1	1	0	0	1
6	1	1	0	1	0	1	0
7	1	1	1	0	1	1	0



Izvedbe permanentne memorije

- polje = matrica
~ *matrični prikaz* (engl. array-logic schematic)





Karakteristične primjene

- primjena ROM:
 - pohranjivanje značajnih podataka važnih za rad cjelokupnog digitalnog sustava (npr. računalo)
 - pohranjivanje sustavskih programa (npr. BIOS)
 - upravljačka memorija kod *mikroprogramiranja*
~ posebna izvedba upravljačke jedinice procesora
 - pretvorba koda
~ naročito generatori znakova za rasterske prikaze (zasloni, matrični pisači)
 - aritmetički sklopovi
~ izvedbe tablica posebnih funkcija (npr. trigonometrijske)
- *problem*
~ ROM je *sporiji*, jer ima više razina logike

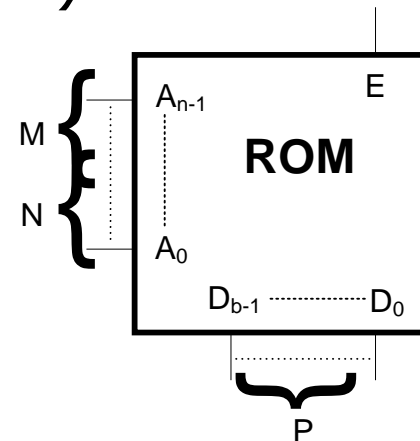
Zadatak: permantnom memorijom ostvariti slijedeće pretvornike koda:

- BCD u 2421
- BCD u 7-segmentni
- koji su parametri (broj riječi x broj bitova/riječ) potrebnih permanentnih memorija?

Karakteristične primjene

Primjer: sklop za množenje 8-bitnih brojeva

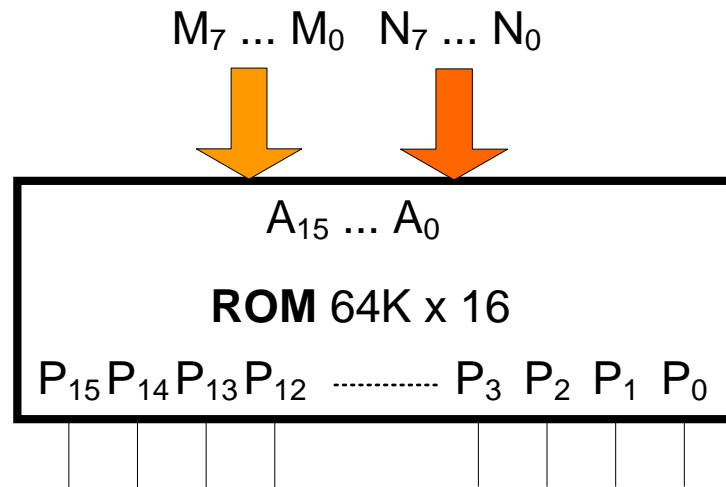
- tablica množenja ugrađena u ROM
~ *pregledna tablica* (engl. Look-Up Table, LUT):
 - faktori M , N se kombiniraju tako da tvore adresu memorijske lokacije u kojoj je pohranjen produkt P
- efikasnija izvedba:
 - *kombinacija* izvjesnog broja ROMova značajno manjeg kapaciteta (parcijalni produkti) i zbrajala
 - veća kašnjenja!



Karakteristične primjene

- množenje 8-bitnih brojeva
~ potreban kapacitet *prevelik*:

$$W = 2^n \cdot b = 2^{8+8} \cdot (8+8) = 2^{16} \cdot 2^4 = 2^{20} = 1Mbit$$



Karakteristične primjene

- "rastavljanje" multiplikanda i multiplikatora:

$$\begin{aligned}M &= m_7m_6m_5m_4\ 0000 + m_3m_2m_1m_0 \\ &= (m_7m_6m_5m_4) \cdot 2^4 + (m_3m_2m_1m_0) = m \cdot 2^4 + \Delta m\end{aligned}$$

$$\begin{aligned}N &= n_7n_6n_5n_4\ 0000 + n_3n_2n_1n_0 \\ &= (n_7n_6n_5n_4) \cdot 2^4 + (n_3n_2n_1n_0) = n \cdot 2^4 + \Delta n\end{aligned}$$

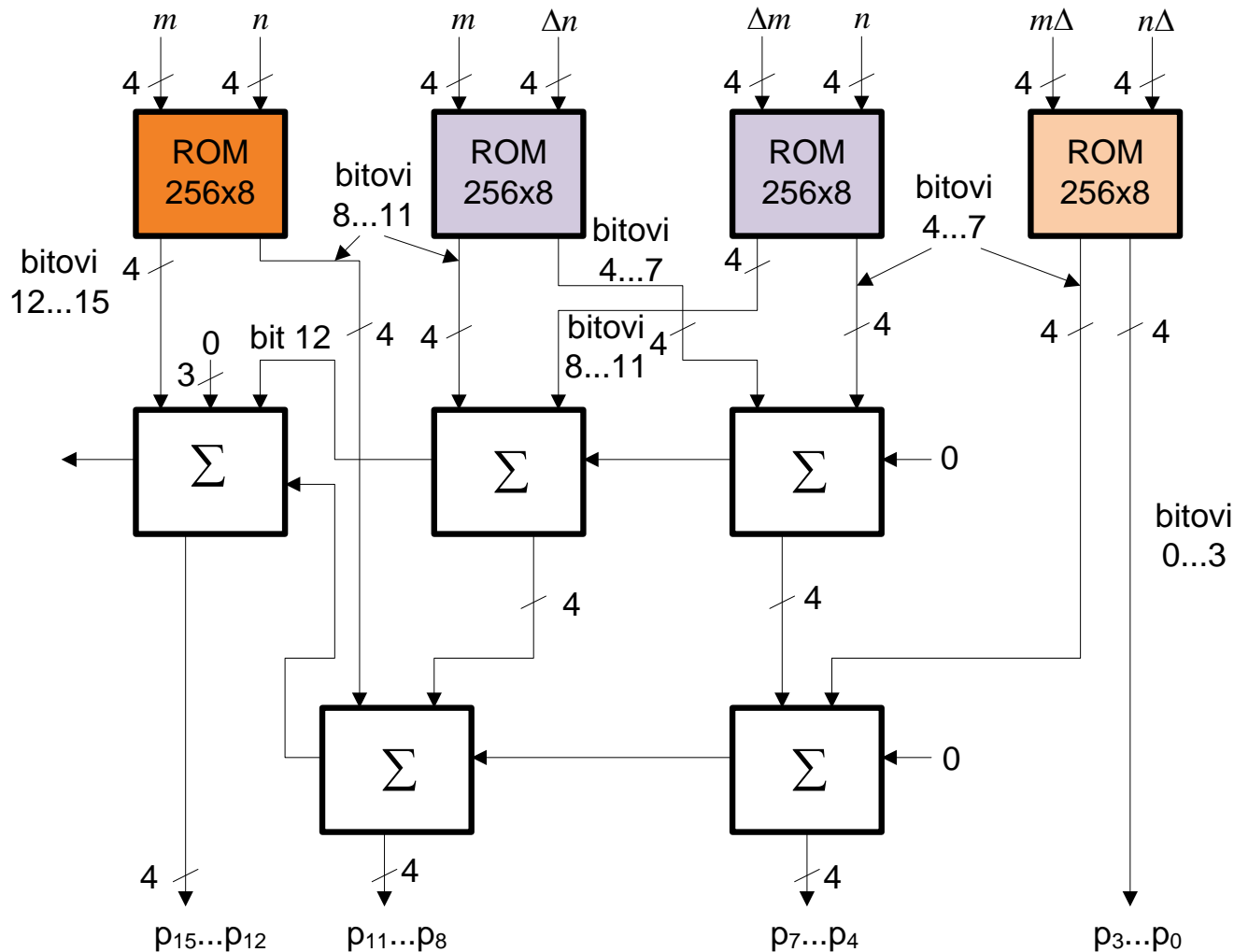
$$\begin{aligned}P &= M \cdot N \\ &= (m \cdot 2^4 + \Delta m) \cdot (n \cdot 2^4 + \Delta n) = \\ &= (m \cdot n) \cdot 2^8 + (\Delta m \cdot n + m \cdot \Delta n) \cdot 2^4 + \Delta m \cdot \Delta n\end{aligned}$$

- dovoljan puno manji kapacitet:

$$W' = 2^{4+4} \cdot (4 + 4) = 2Kbit; W_{ukupni} = 4 \cdot C' = 8Kbit$$

Karakteristične primjene

- sklop zasnovan na kompoziciji manjih ROMova:

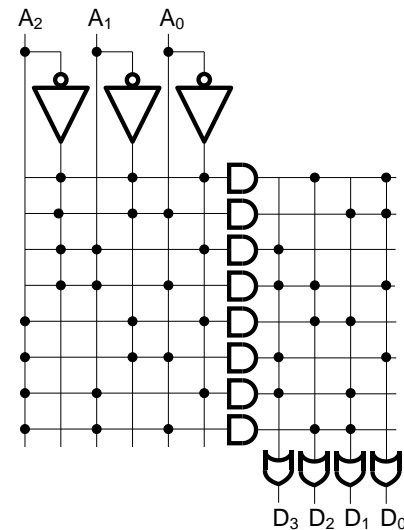


Generiranje Booleovih funkcija

- interpretacija podataka "pohranjenih" u ROM
~ *logičke funkcije* (više njih!):

- svaki izlaz
~ *jedna* logička funkcija
- *svi* izlazi
~ *višeizlazna* funkcija
- ROM
~ generator funkcija:

$$f_i(A_2, A_1, A_0) = D_i \quad 0 \leq i \leq 3$$



$$f_0 = \sum m(0,1,3,5)$$

$$f_1 = \sum m(1,4,6,7)$$

$$f_2 = \sum m(0,3,4,7)$$

$$f_3 = \sum m(2,3,5,6)$$

Generiranje Booleovih funkcija

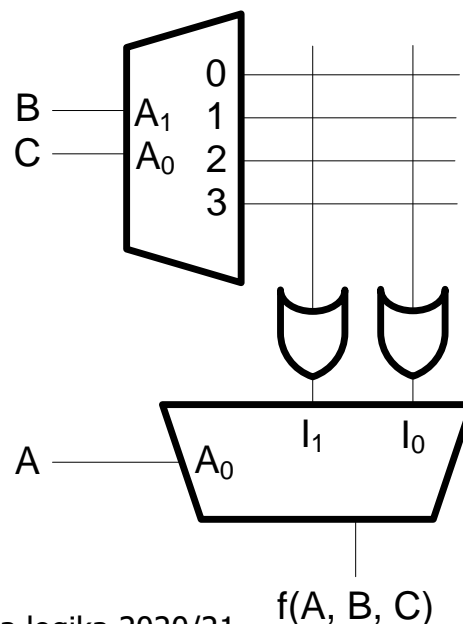
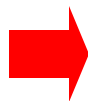
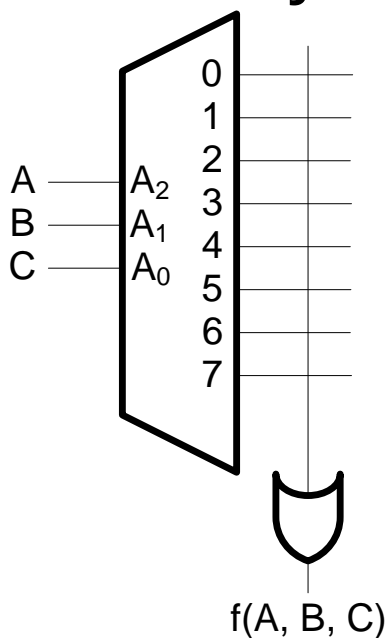
- zapažanje

~ za veći broj adresnih linija sklopovski povoljnije koristiti *raspodijeljeni dekod* (engl. split decoder):

- *manji* dekod

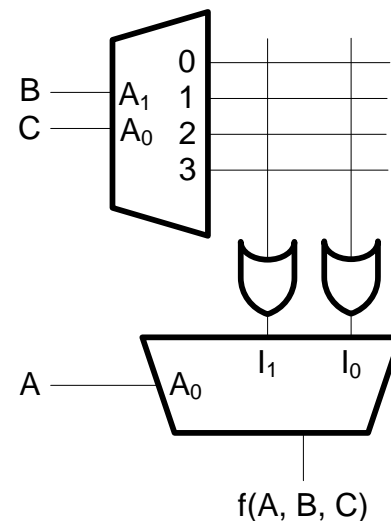
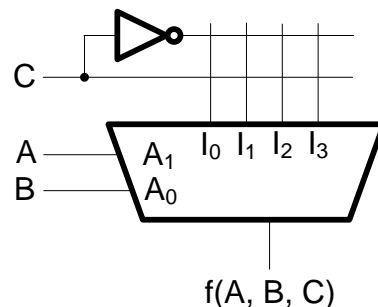
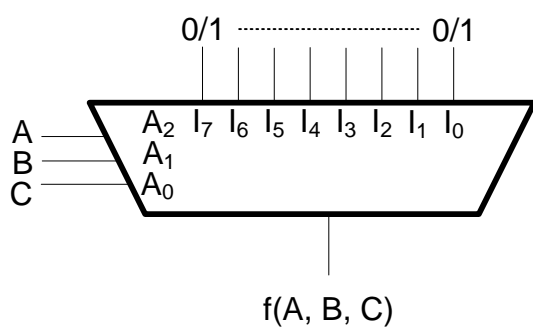
~ idealno dva *jednaka* dekodera = *podjela* tablice definicije funkcije f (parametar varijabla najviše težine)

- kombinacija ROM + MUX (\exists *unutarnji* dekod!



Generiranje Booleovih funkcija

- *raspodijeljeno* dekodiranje ROMa:
 - samo dekodер
 - dekodер + MUX (netrivijalne rezidualne funkcije)
 - dekodер + MUX (trivijalne rezidualne funkcije)
 - samo MUX (simulacija permanentne memorije)



Generiranje Booleovih funkcija

- ostvarivanja funkcija ROMom i multipleksorom
~ *Shannonova ekspanzija*:

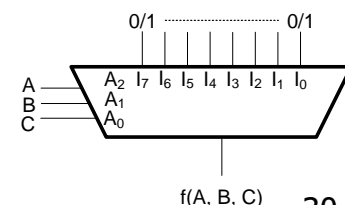
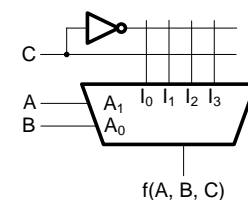
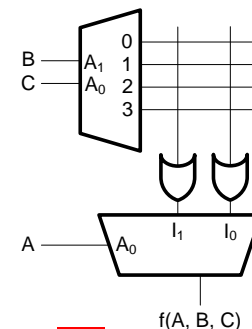
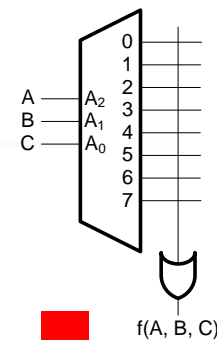
npr. $f(A, B, C)$

$$f(A, B, C) = f(A, B, C)$$

$$f(A, B, C) = f(0, B, C) \cdot \bar{A} + f(1, B, C) \cdot A$$

$$f(A, B, C) = f(0,0,C) \cdot \bar{A} \cdot \bar{B} + f(0,1,C) \cdot \bar{A} \cdot B \\ + f(1,0,C) \cdot A \cdot \bar{B} + f(1,1,C) \cdot A \cdot B$$

$$f(A, B, C) = f(0,0,0) \cdot \bar{A} \cdot \bar{B} \cdot \bar{C} + f(0,0,1) \cdot \bar{A} \cdot \bar{B} \cdot C \\ + f(0,1,0) \cdot \bar{A} \cdot B \cdot \bar{C} + f(0,1,1) \cdot \bar{A} \cdot B \cdot C \\ + f(1,0,0) \cdot A \cdot \bar{B} \cdot \bar{C} + f(1,0,1) \cdot A \cdot \bar{B} \cdot C \\ + f(1,1,0) \cdot A \cdot B \cdot \bar{C} + f(1,1,1) \cdot A \cdot B \cdot C$$



Generiranje Booleovih funkcija

- parcijalne funkcije kod Shannonove ekspanzije u kojima je neki od literala fiksiran (0 ili 1)
~ *kofaktori, rezidui* (ostaci), *rezidualne* funkcije
npr. $\varphi_0(C), \varphi_1(C), \varphi_2(C), \varphi_3(C)$

$$f(A, B, C) = f(0, 0, C) \cdot \bar{A} \cdot \bar{B} + f(0, 1, C) \cdot \bar{A} \cdot B \\ + f(1, 0, C) \cdot A \cdot \bar{B} + f(1, 1, C) \cdot A \cdot B$$

$$\varphi_0(C) = f(0, 0, C)$$

$$\varphi_1(C) = f(0, 1, C)$$

$$\varphi_2(C) = f(1, 0, C)$$

$$\varphi_3(C) = f(1, 1, C)$$

Generiranje Booleovih funkcija

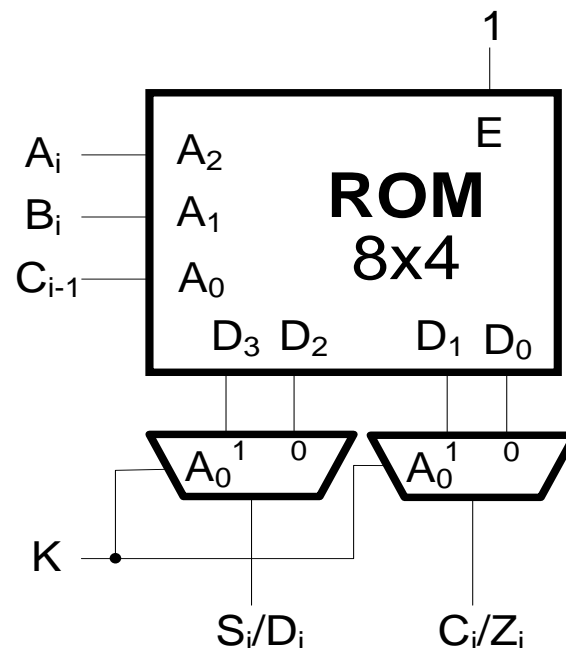
Primjer : potpuno zbrajalo/odbijalo

- izvedba s ROMom 8x4 i 2 MUXa 2/1

- K=0: zbrajalo, K=1: odbijalo

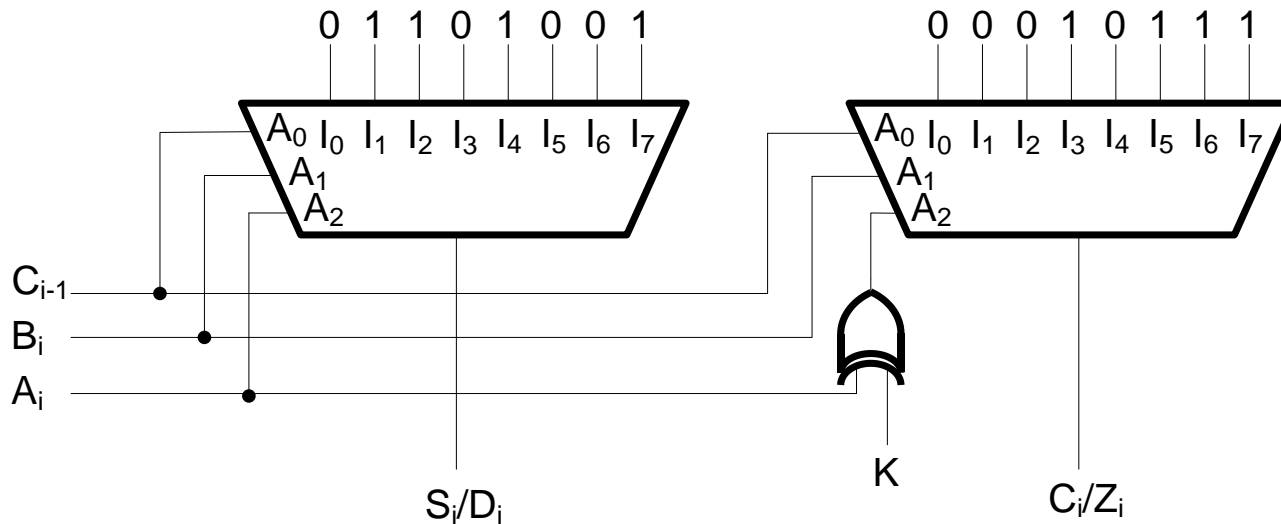
- uočiti: $S_i = D_i$, simetrija C_i i Z_i

A_2	A_1	A_0	D_2	D_0	D_3	D_1
A_i	B_i	C_{i-1}	S_i	C_i	D_i	Z_i
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1



Generiranje Booleovih funkcija

- izvedba s 2 MUXa 8/1 u funkciji ROMova
 - K=0: zbrajalo, K=1: odbijalo





Sadržaj predavanja

- koncept programirljivih modula
- permanentna memorija
- **programirljivo logičko polje**
 - **funkcionalnost i struktura**
 - **tehnologija izvedbe**
 - **generiranje Booleovih funkcija**
- poluprogramirljivo logičko polje
- složeni programirljivi moduli
- programirljivo polje logičkih blokova

Programirljivo logičko polje

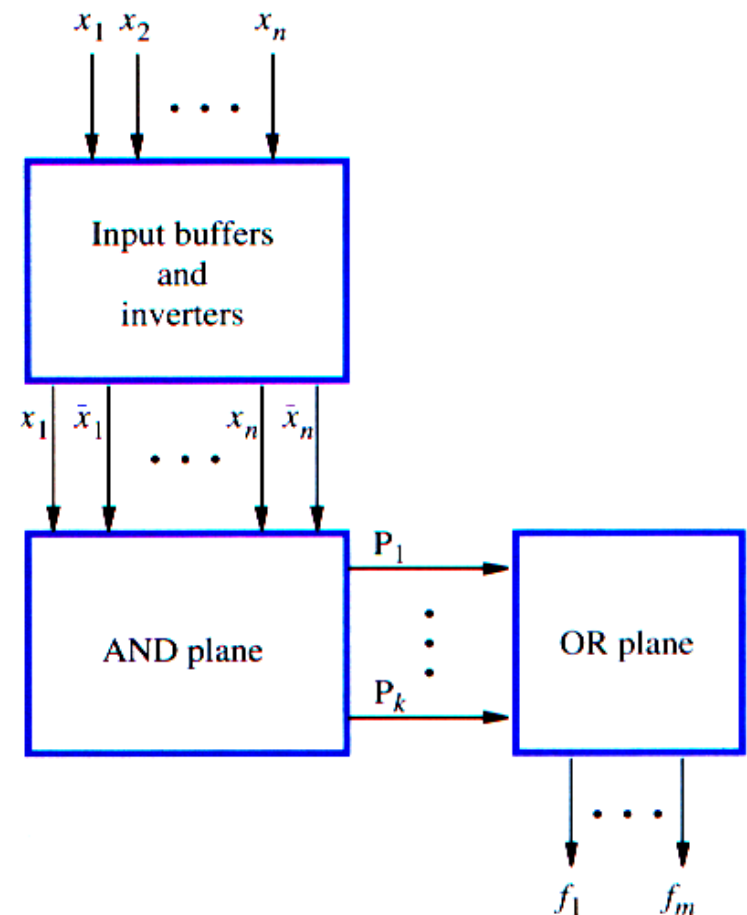
- zapažanje
 - ~ pri generiranju funkcija je spojno polje koda ROMa "slabo popunjeno":
 - obično su funkcije *nepotpuno specificirane*
~ ne koriste se *sve* kombinacije ulaza koda
 - ostvariti uštedu
 - ~ puno manji broj riječi (odgovara mintermima!)
od onog omogućenog dekoderom ($W \ll 2^n$)
 - relativno mali broj kombinacija ulaza
 - ~ obuhvatiti upravo one potrebne!
 - *nepotpuno* dekodiranje
 - ~ potrebno programirati i deko
 - ulazi u koder
 - ~ *primarni implikanti* (produkti) P_i umjesto minterma

Programirljivo logičko polje

- *programirljivo logičko polje*, PLA
(engl. Programmable Logic Array)
~ posebna logička struktura s *oba programirljiva* polja:
 - efikasno i fleksibilno rješenje
 - generirane funkcije su u obliku *sume produkata*
 - ostvarivanja funkcija ograničeno veličinom I polja (brojem P_i)
~ svakako provesti *minimizaciju*!
 - ako višeizlazna funkcija
~ što više P_i *zajedničkih* za što veći broj funkcija:
združena minimizacija

Programirljivo logičko polje

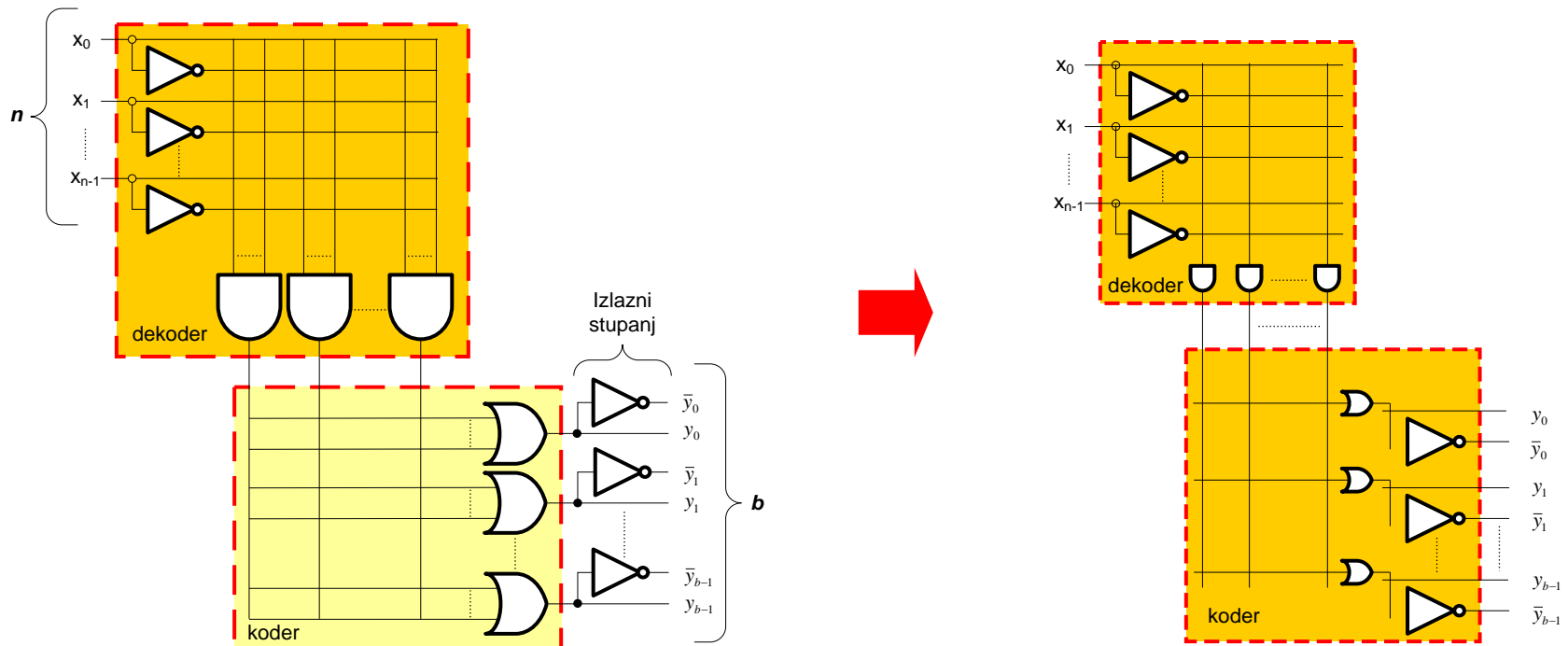
- izvedba PLA:
 - odvojni sklopovi na ulazu
~ generiranje $x_i, \bar{x}_i, \forall i$
 - tipični parametri:
 $n \times k \times m = 16 \times 32 \times 8$
 n = broj ulaza
 k = broj produkata
 m = broj izlaza



Programirljivo logičko polje

- struktura PLA:

- izlaz
~ funkcija i *komplement*
- ušteda izvoda
~ izlazni inverter ostvaren *upravljanim* EX-ILI sklopom (funkcija *i* ili komplement funkcije)

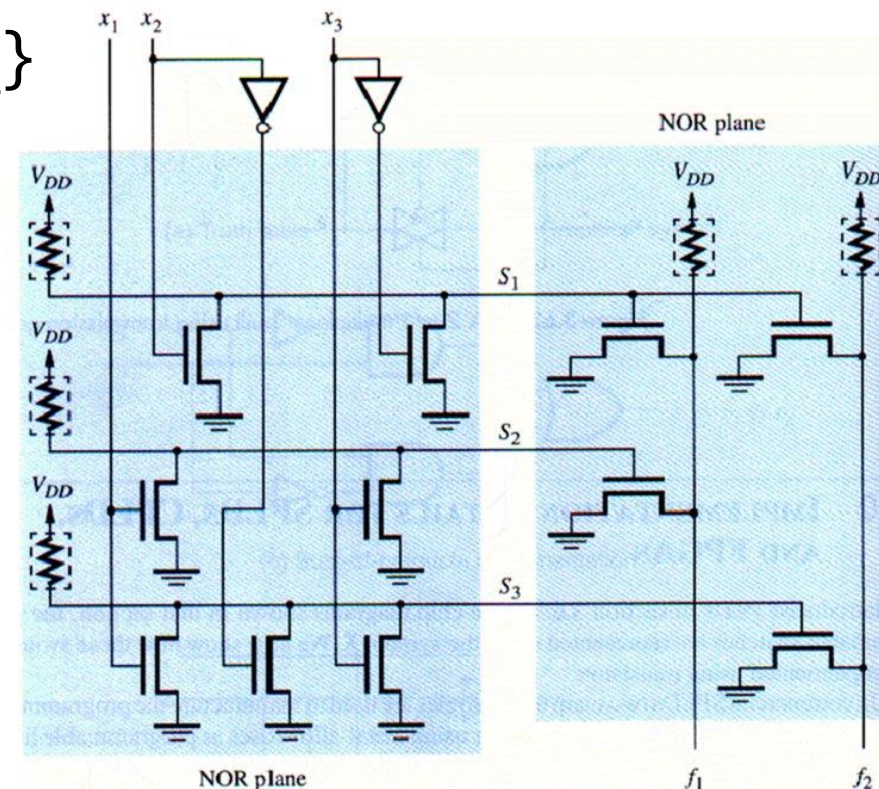


Izvedbe programirljivog logičkog polja

Primjer: izvedba PLA na razini tranzistora (MOSFET)

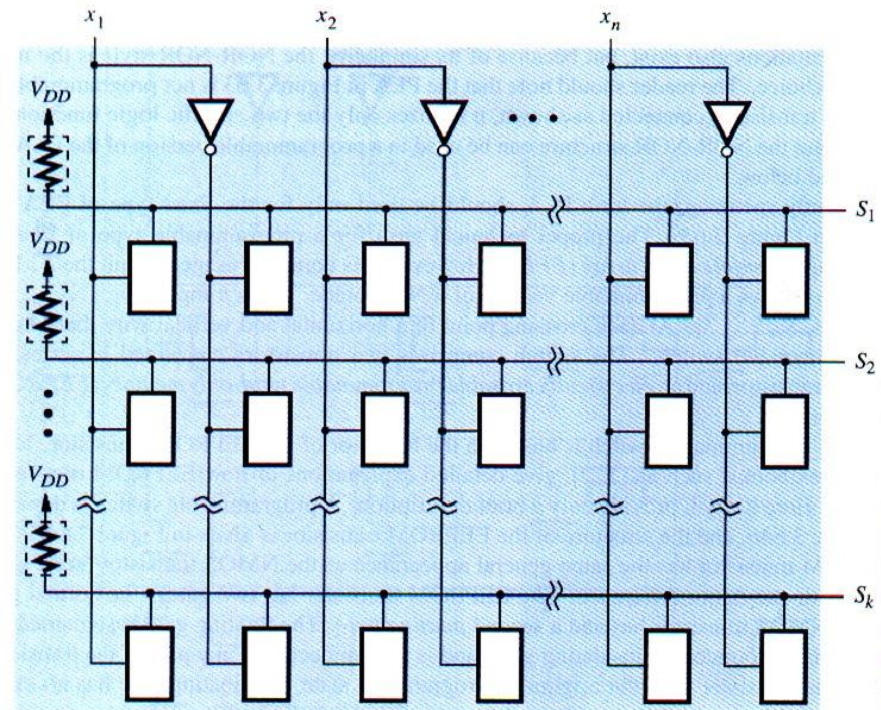
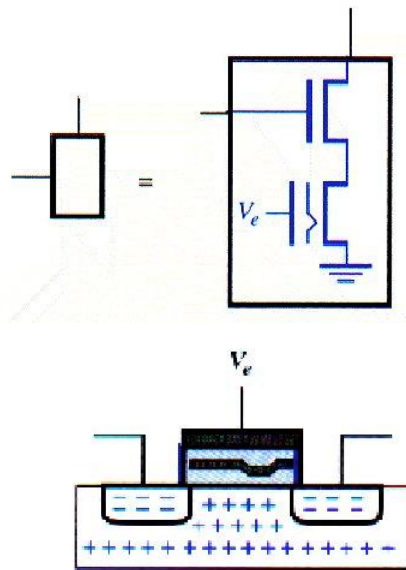
- programiranje u tvornici
- implementacija dvorazinske funkcije tipa NILI-NILI (NILI \sim paralelni NMOSovi)
- višezlazna funkcija: $\{f_1, f_2\}$

$$\begin{aligned}f_1 &= \overline{S_1 + S_2} \\&= \overline{S_1} \cdot \overline{S_2} \\&= (x_2 + \overline{x_3}) \cdot (x_1 + \overline{x_3}) \\f_2 &= \overline{S_1 + S_3} \\&= \overline{S_1} \cdot \overline{S_3} \\&= (x_2 + \overline{x_3}) \cdot (x_1 + \overline{x_2} + x_3)\end{aligned}$$



Izvedbe programirljivog logičkog polja

- izvedba FPLA (engl. Field-Programmable Logic Array):
 - (višekratno) programiranje "na licu mjesta"
 - NMOS Tr + programirljiva sklopka
 - programirljiva sklopka \sim EEPROM



Izvedbe programirljivog logičkog polja

Primjer: ostvarenje funkcija s FPLA

- : neprogramirana sklopka, *ima* spoja
- x: programirana, *nema* spoja

$$f_1 = \overline{S_1 + S_2 + S_3}$$

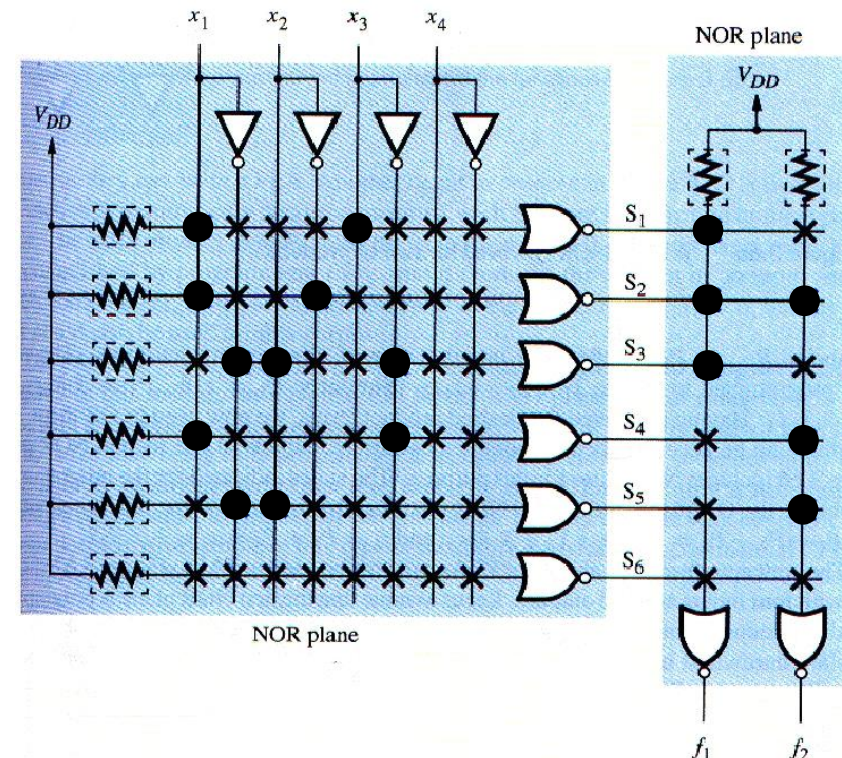
$$= \overline{S_1} \cdot \overline{S_2} \cdot \overline{S_3}$$

$$= (x_1 + x_3) \cdot (x_1 + \overline{x_2}) \cdot (\overline{x_1} + x_2 + \overline{x_3})$$

$$f_2 = \overline{S_2 + S_4 + S_5}$$

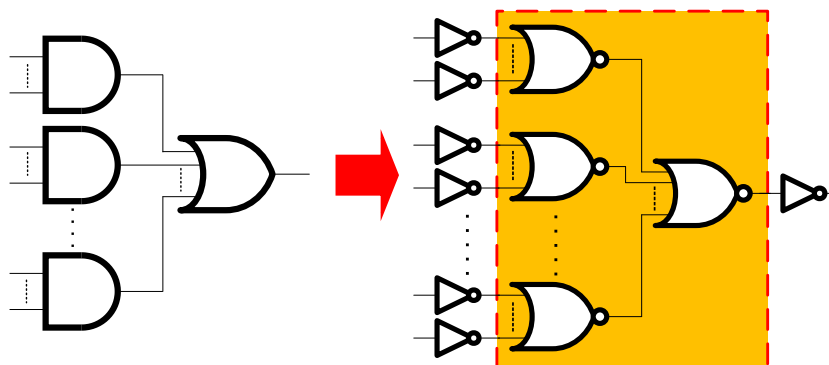
$$= \overline{S_2} \cdot \overline{S_4} \cdot \overline{S_5}$$

$$= (x_1 + \overline{x_2}) \cdot (x_1 + \overline{x_3}) \cdot (\overline{x_1} + x_2)$$



Izvedbe programirljivog logičkog polja

- implementacija dvorazinske funkcije tipa ILI-I s FPLA:
 - invertirati ulaze (x_1, \dots, x_4) ~ de Morgan
 - invertirati izlaze (f_1, f_2)

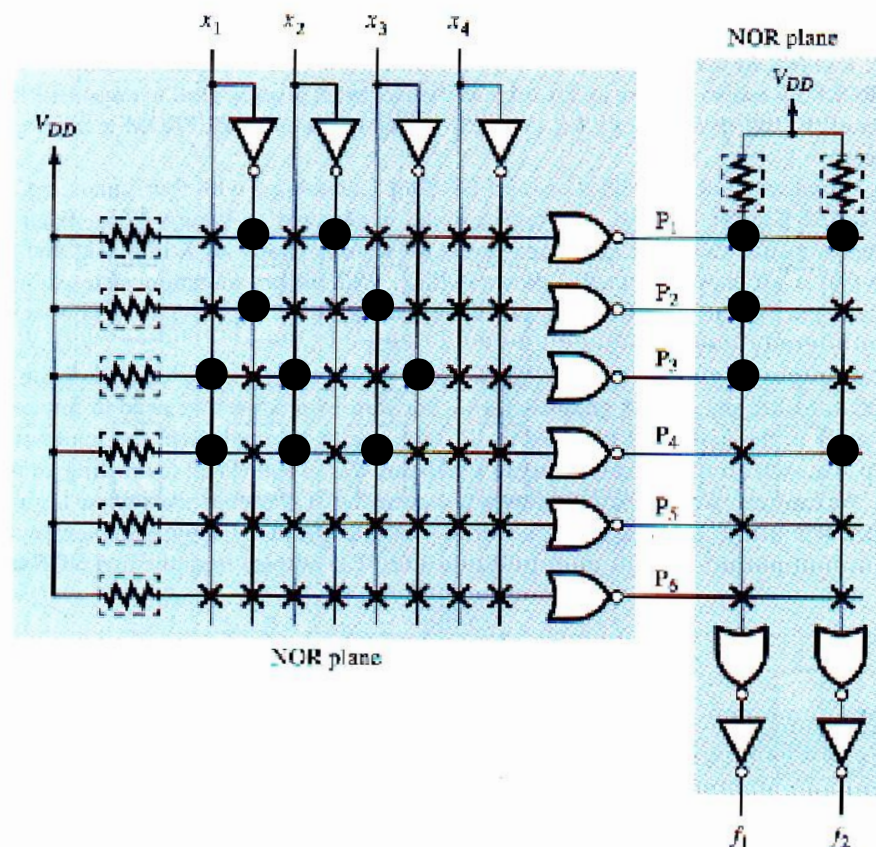


$$f_1 = P_1 + P_2 + P_3$$

$$= x_1 \cdot x_2 + x_1 \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot x_3$$

$$f_2 = P_1 + P_4$$

$$= x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$$





Generiranje Booleovih funkcija putem PLA

- tipično *više* funkcija nad istim skupom varijabli
~ *višeizlazna* funkcija
- ograničen broj produktnih članova
~ *minimizacija višeizlazne funkcije*:
pronaći što više *zajedničkih* primarnih implikanata,
neovisno o tome jesu li optimalni za pojedine
komponentne funkcije
- posebne metode minimizacije;
npr. K-tablice, modificirana Quine-McCluskeyeva metoda

Generiranje Booleovih funkcija putem PLA

Primjer: korištenjem PLA ostvariti višezlaznu funkciju

$$f_0 = AB + AC = AC + ABC\bar{C}$$

$$f_1 = \bar{A}B + B\bar{C} = \bar{A}B + ABC\bar{C}$$

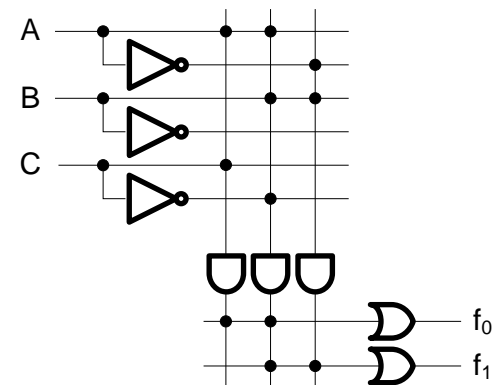
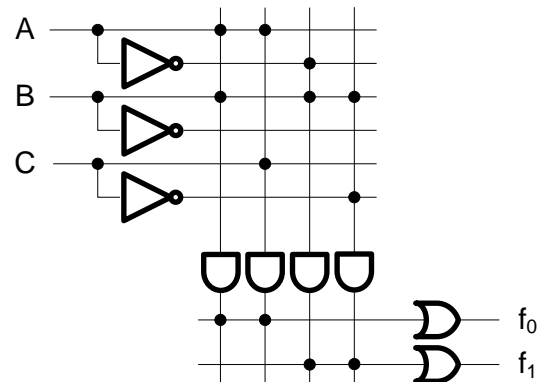
$f_0(A,B,C)$		AB			
		00	01	11	10
C	0			1	
	1			1	1

$f_1(A,B,C)$		AB			
		00	01	11	10
C	0		1	1	
	1		1		

NE!



DA!



Generiranje Booleovih funkcija putem PLA

Zadatak:

- primjerenim sklopom PLA ostvariti:
 - potpuno zbrajalo
 - potpuno zbrajalo/odbijalo
 - BCD zbrajalo
- primjenom K-tablica i metode Quine-McCluskey sklopom PLA ostvariti višezlaznu funkciju:
 - $f_1(A, B, C, D) = \sum m(0,1,2,3,6,7)$
 $f_2(A, B, C, D) = \sum m(0,1,6,7,14,15)$
 $f_3(A, B, C, D) = \sum m(0,1,2,3,8,9)$
 - $f_1(A, B, C, D, E) = \sum m(0,1,2,3,6,7,20,21,26,27,28)$
 $f_2(A, B, C, D, E) = \sum m(0,1,6,7,14,15,16,17,19,20,24,27)$
 $f_3(A, B, C, D, E) = \sum m(0,1,2,3,8,9,16,20,26,28,30)$



Sadržaj predavanja

- koncept programirljivih modula
- permanentna memorija
- programirljivo logičko polje
- **poluprogramirljivo logičko polje**
 - **funkcionalnost i struktura**
 - **generiranje Booleovih funkcija**
- složeni programirljivi moduli
- programirljivo polje logičkih blokova

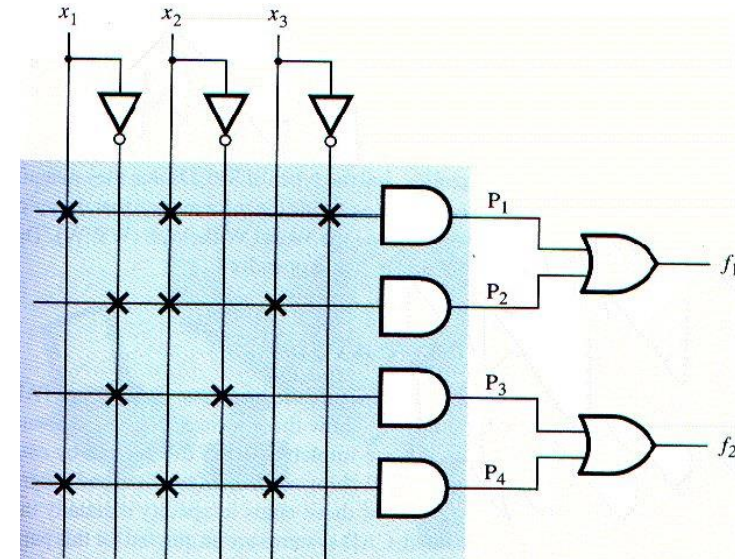


Poluprogramirljivo logičko polje

- problemi s PLA:
 - poteškoće s programirljivim sklopkama:
 - teško ih je ispravno proizvesti
 - redukcija brzine rada sklopa ostvarenog s PLA
 - ograničiti mogućnost programiranja na *samo jedno* polje:
 - jednostavnije za proizvodnju
 - jeftinije
 - bolje performanse
 - smanjena fleksibilnost

Poluprogramirljivo logičko polje

- *poluprogramirljivo polje*, PAL (engl. Programmable Array Logic)
~ jako popularno rješenje:
 - programira se *samo* I polje
~ *radikalno* ograničenje
broja P_i po funkciji f_j
 - komercijalne izvedbe:
~ 1000 programirljivih sklopki
 - programiranje CAD sustavom
 - kompenziranje reducirane funkcionalnosti
~ proizvodnja u širokom rasponu veličina
(broj ulaza i izlaza, broj ulaza u ILI sklopove)

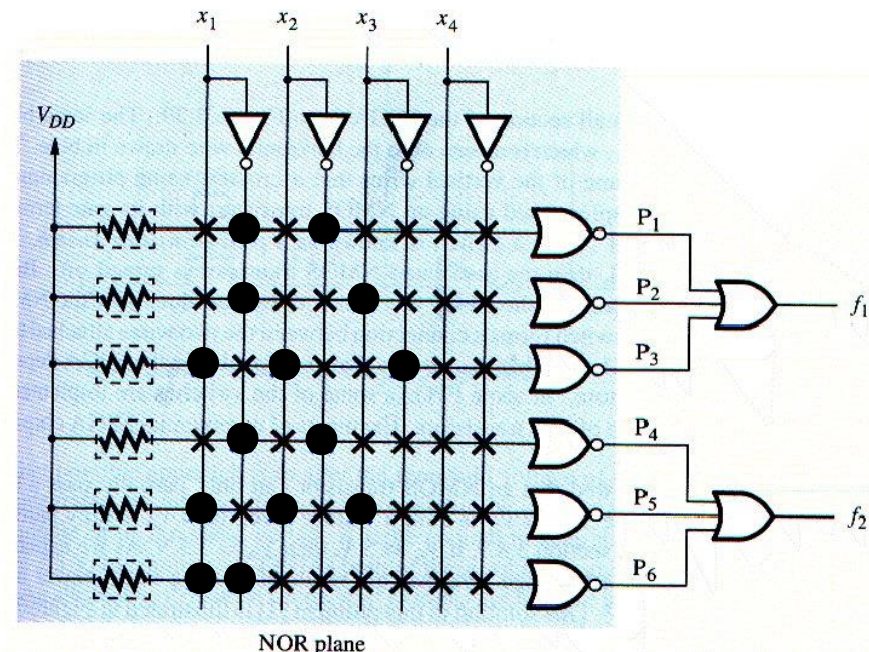


Generiranje Booleovih funkcija

Primjer: ostvarenje funkcije s PAL

- *nema* višestrukog korištenja P_i
(npr. $P_1 = P_4 = x_1 \cdot x_2$)
~ ne koristi se minimiziranje višezlaznih funkcija!!!
- nekorišteni P_i programira se da daje 0 (da ne smeta!):
npr. $P_6 = x_1 \cdot x_1 = 0$

$$\begin{aligned} f_1 &= P_1 + P_2 + P_3 \\ &= x_1 \cdot x_2 + x_1 \cdot x_3 + x_1 \cdot x_2 \cdot x_3 \\ f_2 &= P_4 + P_5 \\ &= x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_1 \\ &= x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + 0 \\ &= x_1 \cdot x_2 + x_1 \cdot x_2 \cdot x_3 \end{aligned}$$



Generiranje Booleovih funkcija

- povećanje fleksibilnosti
~ *makroćelija* (engl. macrocell):

- dodatno sklopovlje
na izlazu ILI sklopa:

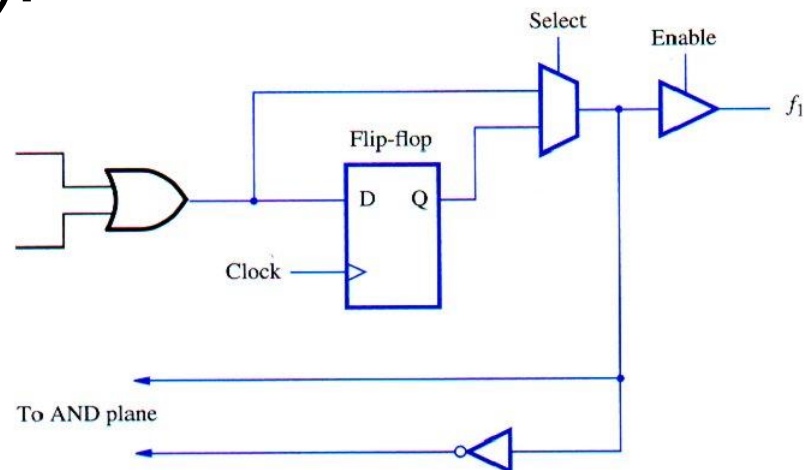
- bistabil
~ sekvencijsko ponašanje:

$$Q^{n+1} = D^n$$

- multipleksor
~ odabir ponašanja (kombinacijsko/sekvenzijsko):

$$\begin{aligned} f_1 &= D \cdot \overline{Select} + Q \cdot Select \\ &= D^n \cdot \overline{Select} + D^{n-1} \cdot Select \end{aligned}$$

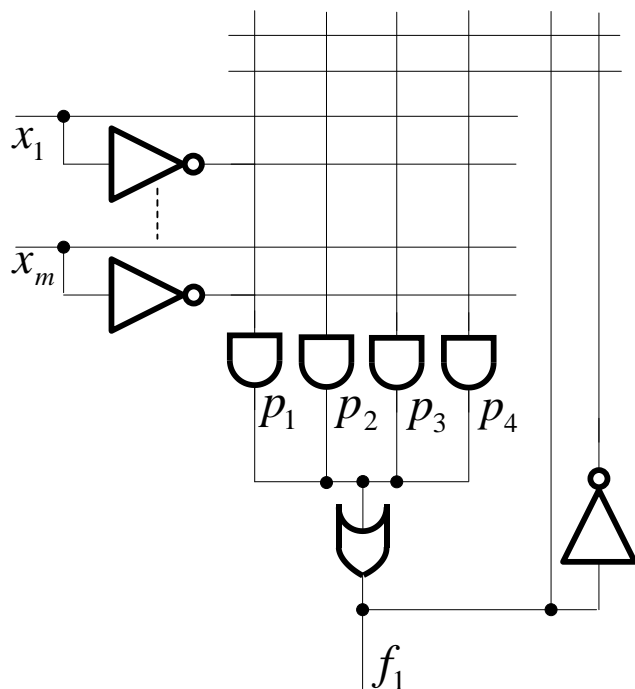
- povratna veza na I polje
~ ostvarivanja složenijih (~ *višerazinskih*) sklopova
istim sklopom PAL



Generiranje Booleovih funkcija

Primjer: potpuno zbrajalo/odbijalo ostvareno makroćelijom

- $K=0$: zbrajalo, $K=1$: odbijalo
- uočiti: $S_i = D_i$, simetrija C_i i Z_i
- zadana primitivna makroćelija

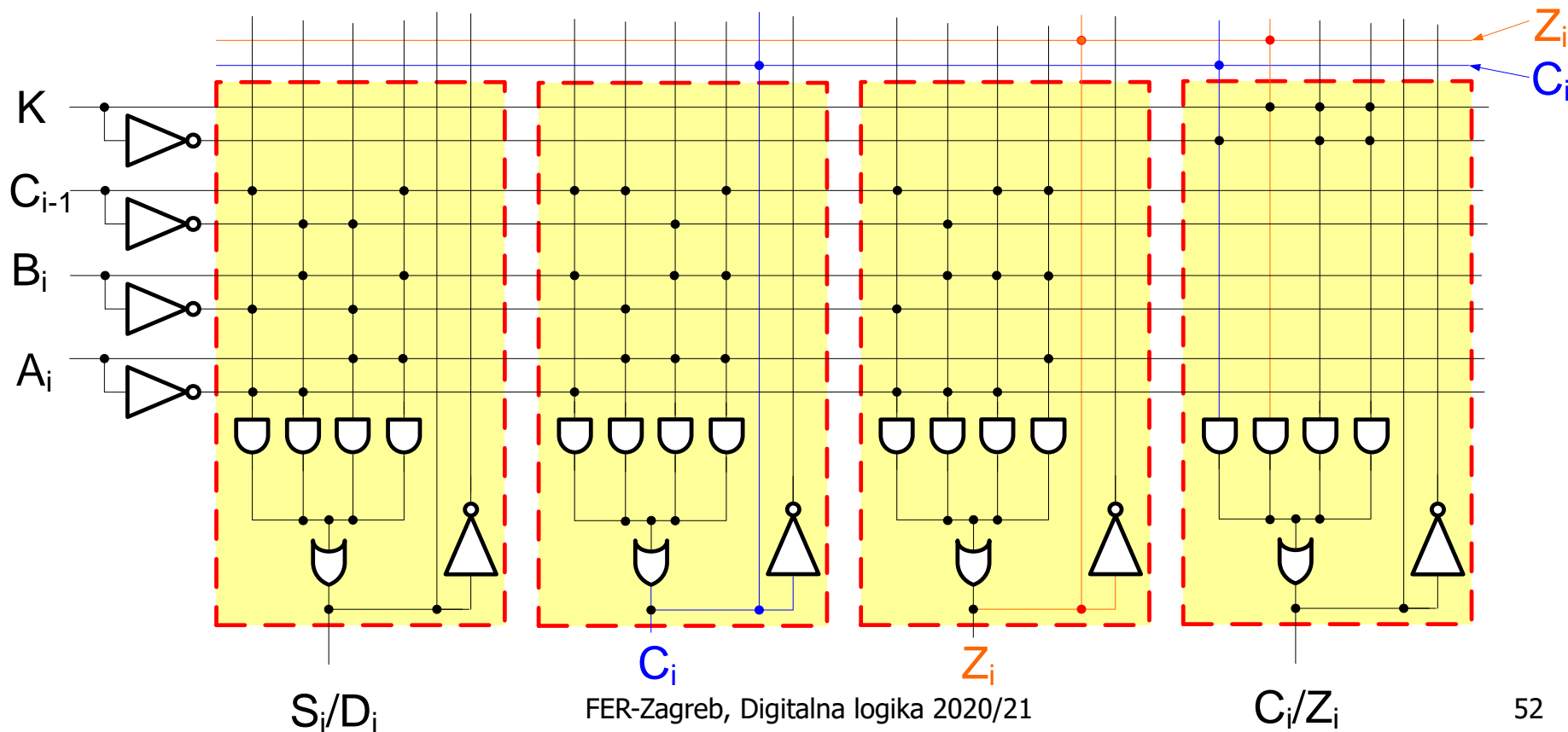


A_i	B_i	C_{i-1}	S_i	C_i	D_i	Z_i
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Generiranje Booleovih funkcija

- potrebne su *četiri* makroćelije (zašto?)

A_i	B_i	C_{i-1}	S_i	C_i	D_i	Z_i
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1





Sadržaj predavanja

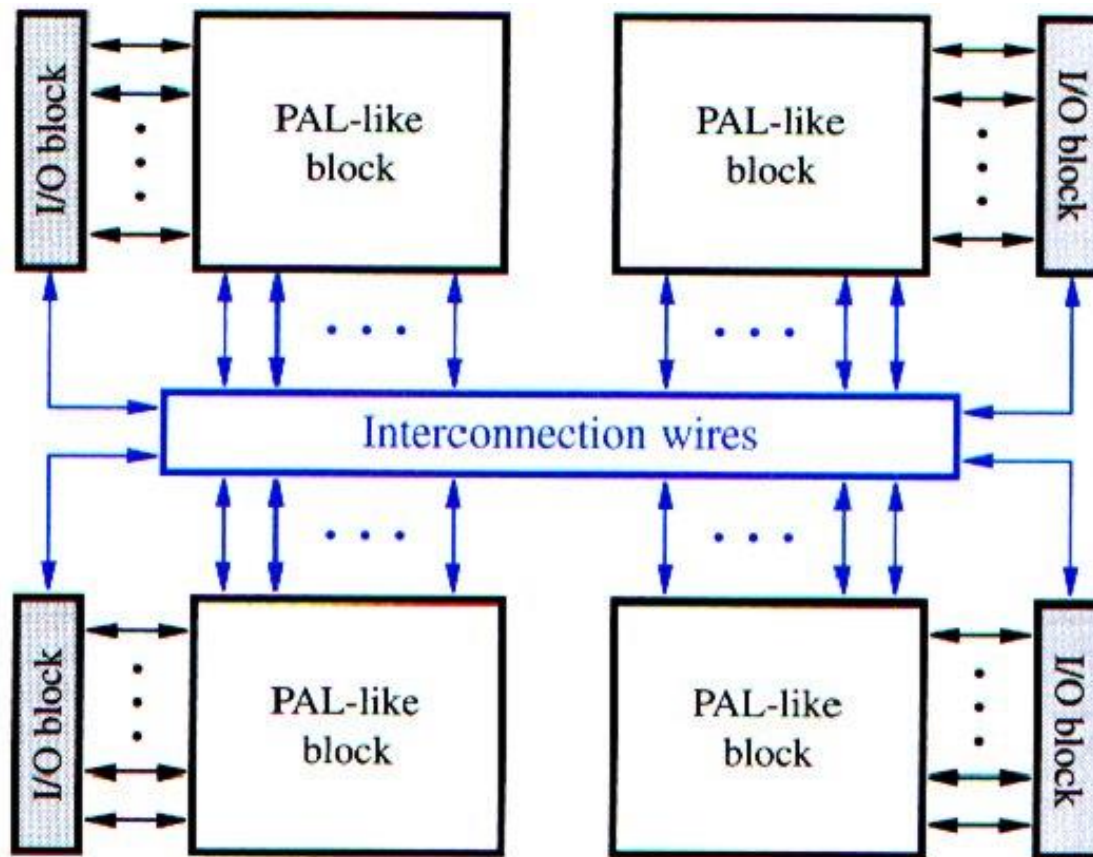
- koncept programirljivih modula
- permanentna memorija
- programirljivo logičko polje
- poluprogramirljivo logičko polje
- **složeni programirljivi moduli**
- programirljivo polje logičkih blokova

Složeni programirljivi moduli

- programirljivi moduli:
 - SPLD (= PLA, PAL)
~ skromne dimenzije sklopa
 - *složeni PLD, CPLD*:
 - *više* blokova sa sklopovljem (~ *više* SPLD)
 - mogućnost internog povezivanja blokova
 - tipične dimenzije:
 - 2÷100 "PALu sličnih blokova "
 - 16 makroćelija u "PALu sličnom bloku "
 - 5÷20 ulaza u ILI sklopove (to su P_i)
 - EX-ILI za programirljivo komplementiranje izlaza
 - izlaz iz makroćelije
~ sklop s tri stanja

Složeni programirljivi moduli

- struktura CPLD:

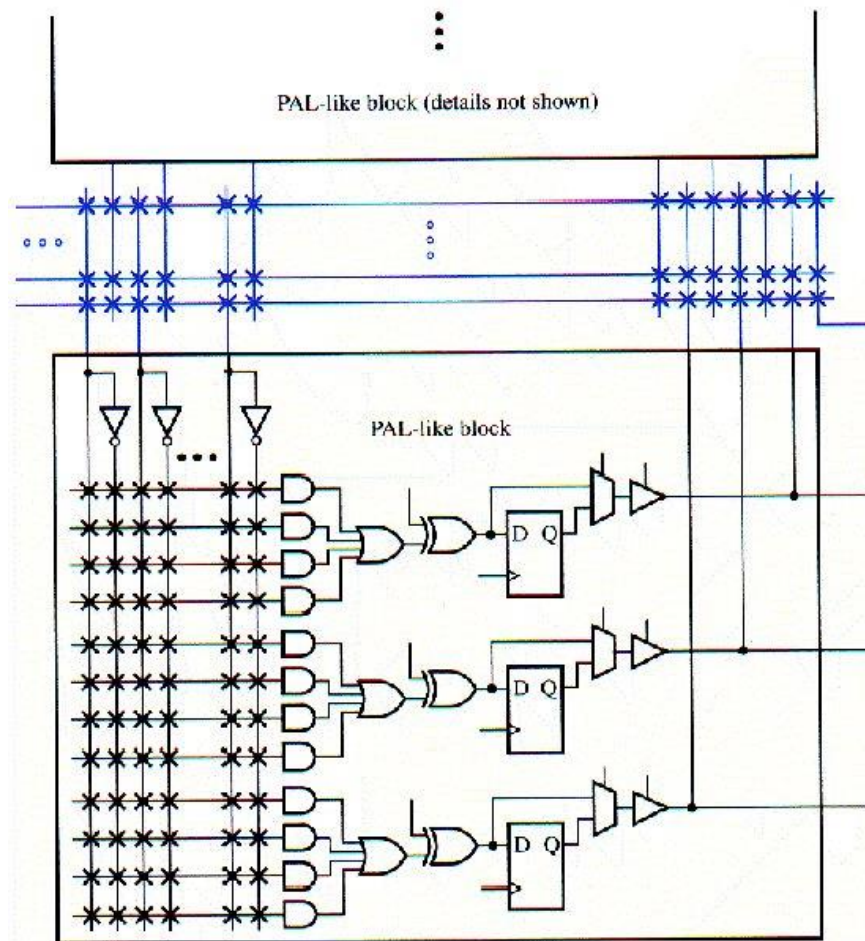


Složeni programirljivi moduli

- struktura CPLD:
 - *PALu slični blokovi* (engl. PAL-like blocks)
~ ostvarenje primitivnije funkcije
 - *UI blokovi*
~ sučelje za svaki PALu slični blok
 - *povezno ožičenje* (engl. interconnection wires):
 - programirljive sklopke za povezivanje PALu sličnih blokova
 - broj programirljivih sklopki:
~ pažljiva procjena! (fleksibilnost ↔ efikasnost)
 - vertikalne linije
~ ulazi u makroćeliju i izlazi iz nje
 - problem:
~ ako je izvod IC korišten kao izlaz,
pripadna je makroćelija neupotrebljiva

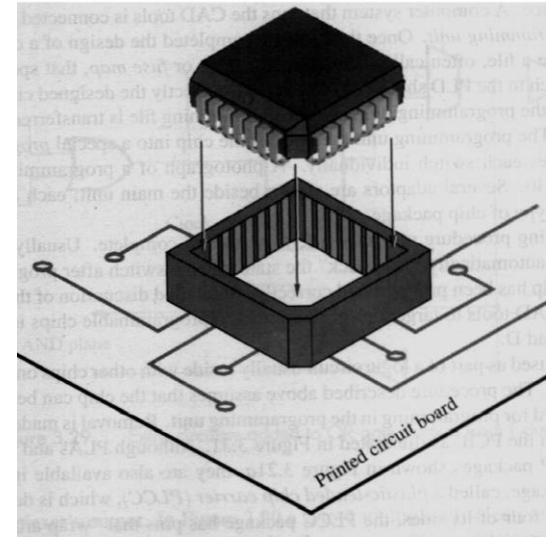
Složeni programirljivi moduli

- struktura PALu sličnog bloka:



Složeni programirajući moduli

- programiranje CPLD
 - ~ tipično *u sustavu u kojem se koriste* (engl. In-System Programming, ISP):
 - mehanički razlozi:
 - ~ IC s velikim brojem ($\sim 100 \div 200$) krhkih i savitljivih izvoda
 - *podnožja* (engl. sockets) su sumjerljivo skupa
 - programiranje putem *posebnog konektora* povezanog na sve CPLD u sustavu
 - ~ *JTAG pristup* (engl. JTAG port, Joint Test Action Group), standard IEEE
 - *stalno* (engl. nonvolatile) programiranje
 - primjena CPLD u $\sim 50\%$ slučajeva korištenja PLD





Sadržaj predavanja

- koncept programirljivih modula
- permanentna memorija
- programirljivo logičko polje
- poluprogramirljivo logičko polje
- složeni programirljivi moduli
- **programirljivo polje logičkih blokova**
 - **struktura i logički blokovi**
 - **generiranje Booleovih funkcija**

Programirljivo polje logičkih blokova

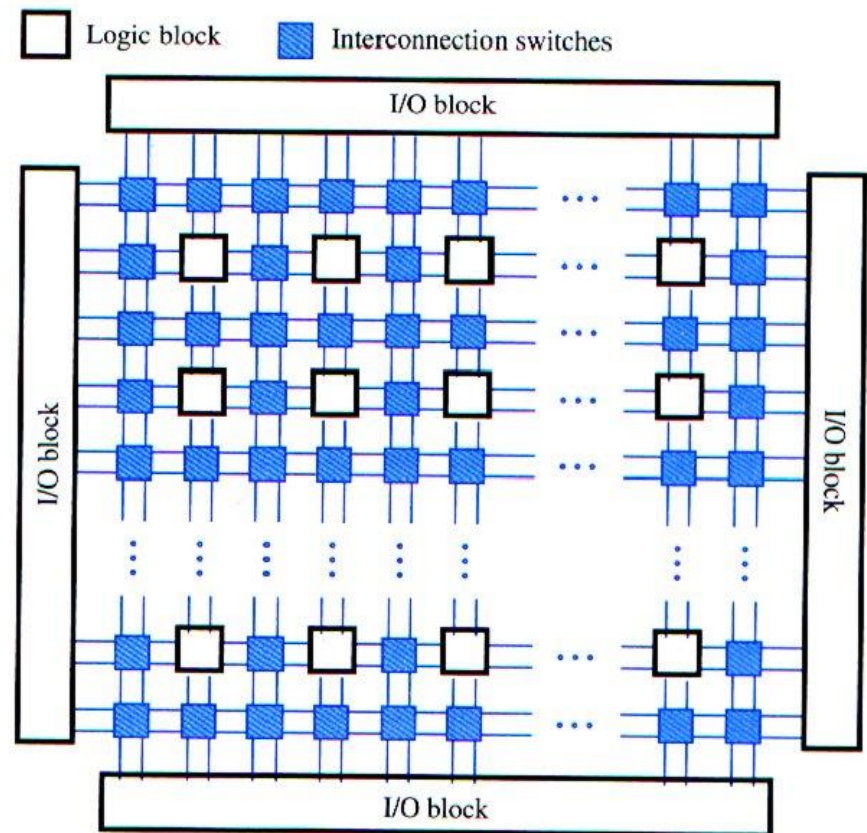
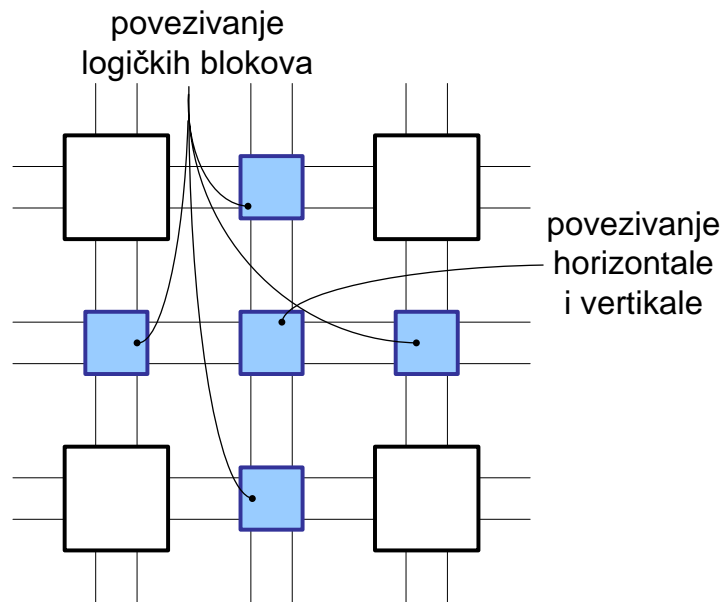
- mjera složenosti digitalnog sklopa
~ broj *ekvivalentnih sklopova* (engl. equivalent gates):
ukupni broj NI sklopova s 2 ulaza koji bi bili
potrebni za njegovo ostvarenje
- SPLD/CPLD podržavaju ostvarenja
relativno *jednostavnijih* sklopova:
 - makroćelije SPLD/CPLD: ~20 ekvivalentnih sklopova
npr. PAL s 8 makroćelija: ~160 ekvivalentnih sklopova
CPLD s 1000 makroćelija: ~20.000 ekvivalentnih
sklopova
 - želja za smanjenjem troškova i povećanjem performansi
~ sklop sa *što manjim* brojem IC

Programirljivo polje logičkih blokova

- *programirljivo polje logičkih blokova*
(engl. Field Programmable Gate Array, FPGA)
 - PLD za ostvarivanje *relativno velikih* digitalnih sklopova (≥ 100.000 ekvivalentnih sklopova)
 - *logički blokovi* (engl. Logic Blocks, LB; Configurable Logic Blocks, LB;) umjesto I/ILI polja
 - tipična struktura FPGA:
 - LB organizirani u *dvodimenzijско* polje
 - *UI blokovi* za sučelje (s izvodima IC)
 - međusobno povezivanje LBova vodovima i programirljivim sklopkama
~ horizontalni i vertikalni *kanali za usmjeravanje* (engl. routing channels) između redaka i stupaca LBova

Programirljivo polje logičkih blokova

- općenita struktura FPGA:
 - bijela polja ~ logički blokovi (LBovi)
 - plava polja ~ programirljive sklopke



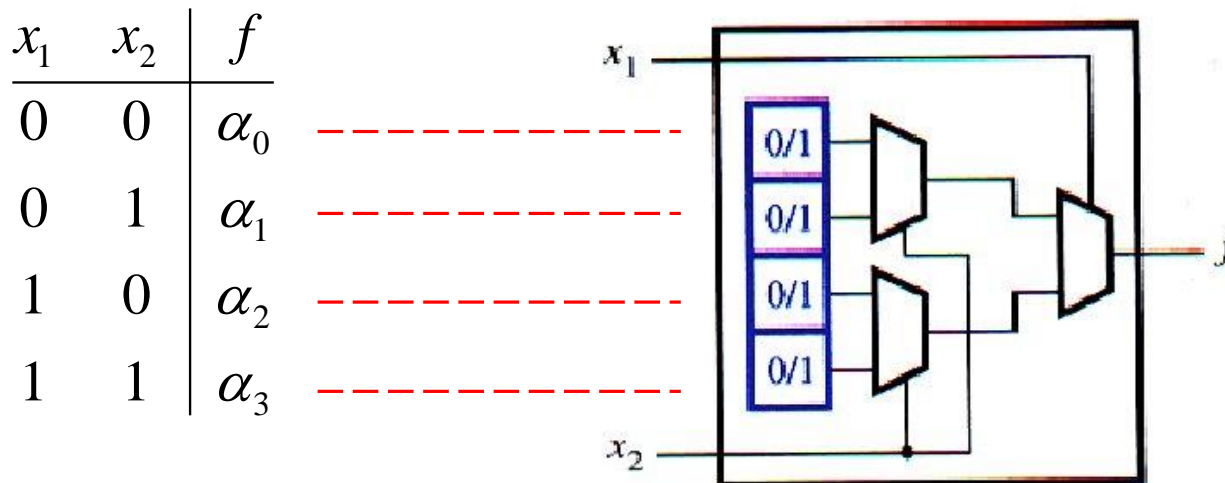
Programirljivo polje logičkih blokova

- općenita struktura *logičkog bloka*:
 - više ulaza, jedan izlaz
~ ostvaruje *jednostavnu* Booleovu funkciju
 - više tipova, najuobičajeniji koristi preglednu tablicu (LUT) s memorijskim ćelijama:
 - ostvarivanje funkcije primjerenim MUX
~ izvedba *multipleksorskim stablom*
 - MUX u funkciji permanentne memorije!
 - tipična konfiguracija:
 - LUT s 4/5/6 ulaza → 16/32/64 ćelija
 - dodatno sklopovlje
~ D bistabil, FA, ali i složenije (množitelji, DSP i sl.)

Generiranje Booleovih funkcija

- model logičkog bloka s 2 ulaza
 - ulazi \sim *adresni* ulazi MUX;
npr. 2 ulaza \rightarrow 4-ulazni MUX: $f = f(x_1, x_2)$
 - α_i : sadržaj memorijskih ćelija

$$\begin{aligned} f &= \overline{x_1} \cdot (\overline{x_2} \cdot \alpha_0 + x_2 \cdot \alpha_1) + x_1 \cdot (\overline{x_2} \cdot \alpha_2 + x_2 \cdot \alpha_3) \\ &= \overline{x_1} \cdot \overline{x_2} \cdot \alpha_0 + \overline{x_1} \cdot x_2 \cdot \alpha_1 + x_1 \cdot \overline{x_2} \cdot \alpha_2 + x_1 \cdot x_2 \cdot \alpha_3 \end{aligned}$$



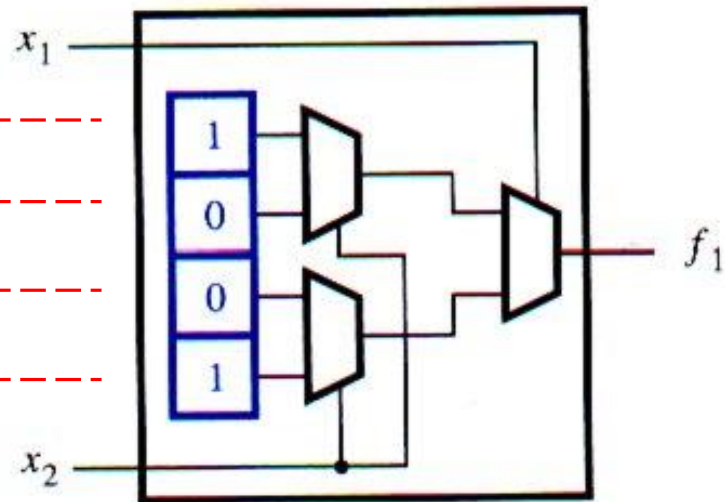
Generiranje Booleovih funkcija

Primjer: logički blok s 2 ulaza

sadržaj ćelija $\underline{y} = \langle 1, 0, 0, 1 \rangle$

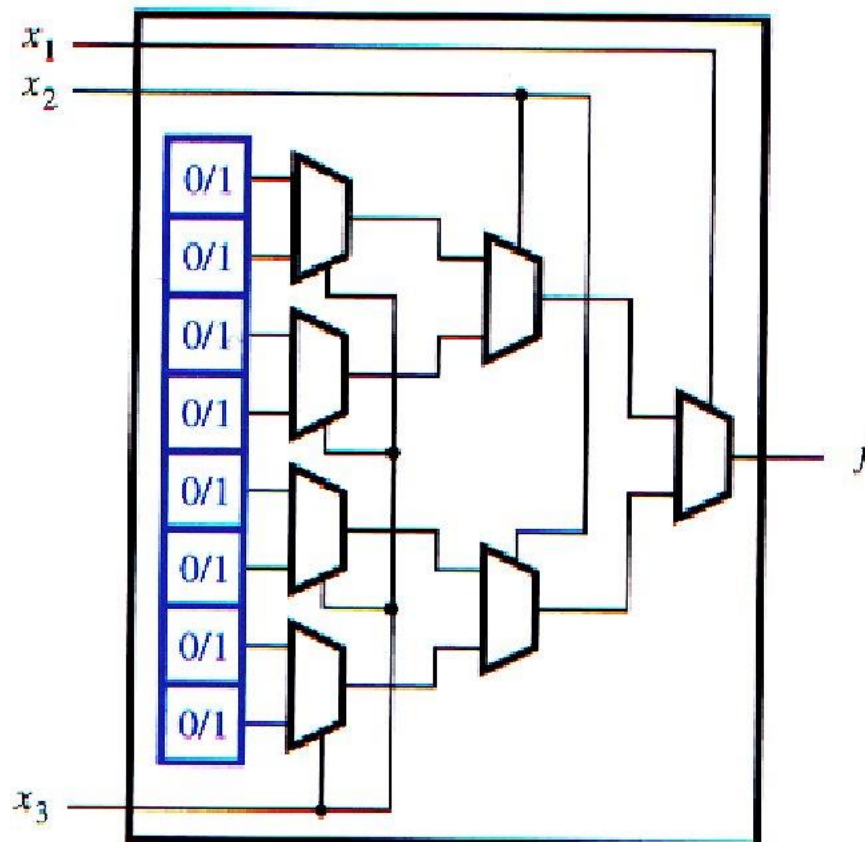
x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	1

$$f_1 = \overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2$$



Generiranje Booleovih funkcija

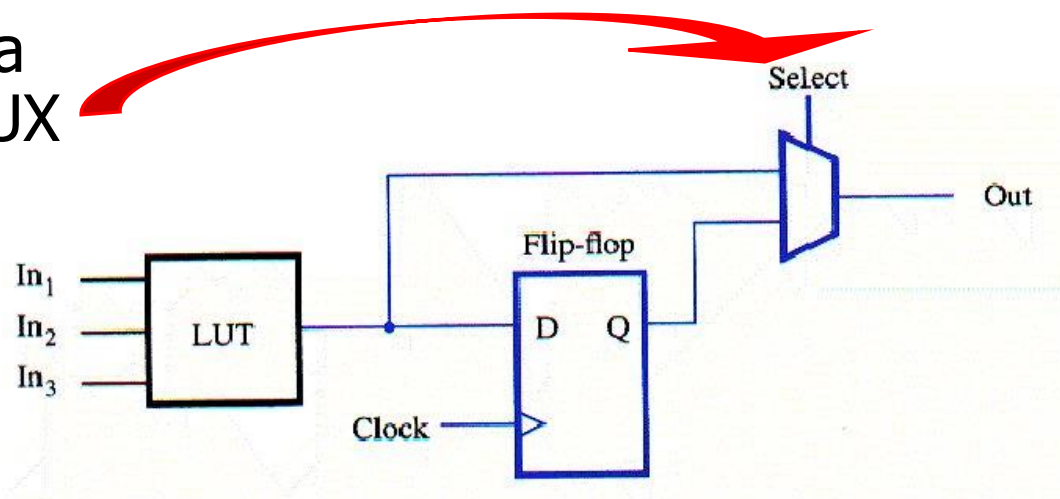
- model logičkog bloka s 3 ulaza
~ 8 memorijskih ćelija



Generiranje Booleovih funkcija

- "dodatna logika" u logičkom bloku
~ makroćelija:

- element za pamćenje
(D bistabil: memorira 1 bit)
- odabir izlaza
~ izlazni MUX



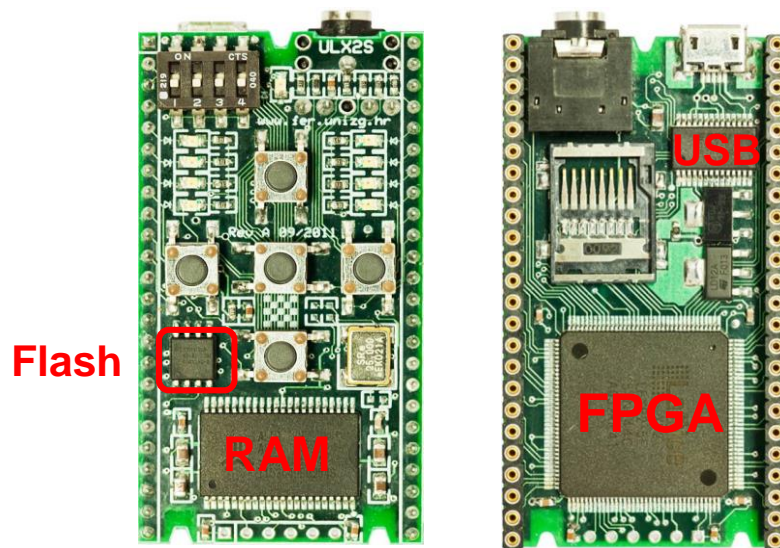
$$Out = \overline{Select} \cdot f + Select \cdot f^{n-1}$$
$$f = f(In_1, In_2, In_3)$$

Generiranje Booleovih funkcija

- *programiranje* FPGA
~ također ISP:
 - memorijske ćelije LUT su *nestalne* (engl. volatile)
~ (EA)ROM za pohranjivanje sadržaja LUT
 - automatsko *punjenje* (engl. loading) prilikom uključivanja uređaja

Primjer: Lattice XP2

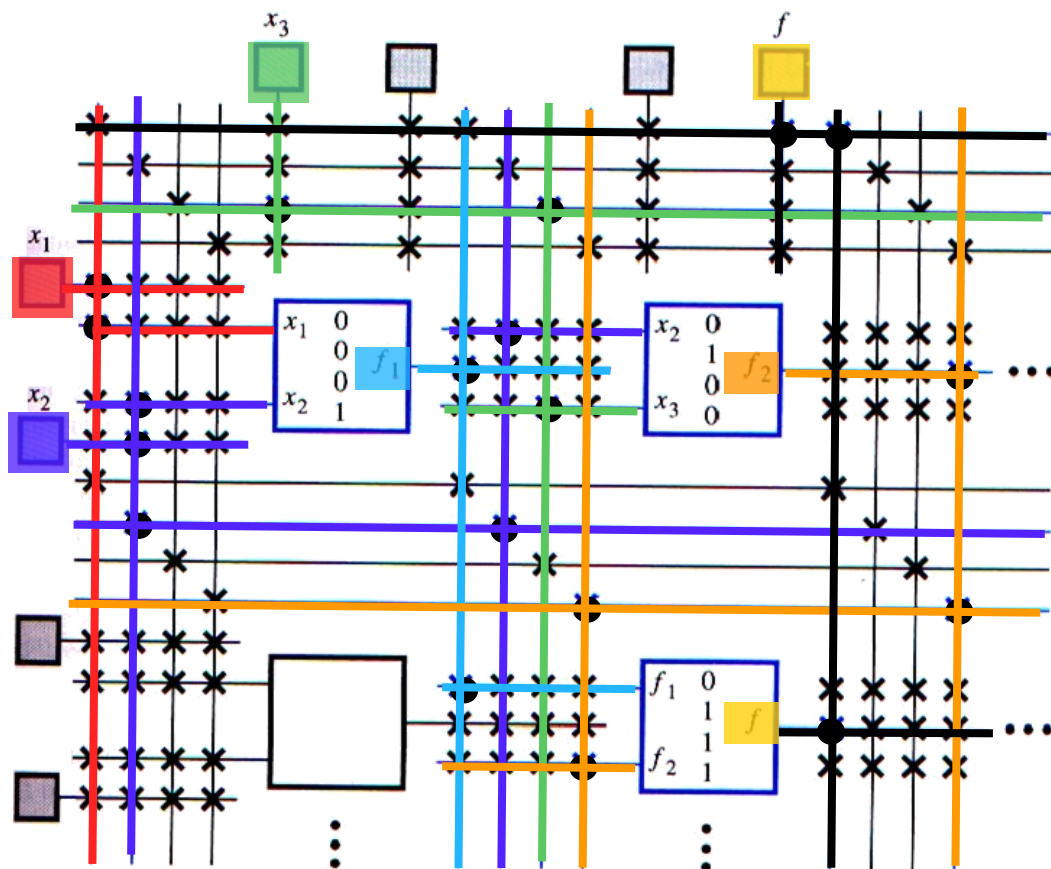
- 5K÷40K LUTova
- LB: 2 × LUT s 4 ulaza
(~ 5 ulaza)
- 2 × D-bistabil
- 86 ÷ 540 I/O ćelija
- sklopovi za DSP



Generiranje Booleovih funkcija

Primjer: (dio) programiranog FPGA

$$f(x_1, x_2, x_3) = ?$$



$$f_1 = x_1 \cdot x_2$$

$$f_2 = \overline{x_2} \cdot x_3$$

$$f = f_1 + f_2$$

$$= x_1 \cdot x_2 + \overline{x_2} \cdot x_3$$

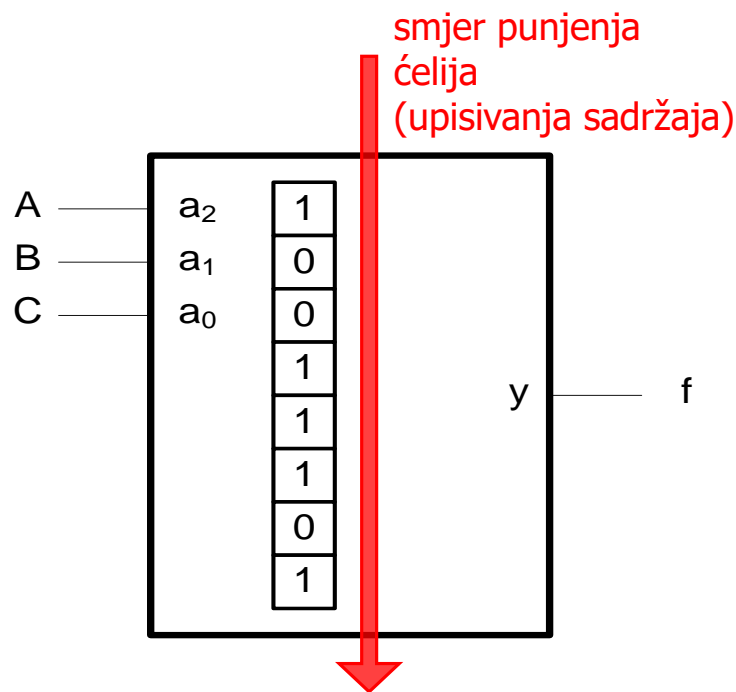
Generiranje Booleovih funkcija

Primjer: $f(A, B, C) = B \cdot C + \overline{B} \cdot \overline{C} + A \cdot C$

- ostvarenje LUTom s 3 ulaza
~ standardno "programiranje" multipleksora
- u LUT se upisuju funkcijske vrijednosti!
(*nema* minimizacije!)

f(A,B,C)		AB			
		00	01	11	10
C	0	1			1
	1		1	1	1

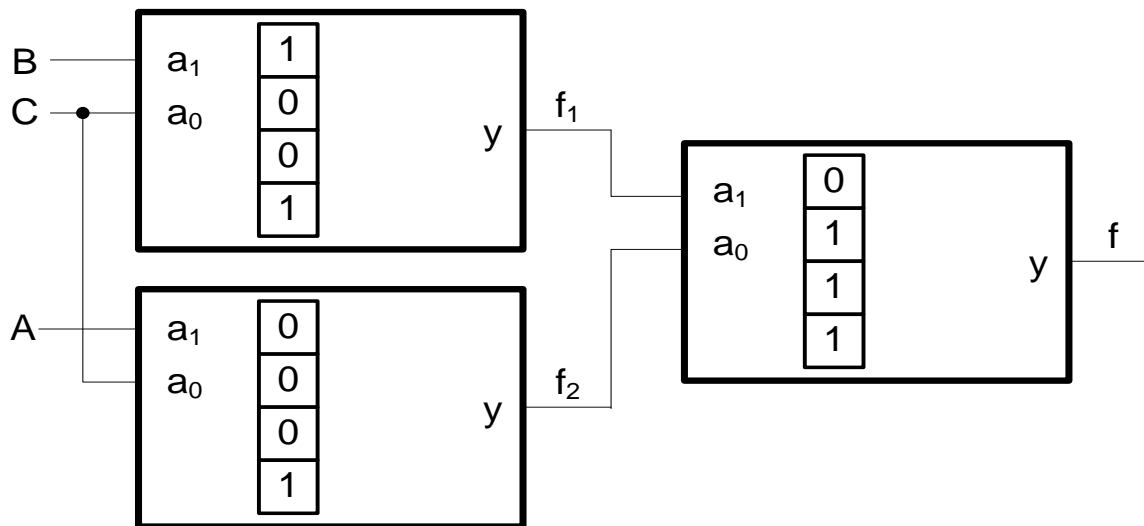
$$f(A, B, C) = \sum (0, 3, 4, 5, 7)$$



Generiranje Booleovih funkcija

- ostvarenje LUTovima s 2 ulaza
~ *kaskadiranje* LUTova

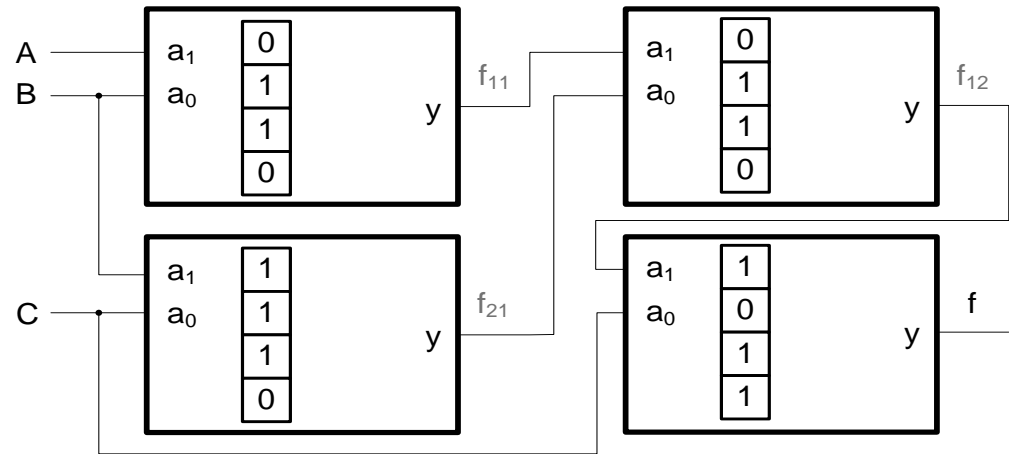
$$\begin{aligned} f(A, B, C) &= B \cdot C + \overline{B} \cdot \overline{C} + A \cdot C = (B \cdot C + \overline{B} \cdot \overline{C}) + A \cdot C \\ &= (B \otimes C) + A \cdot C = \overline{B \oplus C} + A \cdot C \\ &= f_1 + f_2 \end{aligned}$$



Generiranje Booleovih funkcija

Primjer: $f = ?$

$$\begin{aligned}
 f_{11}(A, B) &= A \oplus B \\
 f_{21}(B, C) &= \overline{BC} \\
 f_{12}(f_{11}, f_{21}) &= f_{11} \oplus f_{21} \\
 &= A \oplus B \oplus \overline{BC} \\
 f(f_{12}, C) &= \overline{f_{12}} \cdot C \\
 &= f_{12} + \overline{C} \\
 &= A \oplus B \oplus \overline{BC} + \overline{C} \\
 &= A \oplus B \oplus (\overline{B} + \overline{C}) + \overline{C} \\
 &= \dots \\
 &= \overline{A} + \overline{C} \\
 f &= \overline{A} + \overline{C}
 \end{aligned}$$



Generiranje Booleovih funkcija

$$\begin{aligned}
 f &= A \oplus B \oplus \overline{B \cdot C} + \overline{C} \\
 &= A \oplus (B \oplus \overline{B \cdot C}) + \overline{C} \\
 &= A \oplus (\overline{B \cdot B \cdot C} + \overline{B \cdot \overline{B} \cdot \overline{C}}) + \overline{C} \\
 &= A \oplus (\overline{B \cdot (\overline{B} + \overline{C})} + \overline{B \cdot (B \cdot C)}) + \overline{C} \\
 &= A \oplus (\overline{B + B \cdot C} + B \cdot C) + \overline{C} \\
 &= A \oplus (\overline{B + B \cdot C}) + \overline{C} \\
 &= A \oplus (\overline{B + C}) + \overline{C} \\
 &= \overline{A} \cdot (\overline{B + C}) + A \cdot \overline{\overline{B + C}} + \overline{C} \\
 &= \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{C} + A \cdot B \cdot \overline{C} + \overline{C} \\
 &= \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{C} + \overline{C} \\
 &= \overline{A} \cdot \overline{B} + \overline{A} + \overline{C} \\
 &= \overline{A} + \overline{C}
 \end{aligned}$$

$f_{11}(A,B,C)$

	00	01	11	AB 10
C 0		1		1
1		1		1

$f_{21}(A,B,C)$ \oplus

	00	01	11	AB 10
C 0	1	1	1	1
1	1			1

$f_{12}(A,B,C)$ =

	00	01	11	AB 10
C 0	1		1	
1	1	1		

$f_{22}(A,B,C)$ +

	00	01	11	AB 10
C 0	1	1	1	1
1				

$f(A,B,C)$ =

	00	01	11	AB 10
C 0	1	1	1	1
1	1	1		

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

- koncept programirljivih modula: str. 276
- permanentna memorija: str. 267-275
- programirljivo logičko polje: str. 276-281
- poluprogramirljivo logičko polje: str. 281-282
- složeni programirljivi moduli: str. 283-284
- programirljivo polje logičkih blokova: str. 285-286



Zadaci za vježbu (1)

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

- permanentna memorija: 7.27-7.29
- programirljivo logičko polje: 7.2, 7.30-7.32, 7.34
- programirljivo polje logičkih blokova: 7.33

Zadaci za vježbu (2)

M. Čupić: *Digitalna elektronika i digitalna logika. Zbirka riješenih zadataka*, Cjelina 5: Standardni kombinacijski moduli. Cjelina 6: Standardni programirljivi moduli.

- permanentna memorija:
 - riješeni zadaci: 5.12b-5.15
- programirljivo logičko polje
 - riješeni zadaci: 6.1-6.4, 6.7, 6.13
 - zadaci za vježbu: 1-3
- poluprogramirljivo logičko polje
 - riješeni zadaci: 6.5-6.6
- programirljivo polje logičkih blokova
 - riješeni zadaci: 6.8-6.11