# 3 Kombinacijska vezja

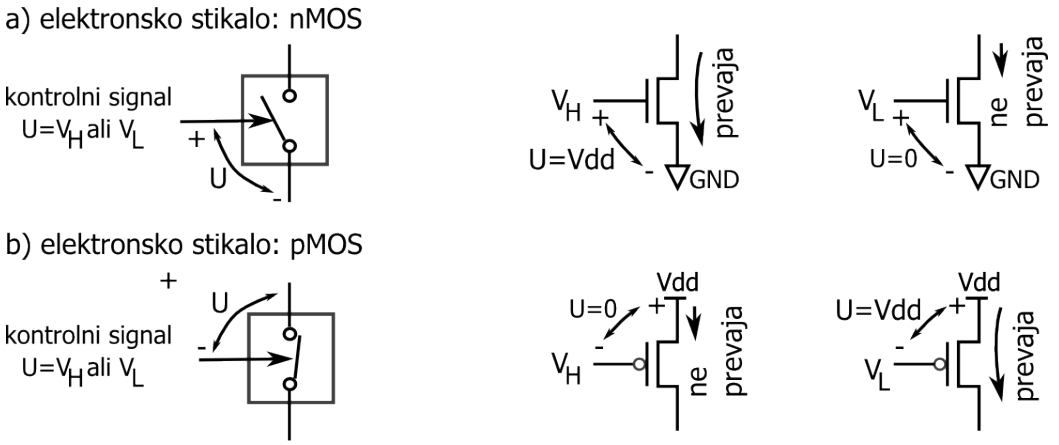
V poglavju o signalih smo spoznali preprost gradnik digitalnih vezij z imenom logični negator. Stanje na izhodu logičnega negatorja je invertirana (zamenjana) vrednost stanja na vhodu. Negator spada v veliko skupino logičnih vezij, pri katerih je stanje na izhodu odvisno le od trenutne kombinacije stanj na vhodih. Takšna digitalna vezja se imenujejo *kombinacijska* vezja in predstavljajo osnovo na kateri temeljijo vsi digitalni sistemi.

## 3.1 Elektronska stikala

Digitalna vezja temeljijo na elektronskih stikalih. Prva digitalna vezja so bila narejena iz elektromehanskih stikal - relejev, ki so jih kasneje zamenjale elektronske cevi (elektronke). To so bili veliki in okorni gradniki, ki so bili pogosto v okvari. Leta 1945 so med iskanjem napake v računalniku Mark II našli hrošča med kontakti releja. Od takrat računalniške napake imenujejo hrošč (angl. bug) in postopek odkrivanja ter odpravljanja napak "razhroščevanje"(debugging).

Precej manjša in bolj zanesljiva elektronska stikala naredimo s polprevodniškimi *transistorji*. Transistor je narejen iz koščka silicija s primesmi, ki mu lahko prevodnost spreminjamo z električnim potencialom na kontrolnem vhodu. Največji skok v razvoju digitalnih sistemov pa so omogočila *integrirana vezja*, v katerih je na majhni rezini silicija narejeno celotno vezje.

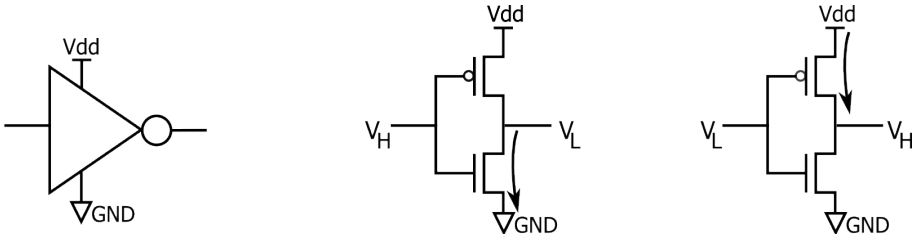
Slika 3.1 prikazuje izvedbo elektronskega stikala s transistorji dveh vrst: nMOS in pMOS. Elektronsko stikalo sklene kontakt, kadar je napetost U med kontrolno sponko in enim izmed kontaktov večja od določenega praga, sicer pa je kontakt razklenjen. Če ima stikalo s transistorjem nMOS spodnji kontakt vezan na maso, ga odpremo z visokim stanjem $V_H$ na kontrolnem vhodu. Kadar je na kontrolnem vhodu nizko logično stanje $V_L$ , pa transistor ne prevaja. Pri stikalu s transistorjem pMOS gledamo napetost med zgornjo sponko in kontrolnim vhodom. Če je zgornja sponka vezana na napajalno napetost Vdd, je stikalo sklenjeno, kadar je na kontrolnem




Slika 3.1: Komplementarni izvedbi elektronskega stikala: a) nMOS in b) pMOS transistor.

vhodu nizko stanje $V_L$. Če je na kontrolnem vhodu visoko stanje $V_H$, pa je napetostna razlika manjša od praga in stikalo ne prevaja.

Digitalna vezja z oznako CMOS so zgrajena iz komplementarnih nMOS in pMOS transistorjev. Slika 3.2 prikazuje zgradbo logičnega negatorja v CMOS izvedbi. Kadar je na vhodu negatorja visok potencial, bo prevajal spodnji (nMOS) transistor in povezal izhod na maso oz. potencial $V_L$. Če je na vhodu $V_L$ pa prevaja pMOS in takrat je na izhodu napajalna napetost oz. potencial $V_H$.

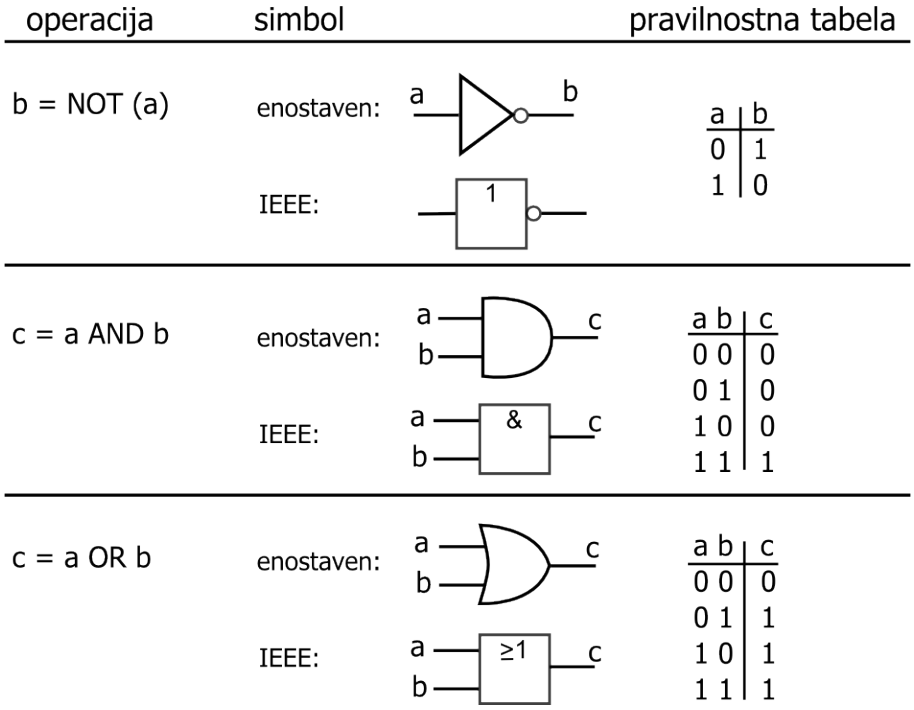



Slika 3.2: Logični negator v CMOS izvedbi.

Kontrolni vhodi CMOS transistorjev so kondenzatorji, skozi katere enosmerni tok ne teče. Zgornji in spodnji transistor sta odprta izmenično, zato pri konstantnem vhodu tudi skozi napajalni sponki CMOS gradnika praktično tok ne teče. Digitalni gradniki v izvedbi CMOS imajo izredno majhno porabo in so idealni za izdelavo kompleksnih vezij. Tehnologija integriranih vezij omogoča izdelavo vedno manjših transistorjev oz. vedno večjega števila transistorjev na enaki površini. Digitalna integrirana vezja vsebujejo danes na milijone polprevodniških transistorjev, ki delujejo kot stikala.

## 3.2 Boolova logična vrata

Transistorji so sicer sestavni deli digitalnih vezij, toda za načrtovanje digitalnih vezij uporabljamo raje nekatere sestavljene gradnike s katerimi lažje in hitreje zgradimo kompleksna vezja. Osnovni gradniki digitalnih vezij so *logična vrata*.




Slika 3.3: Osnovne Boolove operacije, simboli logičnih vrat in pravilnostne tabele.

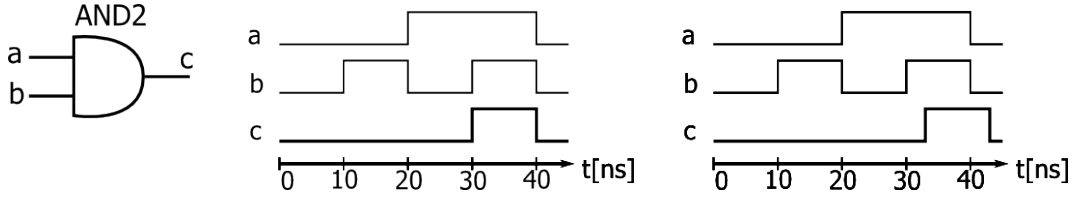
Logična vrata izhajajo iz matematične predstavitve osnovnih operacij nad binarnimi signali, ki jih obravnava Boolova algebra. Gradniki digitalnih vezij izvajajo logične (Boolove) funkcije nad binarnimi signali. Spoznali bomo tri osnovne Boolove operacije: logični IN (angl. AND), logični ALI (angl. OR) in negacijo (angl. NOT). Vrednosti osnovnih izrazov:

- **a AND b** ima vrednost 1, kadar sta oba operanda 1, sicer ima vrednost 0.
- **a OR b** ima vrednost 1, kadar je vsaj en operand 1, torej pri a=1, b=0 ali a=0, b=1 ali a=1, b=1. Vrednost 0 pa ima le ko sta oba operanda 0.
- **NOT(a)** ima vrednost 1, kadar je a=0, pri a=1 pa ima vrednost 0.

Načrtovanje kombinacijskega digitalnega vezja začnemo z opisom problema, ki ga prevedemo v Boolove izraze. Nato narišemo shemo vezja z uporabo grafičnih simbolov logičnih funkcij. V praksi se bomo srečevali z dvema vrstama grafičnih simbolov. Slika 3.3 prikazuje enostavnejše ameriške simbole in natančno definirane standardne grafične simbole (standard IEEE

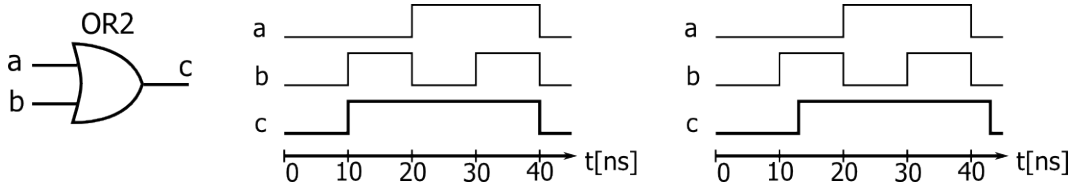
std. 91-1984) osnovnih logičnih vrat NOT, AND in OR. V skripti bomo uporabljali enostavnejše simbole, ki jih bomo srečevali tudi v programski opremi.

Delovanje logičnih gradnikov predstavimo s *pravilnostno tabelo*, v kateri navedemo vrednosti izhodov pri vseh možnih kombinacijah vrednosti vhodov. Logični negator z enim vhodom ima dve možni kombinaciji, logična vrata AND oz. OR z dvema vhodoma pa 4 kombinacije.




Slika 3.4: Idealni in realni časovni diagram logičnih vrat AND2.

Drug način predstavitve je časovni diagram, kot ga prikazujeta sliki 3.4 in 3.5. Na časovnem diagramu vidimo, kako se spreminja izhod ob spreminjanju stanj na vhodih. V realnih logičnih vratih se izhod ne spremeni takoj za spremembo vhoda ampak z neko zakasnitvijo, ki je posledica zakasnitev preklopov stikalnih elementov (transistorjev). V primerih na slikah 3.4 in 3.5 so zakasnitve logičnih vrat 6ns. Zakasnitve posledično omejujejo hitrost spreminjanja signalov na vhodih in s tem hitrost delovanja vezja. Kadar so majhne v primerjavi s pričakovanimi časovnimi intervali spreminjanja signalov, jih lahko ignoriramo. Za analizo vezja pogosto zadostuje idealni časovni diagram, v katerem zakasnitve niso prikazane.




Slika 3.5: Idealni in realni časovni diagram logičnih vrat OR2.

Logična vrata vrste **AND** in **OR** imajo lahko več kot dva vhoda. Izhod vrat AND je 1 v primeru ko so vsi vhodi enaki 1, sicer je izhod enak 0. Izhod vrat OR pa je 1 kadar je vsaj eden izmed vhodov enak 1, 0 pa je le v primeru, ko so vsi vhodi enaki 0.

**Primer 3.1**
Avtomobilski alarm naj se sproži, kadar je nastavljen in kadar je izpolnjen vsaj eden izmed pogojev: ali je aktiviran senzor gibanja ali pa so odprta vrata. Signal $alarm$ naj predstavlja stanje alarma, signal $vklop$ določa ali je alarm nastavljen, signal $vrata$ naj predstavlja stanje vrat (stanje 1 pomeni odprta) in signal $gib$ stanje senzorja. Delovanje alarma opišemo z logično enačbo:

$$alarm = vklop \text{ AND } (vrata \text{ OR } gib)$$

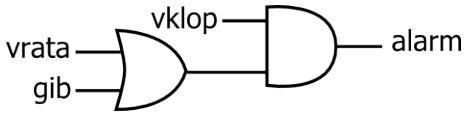
Sedaj lahko računamo vrednosti signala $alarm$ pri različnih vhodnih vrednostih:

- pri $vklop = 1, vrata = 1, gib = 0$ dobimo $alarm = 1$ AND $(1$ OR $0) = 1$ AND $1 = 1$
- pri $vklop = 0, vrata = 1, gib = 0$ dobimo $alarm = 0$ AND $(1$ OR $0) = 0$ AND $1 = 0$

Na podlagi logične enačbe lahko dokažemo nekatere trditve. Npr. trditev, da se alarm ne bo sprožil, če je vklop na 0. Logični izraz **AND** bo imel avtomatsko vrednost 0, če je na eni strani vrednost 0:

$$alarm = 0 \text{ AND } (vrata \text{ OR } gib) = 0$$

Logično enačbo pretvorimo v shemo vezja v par korakih. Najprej narišemo signal $alarm$, ki je izhod iz logičnih vrat **AND**. Vhod teh vrat je enkrat signal $vklop$, drugič pa izhod vrat **OR**, ki imajo na vhodu signala $vrata$ in $gib$, kot prikazuje slika 3.6.

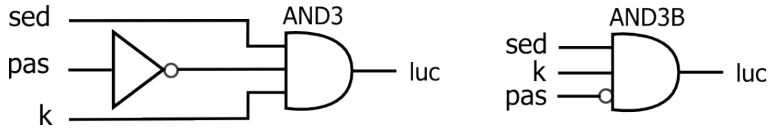



Slika 3.6: Shema vezja za avtomobilski alarm.

**Primer 3.2**
Naredimo vezje, ki prižge opozorilni indikator, če nismo pripeti z varnostnim pasom. Signal s senzorja varnostnega pasu ($pas$) je v stanju 1, ko je pas pripet in v stanju 0, kadar je odpet. Senzor v sedežu postavi signal $sed$ v stanje 1, kadar nekdo sedi v avtu. Poleg tega imamo informacijo ali je ključ ($k$) v položaju, ko je avto v pogonu (stanje 1). Indikatorska luč naj se prižge, če voznik sedi v avtu IN je pas odpet IN je avto v pogonu, kar zapišemo z enačbo:

$$luc = sed \text{ AND NOT}(pas) \text{ AND } k$$

Logično vezje naredimo z enimi tri-vhodnimi logičnimi vrati AND in negatorjem, ali pa s posebnim simbolom vrat AND, kjer negiran vhod označimo s krogcem, kot prikazuje slika 3.7.




Slika 3.7: Dve obliki sheme vezja za opozorilno luč varnostnega pasu.

Opozorilni indikatorji v avtomobilu se prižgejo za kratek čas tudi kadar obrnemo ključ. S tem preverimo ali vse opozorilne lučke res delujejo. Vzemimo, da imamo na voljo testni signal $t$, ki se ko obrnemo ključ za nekaj sekund postavi na 1, potem pa gre nazaj na 0. Indikatorska luč naj se prižge kadar je testni signal na 1 ALI pa v primeru, če voznik ni pripet:

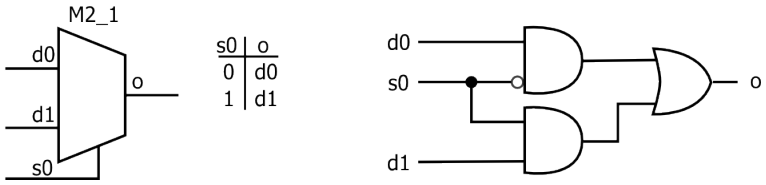
$$luc = t \text{ OR } sed \text{ AND NOT}(pas) \text{ AND } k$$

## 3.3 Kombinacijski bloki

Kompleksna kombinacijska vezja sestavljamo s povezavo vnaprej pripravljenih logičnih blokov, ki izvajajo različne funkcije. Nekateri bloki se pri razvoju vezij pogosto pojavljajo in so na voljo v obliki knjižničnih komponent. V tem poglavju bomo spoznali tri standardne komponente kombinacijskih vezij: izbiralnik, dekodirnik in seštevalnik.

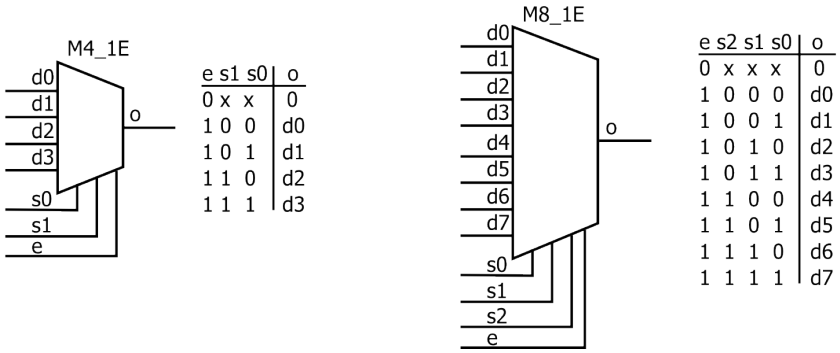
### 3.3.1 Izbiralnik

Izbiralnik (angl. multiplexer) je vezje, ki prenese na izhod stanje na enemu izmed izbranih podatkovnih vhodov. Vrsto izbiralnika določa število podatkovnih vhodnih signalov. Število podatkovnih vhodov je običajno potenca $2^n$ in izbiralniki so označeni kot 2-1, 4-1, 8-1 ipd. Oznake pomenijo, da za stanje izhoda izberemo enega izmed dveh, štirih ali osmih vhodov. Izbiralnik z $2^n$ podatkovnimi vhodi potrebuje še n-bitni izbirni vhod.




Slika 3.8: Simbol izbiralnika 2-1, pravilnostna tabela in izvedba z logičnimi vrati.

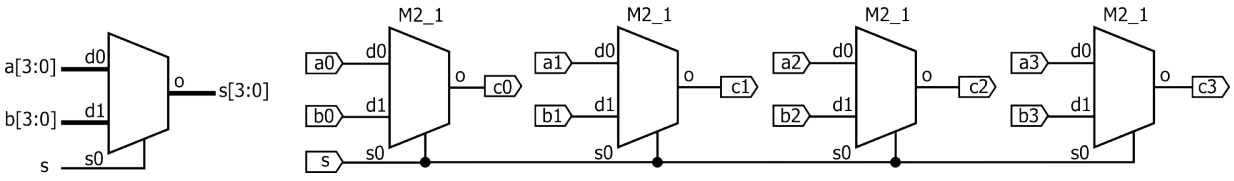
Slika 3.8 prikazuje najbolj enostaven izbiralnik 2-1, ki ima podatkovna vhoda $d0$ in $d1$ ter izbirni vhod $s0$. Ko je izbirni vhod enak 0, dobimo na izhodu vrednost iz vhoda $d0$, sicer pa vrednost iz vhoda $d1$. Logično vezje tega izbiralnika je sestavljeno iz treh osnovnih logičnih vrat. Izbiralnik ima včasih tudi dodaten vhodni signal $e$ (angl. enable) za omogočanje izhoda. Kadar je signal $e$ enak 1 deluje izbiralnik normalno, kadar pa je $e$ enak 0, je izhod na konstantni vrednosti 0, ne glede na stanje izbranega vhoda. Slika 3.9 prikazuje izbiralnika 4-1 in 8-1 z vhodom za omogočanje in njuni pravilnostni tabeli.




Slika 3.9: Izbiralnika 4-1 in 8-1 z dodatnim vhodom za omogočanje $e$.

Logične izvedbe vseh izbiralnikov so lahko narejene po podobnem principu: podatkovni

vhodi gredo skupaj z izbirnimi signali na vrata AND, nato pa se združijo skupaj v vratih OR. Izbiralniki s signalom $e$ imajo dodaten vhod na vseh vratih AND, kamor je ta signal vezan.

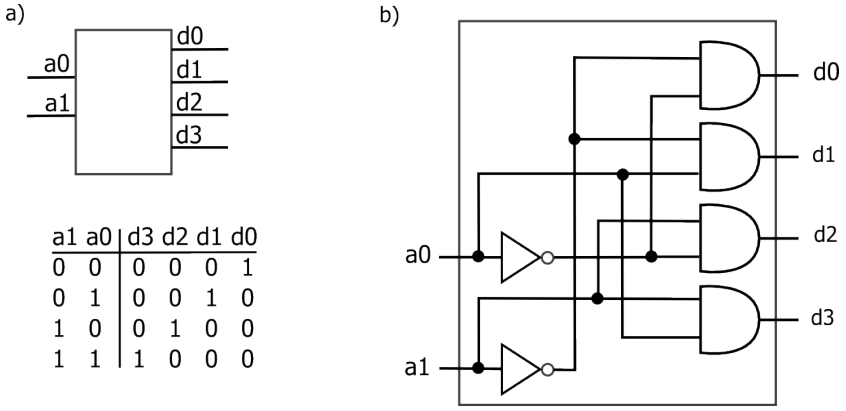



Slika 3.10: Simbol in vezje izbiralnika 4-bitnih vrednosti.

Kadar potrebujemo izbiralnike večbitnih vrednosti, vzamemo več osnovnih izbiralnikov in njihove izbirne vhode vežemo skupaj, podatkovne vhode in izhode pa na posamezne bite. Izbiralnik, ki na 4-bitni izhod $c$ prenese enega izmed 4-bitnih vhodnih signalov $a$ ali $b$ je prikazan na sliki 3.10.

### 3.3.2 Dekodirnik

*Binarni dekodirnik* je logični gradnik, ki glede na vrednost na n-bitnem vhodu postavi enega izmed izhodov na 1, ostali izhodi pa so 0. Dekodirnik ima ima $n$ vhodov in $2^n$ izhodov: 2-bitni dekodirnik ima 2 vhoda in 4 izhode, 3-bitni ima 3 vhode in 8 izhodov, 4-bitni pa 4 vhode in 16 izhodov. Slika 3.11 prikazuje grafični simbol dvobitnega binarnega dekodirnika, pravilnostno tabelo in izvedbo z logičnimi vrati. Vhod dekodirnika $a1$ in $a0$ obravnavamo kot 2-bitno število. Imena izhodnih signalov: $d0$, $d1$, $d2$ in $d3$ označujejo desetiške vrednosti binarne kombinacije na vhodu. Vrednost 1 ima le tisti izhod, katerega oznaka je enaka trenutni vrednosti vhodne kombinacije. Kadar je na vhodu 0 0, je izhod $d0$ postavljen na 1, pri 0 1 je postavljen izhod $d1$ itn.



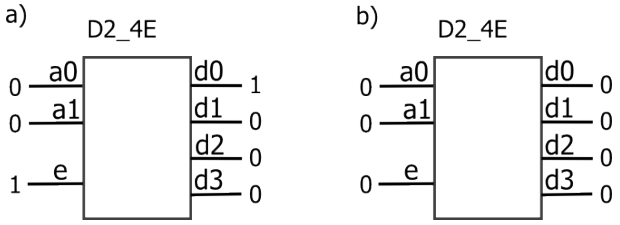

| a1 | a0 | d3 | d2 | d1 | d0 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

Slika 3.11: 2-bitni dekodirnik: a) simbol dekodirnika s pravilnostno tabelo in b) izvedba.

Za posamezne izhodne signale lahko zapišemo logične enačbe, npr. izhod $d0$ je enak 1, kadar sta oba vhoda $a0$ in $a1$ enaka 0: vhoda torej negiramo in uporabimo operacijo AND. Podobno razmišljamo za ostale izhode in zapišemo enačbe:

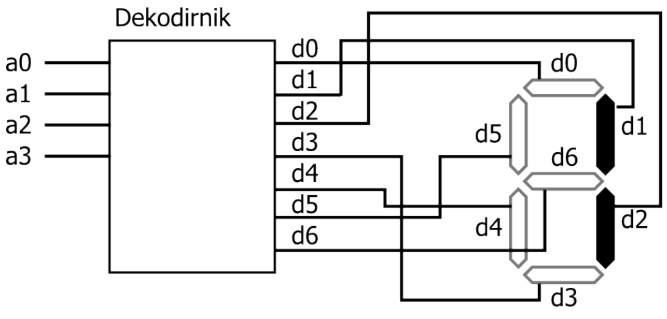
$$d0 = \text{NOT}(a0)\ \text{AND}\ \text{NOT}(a1)$$
$$d1 = a0\ \text{AND}\ \text{NOT}(a1)$$
$$d2 = \text{NOT}(a0)\ \text{AND}\ a1$$
$$d3 = a0\ \text{AND}\ a1$$

Binarni dekodirnik je narejen iz štirih logičnih vrat AND in negatorjev. V logičnih izrazih opazimo, da se negirana signala $a0$ in $a1$ pojavljata večkrat, kar izkoristimo za optimizacijo vezja. Namesto da bi uporabili štiri negatorje, uporabimo le dva in povežemo negiran signal na več vrat AND, kot prikazuje slika 3.11 b). Slika 3.12 prikazuje delovanje dekodirnika s signalom za omogočanje, ki ima podobno funkcijo kot pri izbiralniku. Kadar je signal $e$ enak 1 deluje dekodirnik normalno, kadar pa je $e$ enak 0, so vsi izhodi dekodirnika enaki 0.




Slika 3.12: Binarni dekodirnik z omogočanjem a) pri e=1 deluje normalno, b) pri e=0 pa so vsi izhodi 0.

Poleg binarnih dekodirnikov poznamo še vrsto drugih dekodirnih vezij, ki imajo vsi skupno lastnost, da kombinacijo stanj na vhodih pretvorijo v neko drugo kombinacijo na izhodih. Dekodirniki spadajo po definiciji med kombinacijska vezja. Med bolj znanimi je dekodirnik za 7-segmentne prikazovalnike, ki pretvarja 4-bitno število v kombinacijo za prikaz desetiške števke (slika 3.13). Prikazovalnik je sestavljen iz sedmih svetlečih diod, ki so razporejene tako, da lahko prikažejo desetiške števke.

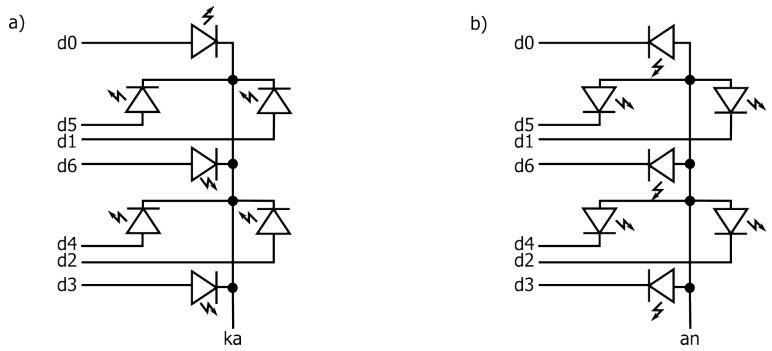



| a3..a0 | d6..d0 |
|---|---|
| 0000 | 0111111 |
| 0001 | 0000110 |
| 0010 | 1011011 |
| 0011 | 1001111 |
| 0100 | 1100110 |
| 0101 | 1101101 |
| 0110 | 1111101 |
| 0111 | 0000111 |
| 1000 | 1111111 |
| 1001 | 1101111 |

Slika 3.13: Vezava dekodirnika za 7-segmentni prikazovalnik in pravilnostna tabela.

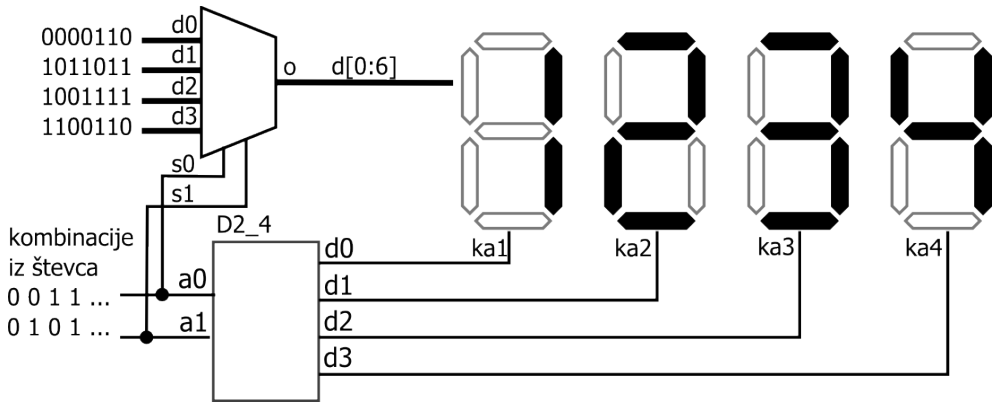
Poznamo dve vrsti LED prikazovalnikov: s skupno katodo in s skupno anodo, kot prikazuje slika 3.14. Segmente prikazovalnikov s skupno katodo prižigamo z logično 1, katodo pa vežemo preko upora na maso. Prikazovalnik s skupno anodo pa ima skupni priključek vezan na Vdd in

posamezne segmente prižigamo z logično 0. V tem primeru moramo izhode dekodirnika negirati. Nekateri prikazovalniki imajo še dodatno diodo (signal $d7$) za prikaz decimalne pike.




Slika 3.14: LED prikazovalniki: a) s skupno katodo in b) s skupno anodo.

Kadar imamo v vezju več 7-segmentnih prikazovalnikov, so običajno podatkovni signali ($d0$ do $d6$) vezani skupaj, s signali na skupnih anodah ali katodah pa izmenično prižigamo posamezne prikazovalnike. Če imamo štiri prikazovalnike, tako namesto $4 \times 7 = 28$ signalov potrebujemo le 7 podatkovnih in 4 izbirne signale. Z logičnim vezjem poskrbimo, da je v vsakem trenutku prižgan le en prikazovalnik. Če izbrani prikazovalnik dovolj hitro menjavamo, npr. vsaj $20\times$ v sekundi, naše oko ne bo opazilo zamenjav in bomo videli kot da so prižgani vsi prikazovalniki skupaj. Izbrani prikazovalnik določimo s števcem, ki poskrbi za zaporedne binarne kombinacije in binarnim dekodirnikom, kot prikazuje shema na sliki 3.15. V vezju je uporabljen še izbiralnik 4-1, ki poskrbi, da se v ritmu menjave izbranih prikazovalnikov spreminjajo tudi podatkovni signali.




Slika 3.15: Skupna vezava štirih LED prikazovalnikov s skupno katodo.

Dekodirnik lahko vedno naredimo z uporabo tolikšnega števila izbiralnikov, kolikor imamo izhodov. Velikost uporabljenih izbiralnikov pa določa število signalov na vhodu dekodirnika. Vzemimo naprimer vezje, ki dekodira smer vetra. Na vhodu naj bodo 4 signali: $s$, $j$, $v$ in $z$, od katerih je vedno postavljen na 1 le en signal, ostali pa so 0. Naloga dekodirnika je, da na podlagi vrednosti vhodov določi dvobitni signal smer po pravilu:

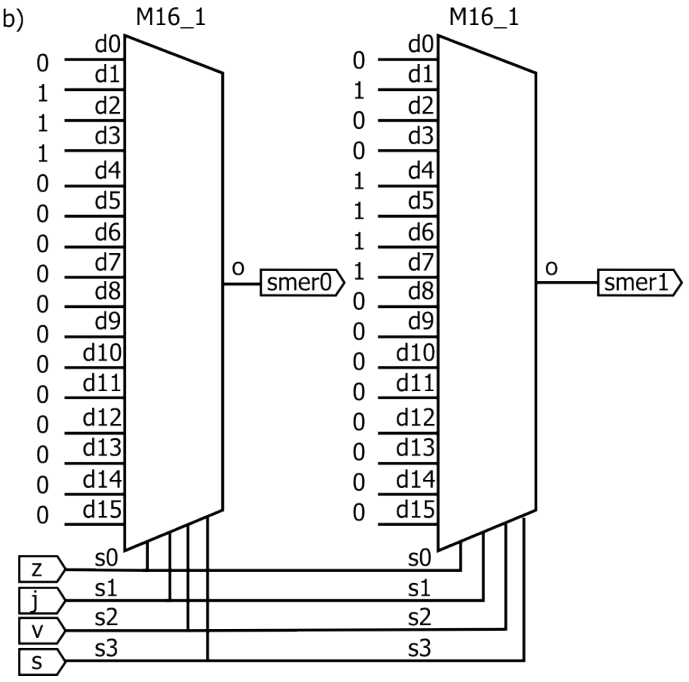
- $smer =$ "00" kadar je $s = 1$,
- $smer =$ "01" kadar je $v = 1$,
- $smer =$ "10" kadar je $j = 1$,
- $smer =$ "11" kadar je $z = 1$.

Iz opisa naredimo pravilnostno tabelo z vsemi kombinacijami štirih vhodov, pri kateri določimo kakšen naj bo izhod tudi v primerih, ko je več vhodov postavljenih na 1. Na sliki 3.16 a) je pravilnostna tabela dekodirnika smeri vetra v kateri so z zvezdico označene pričakovane kombinacije vhodov.

a)

| s | v | j | z | smer[1:0] | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 0 | |
| 0 | 0 | 0 | 1 | 1 1 | * |
| 0 | 0 | 1 | 0 | 1 0 | * |
| 0 | 0 | 1 | 1 | 1 0 | |
| 0 | 1 | 0 | 0 | 0 1 | * |
| 0 | 1 | 0 | 1 | 0 1 | |
| 0 | 1 | 1 | 0 | 0 1 | |
| 0 | 1 | 1 | 1 | 0 1 | |
| 1 | 0 | 0 | 0 | 0 0 | * |
| 1 | 0 | 0 | 1 | 0 0 | |
| 1 | 0 | 1 | 0 | 0 0 | |
| 1 | 0 | 1 | 1 | 0 0 | |
| 1 | 1 | 0 | 0 | 0 0 | |
| 1 | 1 | 0 | 1 | 0 0 | |
| 1 | 1 | 1 | 0 | 0 0 | |
| 1 | 1 | 1 | 1 | 0 0 | |

b)




Slika 3.16: Dekodirnik smeri vetra: a) pravilnostna tabela in b) izvedba z izbiralniki.

Kodirnik smeri vetra ima $2^4 = 16$ kombinacij in za vsak bit izhoda uporabimo en izbiralnik 16-1, kot prikazuje slika 3.16 b). Na podatkovne vhode postavimo konstantne vrednosti, ki predstavljajo posamezen stolpec pravilnostne tabele. Izbirni vhodi $s0$ do $s3$ obeh izbiralnikov so vezani skupaj in priključeni na vhodne signale. Pri tem moramo paziti na vrstni red: izbirni signal $s3$ je vezan na signal na levi strani pravilnostne tabele, $s2$ na naslednji signal itn.

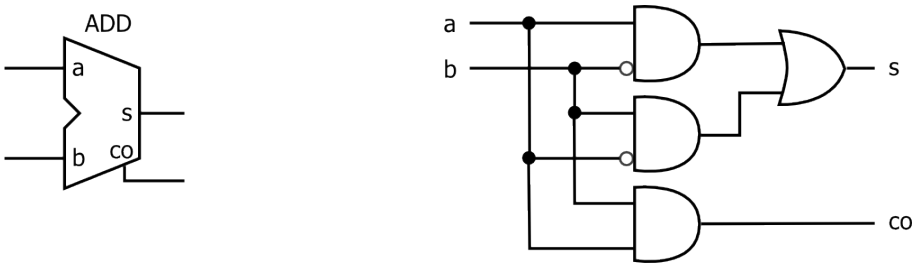
### 3.3.3 Seštevalnik

S signali, ki predstavljajo številsko kodo, lahko izvajamo aritmetične (računske) operacije. Osnovna aritmetična operacija je seštevanje dveh vrednosti. Najprej si poglejmo vsoto dveh enobitnih vrednosti pri vseh možnih kombinacijah:

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

prenos

Pri zadnji kombinaciji smo dobili prenos, ki ga upoštevamo kot dodatno števko. Logično vezje za izračun vsote se imenuje seštevalnik. Slika 3.17 prikazuje simbol in logično shemo enobitnega seštevalnika. Na vhodu sta operanda $a0$ in $b0$, na izhodu pa je vsota $s0$ (angl. sum) in prenos $c0$ (angl. carry). Logično vezje enobitnega seštevalnik je sestavljeno iz štirih osnovnih logičnih vrat.

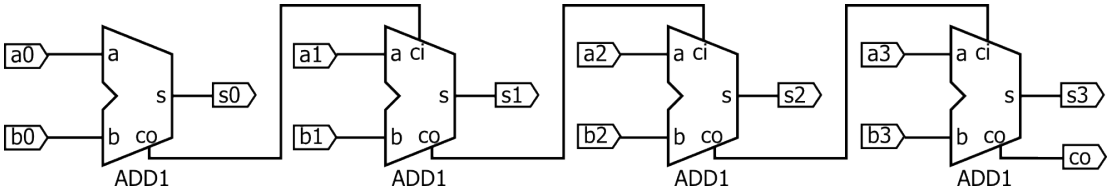



Slika 3.17: Simbol in logična shema enobitnega seštevalnika.

Postopek seštevanja večmestnih števil poznamo že iz ročnega seštevanja v desetiškem sistemu: števili poravnamo in seštevamo števke od desne proti levi. Če pride pri vsoti števk do prenosa, ga upoštevamo pri naslednji vsoti. Poglejmo si primer seštevanja dveh 4-bitnih vrednosti:

$$\begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 1\,0\,0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,1\,0\,0 \end{array}$$

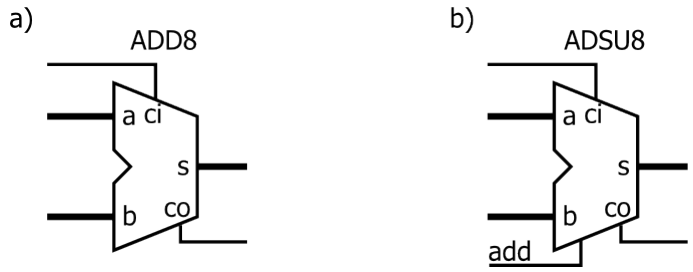
Vezje 4-bitnega seštevalnika naredimo s povezavo enobitnih seštevalnikov z vhodnim in izhodnim prenosom. Seštevalniki ADD1 na sliki 3.18 izračuna vsoto operandov $a$, $b$ in vhodnega prenosa $ci$. Vsak izmed seštevalnikov izračuna en bit vsote. Po seštevanju nižjih bitov se izhodni prenos upošteva kot vhod v naslednji seštevalnik. Vsoto sestavljajo biti od $s0$ do $s3$ in izhodni prenos $co$.




Slika 3.18: Shema 4-bitnega seštevalnika.

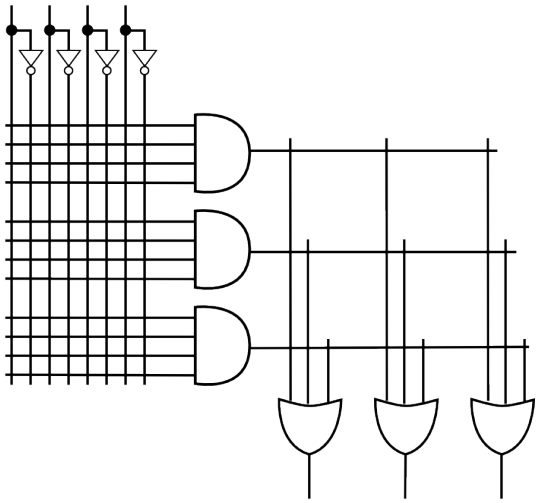
Seštevalniki so pogosto uporabljeni elementi, zato dobimo v knjižnici osnovnih gradnikov že pripravljene večbitne seštevalnike. Slika 3.19 a) prikazuje simbol seštevalnika ADD8 z 8-bitnimi vhodi $a$ in $b$ in izhodom $s$. Seštevalnik ima tudi vhod in izhod za prenos, tako da lahko z zaporedno vezavo zgradimo še večje seštevalnike.

Enak simbol se uporablja tudi za splošno aritmetično enoto, ki izvaja različne računske operacije nad večbitnimi signali. Slika 3.19 b) prikazuje simbol 8-bitne seštevalno/odštevalne enote ADSU8. Dodatni vhod $add$ določa ali enota izvaja operacijo seštevanja ($add = 1$) ali pa odštevanja ($add = 0$).




Slika 3.19: Simboli: a) 8-bitni seštevalnik in b) 8-bitni seštevalnik/odštevalnik.

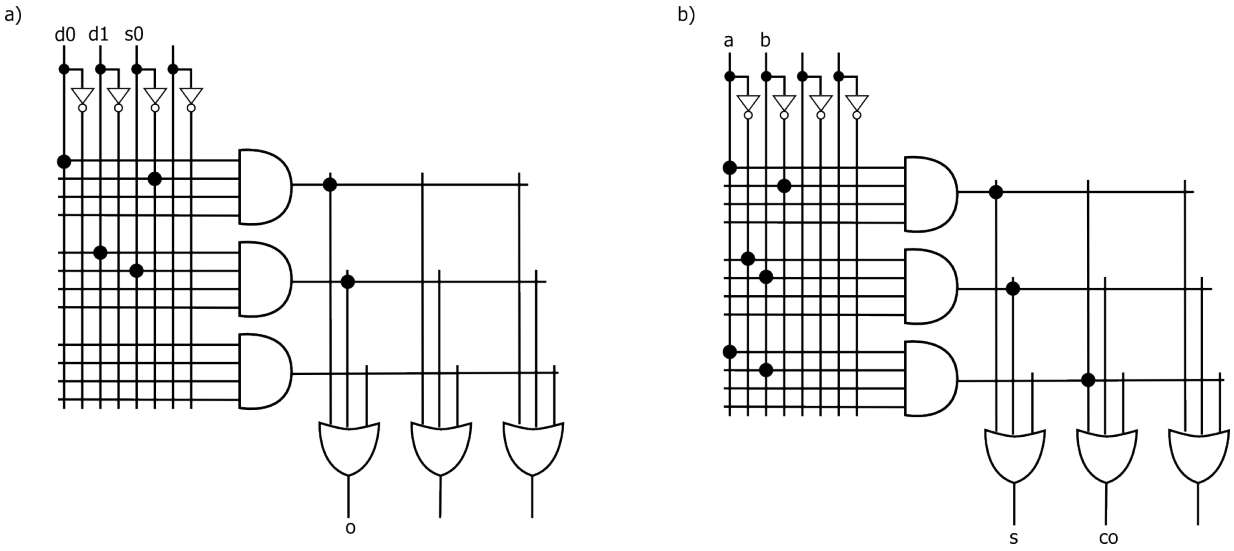
## 3.4 Programirljive matrike



Slika 3.20: Programirljiva matrika PLA.

Tehnologija programirljivih vezij z oznako PLD (angl. Programmable Logic Device) temelji na dejstvu, da lahko poljubno kombinacijsko vezje naredimo z logičnimi vrati AND, OR in NOT. Osnovni element teh vezij je programirljiva matrika (PLA), sestavljena iz vrstic z večvhodnimi vrati AND, ki imajo izhode vezane na celice z vrati OR. Vhodni signali se direktno ali v negirani obliki povežejo na vrata AND, kot prikazuje slika 3.20.

V postopku programiranja vzpostavimo povezave in s tem določimo kateri vhodi so vezani na posamezna vrata AND in kateri izhodi vrat AND so vezani na OR celice. S takšno strukturo hitro ž naredimo osnovne gradnike: slika 3.21 prikazuje izvedbo izbiralnika 2-1 in enobitnega seštevalnika z matriko PLA.




Slika 3.21: Izvedba gradnikov s PLA: a) izbiralnik 2-1 in b) enobitni seštevalnik.

Velikost PLA je v pravih programirljivih vezjih precej večja. Kompleksna programirljiva vezja CPLD iz družine Coolrunner-II proizvajalca Xilinx so sestavljena iz PLA blokov z 40 vhodnimi signali, 56 večvhodnimi vrati AND in 16 celicami z vrati OR. Kompleksne gradnike, ki jih ne moremo naredito z enim PLA blokom, razdelimo in naredimo z več bloki, ki jih povežemo znotraj programirljivega integriranega vezja. Preslikavo logične sheme v PLA bloke opravlja programska oprema, tako da za njihovo uporabo ni potrebno podrobno poznavanje zgradbe vezij CPLD .