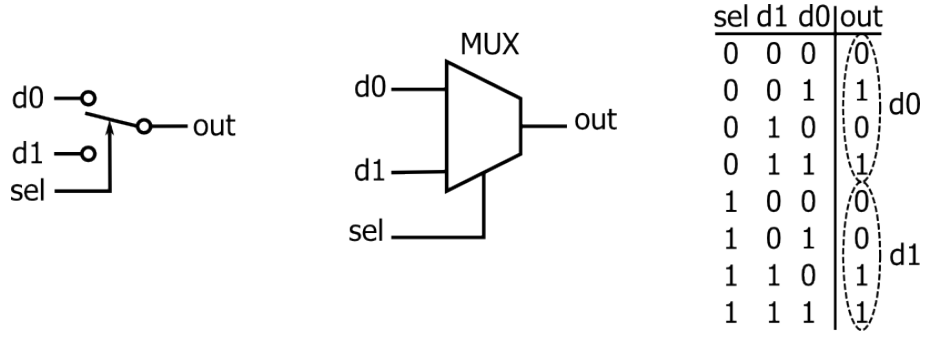
3
Osnovni gradniki

Digitalna vezja in sisteme sestavljamo s povezavo vnaprej pripravljenih gradnikov, ki izvajajo različne logične funkcije. Gradniki so logična vezja, ki delimo na kombinacijska in sekvenčna. Kombinacijska vezja imajo izhod odvisen le od trenutne kombinacije na vhodu, pri sekvenčnih vezjih pa je odvisen še od shranjenega stanja. Obravnavali bomo kombinacijske izbiralnike, dekodirnike in osnovne pomnilne elemente.

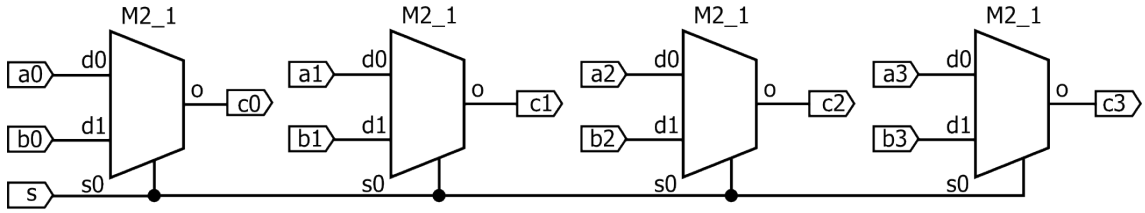
3.1 Kombinacijski izbiralnik

Kombinacijski izbiralnik ali multiplekser deluje kot preklopno stikalo, ki prenese na izhod vrednost enega izmed podatkovnih vhodov. Izbiralnik na sliki 3.1 ima dva podatkovna vhoda, izbirni vhod in en izhod. Ko je izbirni signal sel enak 0, dobimo na izhodu vrednosti iz vhoda $d0$, sicer pa vrednosti iz vhoda $d1$, kot prikazuje pravilnostna tabela.



Slika 3.1: Izbiralnik 2-1: stikalna shema, simbol in pravilnostna tabela.

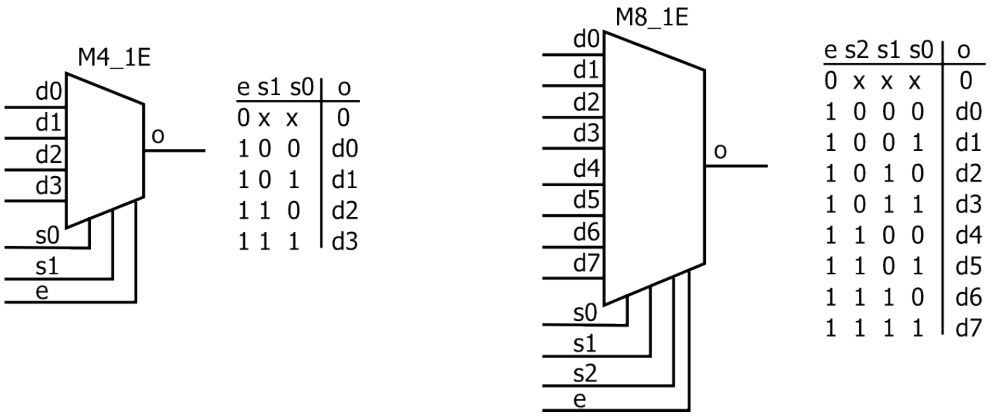
Podatkovni vhodi in izhod so lahko večbitne vrednosti. Izbiralnik z k-bitnimi podatkovnimi signali je narejen iz k osnovnih izbiralnikov pri katerih izbirni vhod vežemo skupaj. Primer izbiralnika 2-1, ki na 4-bitni izhod c prenese enega izmed 4-bitnih vhodnih signalov a ali b je prikazan na sliki 3.2.



s	c3 c2 c1 c0
0	a3 a2 a1 a0
1	b3 b2 b1 b0

Slika 3.2: Vezava štirih izbralnikov v izbiralnik 4-bitnih vrednosti.

Naloga izbiralnika je prenašanje večjega števila vhodnih vrednosti preko enega izhodnega signala, kar izkoriščamo naprimer v komunikacijskih vmesnikih. Nekateri izbiralniki imajo dodaten vhodni signal e (angl. enable) za omogočanje izhoda. Kadar je ta signal na logični 0, je izhod postavljen na 0 ne glede na stanje ostalih vhodov, kadar je signal e enak 1, pa deluje izbiralnik normalno.



e	s1	s0	o
0	x	x	0
1	0	0	d0
1	0	1	d1
1	1	0	d2
1	1	1	d3

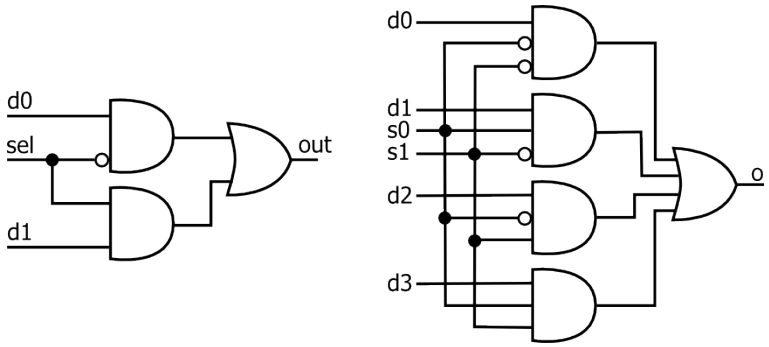
e	s2	s1	s0	o
0	x	x	x	0
1	0	0	0	d0
1	0	0	1	d1
1	0	1	0	d2
1	0	1	1	d3
1	1	0	0	d4
1	1	0	1	d5
1	1	1	0	d6
1	1	1	1	d7

Slika 3.3: Izbiralnika 4-1 in 8-1 z dodatnim vhodom za omogočanje e.

V splošnem imajo izbiralniki več podatkovnih vhodov in večbitni izbirni vhod. Izbiralnik z oznako n-1 ima n-bitni izbirni vhod in 2^n podatkovnih vhodov. Slika 3.3 prikazuje simbole izbiralnikov 4-1 in 8-1. Podrobnosti delovanja teh izbiralnikov razberemo iz pravilnostnih tabel, ki so zapisane v zgoščeni obliki. Tabele ne vsebujejo vseh kombinacij vhodnih signalov, ker bi že za manjši izbiralnik 4-1 potrebovali kar 128 vrstic. Ko je vhodni signal e postavljen na 0, je izhod definiran ne glede na stanje ostalih vhodov, zato so njihova stanja označena z x.

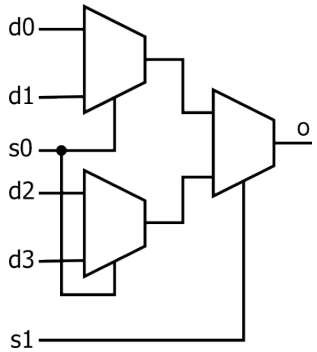
3.1.1 Izdelava izbiralnikov

Izbiralnik je kombinacijsko vezje, ki ga lahko naredimo iz osnovnih logičnih vrat. Vezje sestavimo iz vrat AND, ki prepuščajo izbrani podatkovni vhod in vrat OR, s katerimi združimo vse izhode. Izbirni signali gredo neposredno ali pa preko negatorjev na vrata AND. Kadar je kombinacija teh signalov na logični 1, bo izhod vrat enak podatkovnemu vhodu, v vseh ostalih primerih pa bo izhod enak 0. S primerno vezavo izbirnega signala zagotovimo, da bodo le ena logična vrata prepuščala izbrani vhodni signal. Slika 3.4 prikazuje izvedbo izbiralnikov 2-1 in 4-1 z logičnimi vrati.



Slika 3.4: Izvedba izbiralnikov 2-1 in 4-1 z logičnimi vrati.

Število podatkovnih vhodov izbiralnika lahko enostavno podvojimo z zaporedno vezavo treh izbiralnikov. Na sliki 3.5 je vezje izbiralnika 4-1, ki je narejeno iz treh manjših izbiralnikov 2-1.



Slika 3.5: Izvedba izbiralnika 4-1 z uporabo izbiralnikov 2-1.

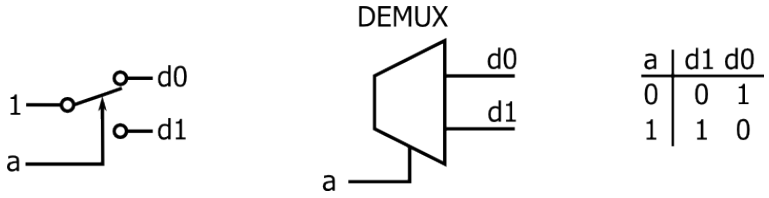
Razlika med obema prikazanima izvedbama izbiralnika 4-1 je v vrsti in številu logičnih vrat; prva zahteva 4 trivhodna AND in ena OR, druga pa 6 dvovhodnih AND in 3 vrata OR. Pomembno je tudi število nivojev logičnih vrat med vhodom in izhodom; pri prvi izvedbi sta dva nivoja, pri drugi pa so štirje. Zaradi zakasnitev na štirih zaporednih logičnih vratih je drugo vezje počasnejše v primerjavi s prvim.

3.2 Kombinacijski dekodirnik

Z izrazom dekodirnik označujemo digitalna vezja, ki pretvarjajo eno obliko binarne kode v drugo. Med kombinacijske dekodirnike spadajo razdeljevalnik, binarni dekodirnik in splošno dekodirno vezje.

3.2.1 Razdeljevalnik in binarni dekodirnik

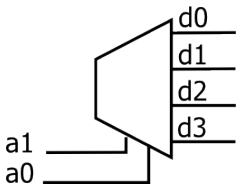
Razdeljevalnik ali demultiplekser ima nasprotno funkcijo od izbiralnika in zrcaljen simbol, kot prikazuje slika 3.6. Razdeljevalnik ima v splošnem n vhodnih signalov in 2^n izhodnih signalov. Izhodnih signalov je toliko, kolikor je dvojiških kombinacij na vhodu: enovhodni razdeljevalnik ima 2 izhoda, 2-vhodni ima 4 izhode, 3-vhodni ima 8 izhodov, 4-vhodni pa 16 izhodov. Razdeljevalnik ima le na enem izbranem izhodu logično 1 in sicer na tistem, ki ustreza trenutni vhodni kombinaciji.



a	d1	d0
0	0	1
1	1	0

Slika 3.6: Razdeljevalnik 1-2: stikalna shema, simbol in pravilnostna tabela.

Slika 3.12 prikazuje grafični simbol dvobitnega razdeljevalnika in njegovo pravilnostno tabelo. Pri vhodni kombinaciji 0 0 je logična 1 na izhodu $d0$, pri kombinaciji 0 1 je logična 1 na izhodu $d1$, itn.

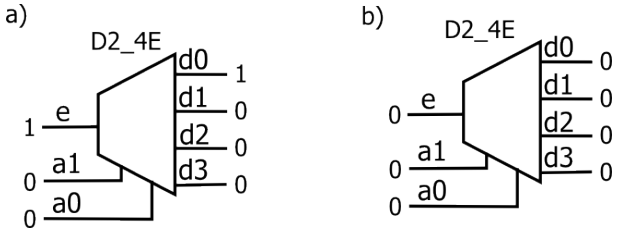


a1	a0	d3	d2	d1	d0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Slika 3.7: 2-bitni razdeljevalnik.

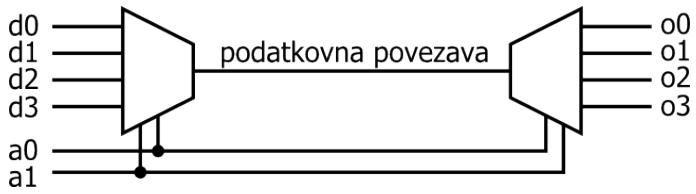
V digitalnem vezju uporabljamo razdeljevalnik tam, kjer želimo s kombinacijo na vhodih aktivirati natančno enega izmed množice gradnikov. Kombinacijo stanj na vhodih obravnavamo kot naslov (a, angl. address), ki je dodeljen izbranemu gradniku vezja. Razdeljevalnik v tem primeru služi kot dekodirnik naslovov.

Razdeljevalnik z dodatnim vhodom za omogočanje imenujemo *Binarni dekodirnik*. Signal za omogočanje obravnavamo kot podatkovni vhod, katerega vrednost se prenese na izbrani izhod. Kadar je signal e enak 1, dobimo na izbranem izhodu logično 1, kadar je e enak 0, so pa vsi izhodi na logični 0, kot prikazuje slika 3.8.



Slika 3.8: Binarni dekodirnik a) pri e=1 deluje normalno, b) pri e=0 so vsi izhodi 0.

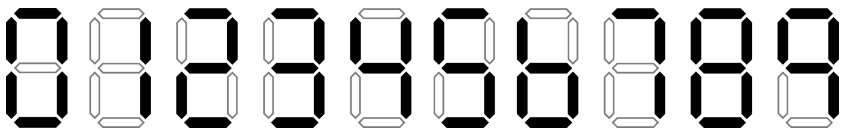
Binarni dekodirnik uporabljamo v komunikacijskih sistemih. Primer uporabe prikazuje slika 3.9, kjer z izbiralnikom združimo več vhodov in jih prenesemo preko ene podatkovne povezave, binarni dekodirnik pa opravi inverzno operacijo.



Slika 3.9: Vezje za kombiniranje (multipleksiranje) podatkovnih signalov.

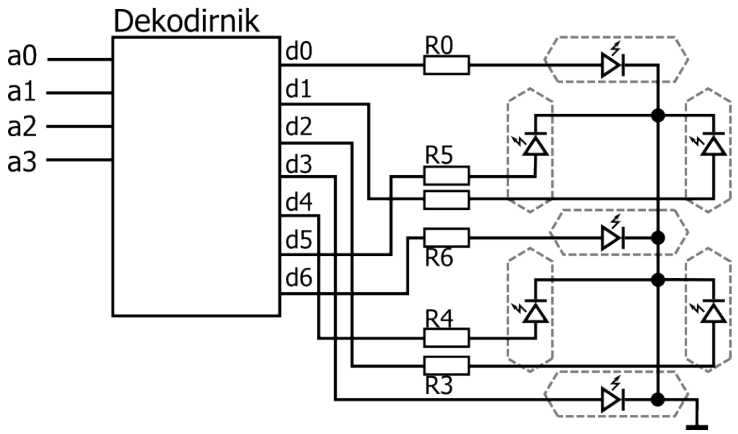
3.2.2 7-segmentni dekodirnik

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. Med bolj znanimi je dekodirnik za 7-segmentne prikazovalnike. Prikazovalnik je sestavljen iz sedmih svetlečih diod, ki so razporejene tako, da lahko prikažejo desetiške števke (slika 3.10).



Slika 3.10: Prikaz desetiških števk na 7-segmentnem prikazovalniku.

Desetiške števke predstavimo v dvojiški obliki s 4-bitno vrednostjo. Dekodirnik pretvarja 4-bitne vhode $a0$ do $a3$ v 7-bitne izhode $d0$ do $d6$, ki so vezani na posamezne segmente za prikaz desetiških števk. Slika 3.11 prikazuje priklop dekodirnika na prikazovalnik in pravilnostno tabelo za prikaz števk med 0 in 9.



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

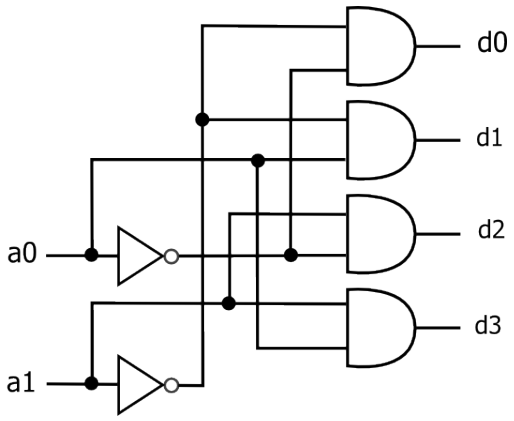
Slika 3.11: Vezava dekodirnika za 7-segmentni prikazovalnik in pravilnostna tabela.

Poznamo prikazovalnike LED s skupno katodo in prikazovalnike LED s skupno anodo. Segmente prikazovalnikov s skupno katodo prižigamo z logično 1, katoda vseh svetlečih diod pa je vezana na maso, kot prikazuje slika 3.11. Prikazovalniki s skupno anodo pa imajo skupni priključek vezan na Vdd in posamezne segmente prižigamo z logično 0. V tem primeru moramo izhode dekodirnika negirati, saj delujejo LED v negativni logiki. Nekateri prikazovalniki imajo še dodatno diodo (signal $d7$) za prikaz decimalne pike.

3.2.3 Izdelava razdeljevalnikov

Do izvedbe razdeljevalnika z osnovnimi logičnimi vrati pridemo tako, da zapišemo logične enačbe za posamezne signale. V dvobitnem razdeljevalniku je izhod $d0$ 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 = NOT a0 AND NOT a1
d1 = a0 AND NOT a1
d2 = NOT a0 AND a1
d3 = a0 AND a1



Slika 3.12: Izvedba 2-bitnega razdeljevalnika z logičnimi vrati.

Razdeljevalnik 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.

Binarni dekodirnik ima enako strukturo kot izbiralnik. Dodaten vhodni signal e pripeljemo na vsa logična vrata AND, ker predstavlja dodaten pogoj, da je na izbranem izhodu logična 1. V shemi dvobitnega binarnega dekodirnika bi tako imeli 3-vhodna logična vrata AND.

3.3 Načrtovanje vezij z izbiralniki

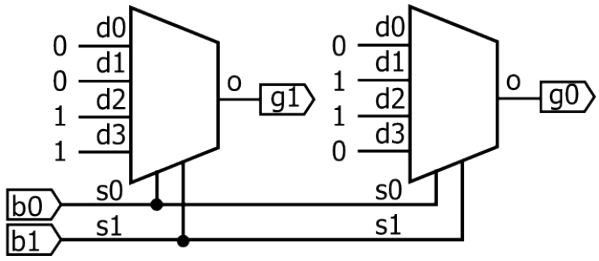
Kombinacijska vezja naredimo na podlagi pravilnostne tabele z osnovnimi logičnimi vrati ali pa z izbiralniki. V pravilnostni tabeli izbiralnika so na izhodu vsi podatkovni vhodi. Če povežemo na podatkovne vhode konstantne vrednosti 0 ali 1, lahko naredimo poljubno pravilnostno tabelo.

Za izvedbo kombinacijskega vezja z več izhodnimi signali uporabimo več izbiralnikov. V splošnem kombinacijskem vezju, brez optimizacije, potrebujemo toliko izbiralnikov, kot je izhodnih signalov. Velikost izbiralnikov je odvisna od števila vhodov kombinacijskega vezja.

3.3.1 Grayev dekodirnik

Naredimo dvobitni dekodirnik, ki pretvarja vhodno dvojiško kodo v Grayevo kodo. Značilnost Grayeve kode je, da se pri zaporednih vrednostih vedno spremeni le en bit. Slika 3.13 prikazuje logično tabelo 2-bitne pretvorbe in izvedbo vezja z izbiralniki.

b1	b0	g1	g0
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0



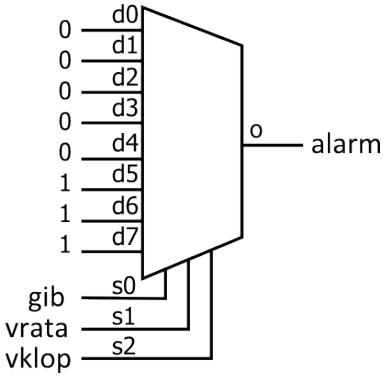
Slika 3.13: Grayev dekodirnik: pravilnostna tabela in izvedba z izbiralniki.

Dvobitni dekodirnik ima dva vhodna signala in dva izhodna signala. Za vsak izhod v splošnem potrebujemo svoj izbiralnik 4-1, ki ima dva izbirna vhoda. Vezje lahko poenostavimo, če upoštevamo da so vrednosti signala $g1$ kar enake signalu $b1$, zato v praksi potrebujemo le en izbiralnik za določitev signala $b0$.

3.3.2 Avtomobilski alarm z izbiralnikom

Z izbiralnikom naredimo vezje avtomobilskega alarma, ki se sproži kadar je vklopljen in je aktiviran senzor gibanja ali pa so odprta vrata. Najprej naredimo pravilnostno tabelo za izhodni signal. Vezje ima 3 vhodne signale in v pravilnostni tabeli je $2^3 = 8$ kombinacij.

vklop	vrata	gib	alarm
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Slika 3.14: Izvedba avtomobilskega alarma z izbiralnikom.

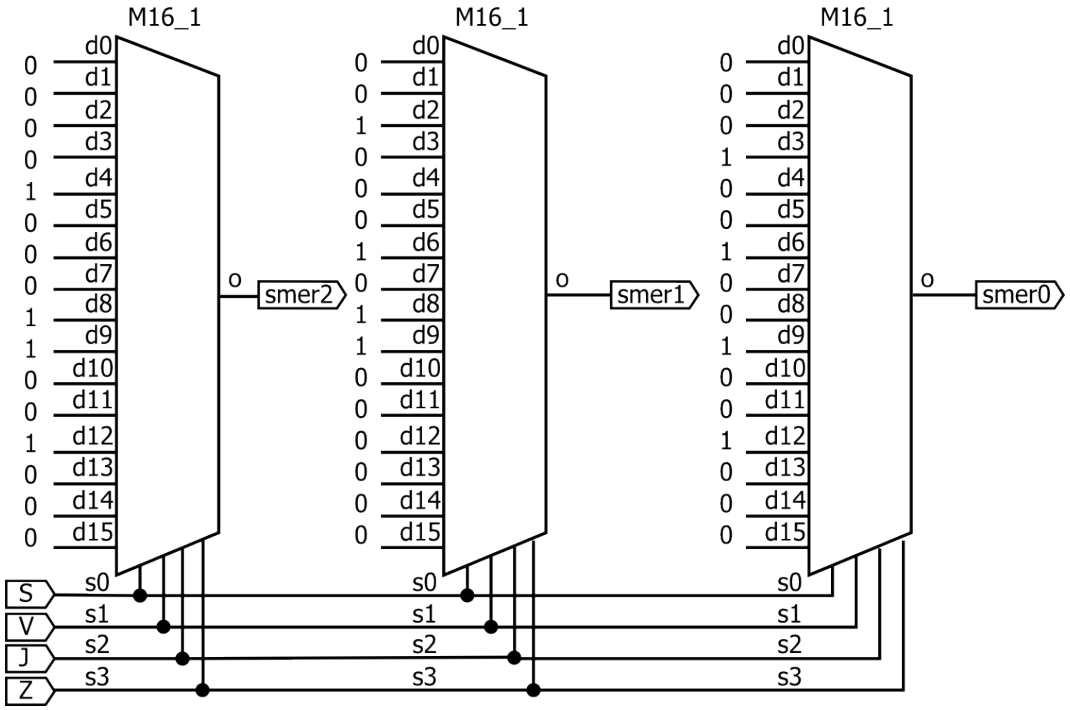
3.3.3 Dekodirnik za smer vetra

Naredimo vezje za dekodiranje smeri vetra. Iz vetrnice dobimo 4 signale: S, J, V in Z, med katerimi je eden ali dva postavljen na 1, ostali pa so 0. Naloga dekodirnika je pretvoriti 4-bitni vhod v 3-bitno kombinacijo, ki kodira smer vetra po tabeli:

veter	Z	J	V	S	smer2:0		
-	0	0	0	0	0	0	0
S	0	0	0	1	0	0	0
V	0	0	1	0	0	1	0
SV	0	0	1	1	0	0	1
J	0	1	0	0	1	0	0
-	0	1	0	1	0	0	0
JV	0	1	1	0	0	1	1
-	0	1	1	1	0	0	0
Z	1	0	0	0	1	1	0
SZ	1	0	0	1	1	1	1
-	1	0	1	0	0	0	0
-	1	0	1	1	0	0	0
JZ	1	1	0	0	1	0	1
-	1	1	0	1	0	0	0
-	1	1	1	0	0	0	0
-	1	1	1	1	0	0	0

Slika 3.15: Pravilnostna tabela za dekodiranje smeri vetra.

V pravilnostni tabeli, ki ima $2^4 = 16$ vrstic moramo določiti izhode za vse kombinacije, tudi za tiste ki jih ne pričakujemo na izhodu (označene s -).



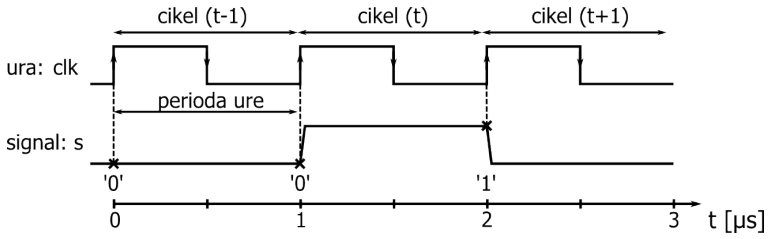
Slika 3.16: Dekodirnik smeri vetra z izbiralniki.

V vezju kodirnik smeri vetra uporabimo za vsak izhodni bit po en izbiralnik 16-1, kot prikazuje slika 3.16. Na podatkovne vhode postavimo konstantne vrednosti, ki predstavljajo posamezen stolpec pravilnostne tabele. Izbirni vhodi $s0$ do $s3$ 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.

3.4 Sekvenčni gradniki

Sekvenčni gradniki so logična vezja, pri katerih je izhod odvisen od vhodov in od shranjenega stanja. Osnovni sekvenčni gradniki so pomnilni elementi v katerih se shranjujejo logične vrednosti. Najpreprostejši gradniki shranijo le en bit (logično 0 ali 1), sestavljeni gradniki pa shranijo večbitno vrednost. *Stanje vezja* predstavljajo vse logične vrednosti, ki so v nekem trenutku shranjene v vezju.

V sekvenčnih vezjih imamo običajno poseben signal, ki določa kdaj naj se izhodi spremenijo. Takšen signal je v obliki periodičnih impulzov in ga označujemo z imenom ura (angl. clock, clk). Uro dobimo iz posebnega vezja, ki se imenuje oscilator. V digitalnih sistemih bomo najpogosteje naleteli na kvarčne oscilatorje, s katerimi dobimo stabilno uro natančno določene frekvence.



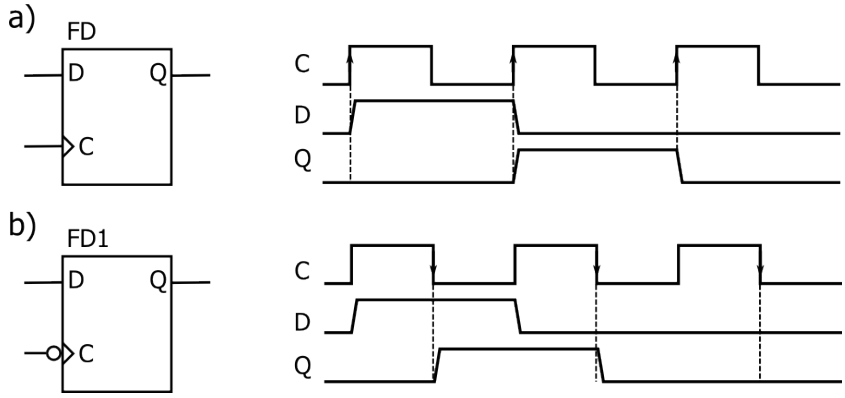
Slika 3.17: Proženje na nivo ali na fronto ure.

Slika 3.17 prikazuje primer časovnega diagrama urnega signala. Glavni parameter ure je perioda, ki predstavlja časovni interval med sosednjimi impulzi. Časovni interval ene periode v katerem se stanje spremeni iz nizkega v visoko (ali obratno), imenujemo *urni cikel*. Včasih je pomembno tudi razmerje med trajanjem visokega in nizkega cikla, ki ga podajamo v odstotkih. Obratna vrednost periode je frekvenca, v primeru s slike velja: $f = 1/1\,\mu\mathrm{s} = 1\,\mathrm{MHz}$. Trenutek prehoda iz nizkega v visoko stanje imenujemo prednja ali naraščajoča fronta ure. Prehod iz visokega v nizko stanje pa imenujemo zadnja ali padajoča fronta. Fronte ure lahko na časovnem diagramu posebej označimo s puščicami.

3.4.1 Flip-flop

Pomnilni elementi, ki spreminjajo stanje ob taktu ure, se imenujejo *flip-flopi*. Obstaja veliko vrst flip-flopov, vsi pa imajo skupno lastnost, da se izhodni signal spreminja le ob prednji ali zadnji fronti ure, kot prikazuje slika 3.18.

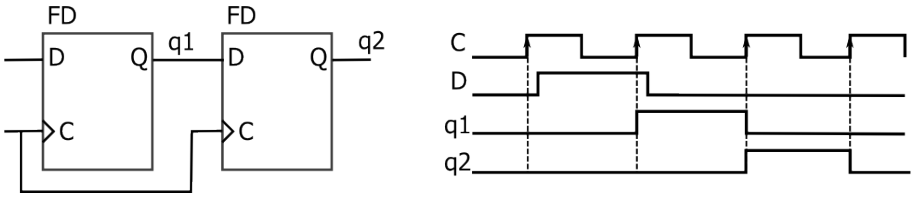
Podatkovni flip-flop (angl. Data Flip-Flop, DFF) prenaša logično stanje iz vhoda D na izhod Q ob fronti ure. Vhodni podatkovni signal določa vrednost, ki bo na izhodu vezja v naslednjem urnem ciklu: $Q(t) = D(t-1)$, $Q(t+1) = D(t)$, itn.



Slika 3.18: Simbola in časovna diagrama podatkovnih flip-flopov.

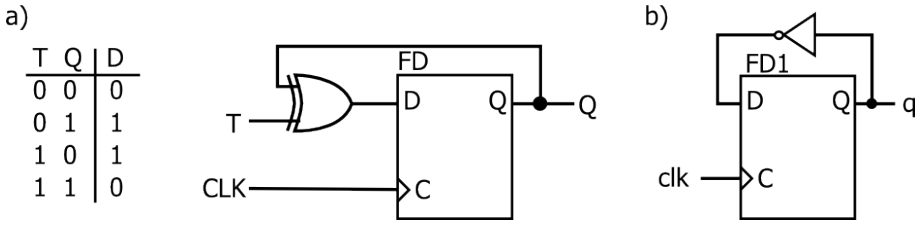
Podatkovni flip-flop na sliki 3.18 b), ki je prožen na zadnjo fronto ure, je označen z negacijo (krožcem) pred vhodom za uro.

Če vežemo drug za drugim dva podatkovna flip-flopa, bomo zakasnili signal za dva cikla, kot prikazuje slika 3.19. Takšna vezava je osnova za izvedbo elementov, ki jih imenujemo pomikalni registri.



Slika 3.19: Zakasnitev signalov pri prehodu čez zaporedne flip-flope D.

Preklopni flip-flop ali flip-flop T (angl. Toggle) ima krmilni signal T, ki povzroči, da izhod ob naraščajoči fronti ure zamenja vrednost: $Q(t+1) = \text{NOT } Q(t)$. Kadar je signal T na 0, izhod ohranja vrednost. Preklopni flip-flop naredimo iz podatkovnega flip-flopa in logike na vhodu. Podatkovni flip-flop izvaja funkcijo: $Q(t+1) = D(t)$, logika na vhodu pa določa $D(t)$ v odvisnosti od vhoda T in stanja $Q(t)$, kar opišemo s pravilnostno tabelo:

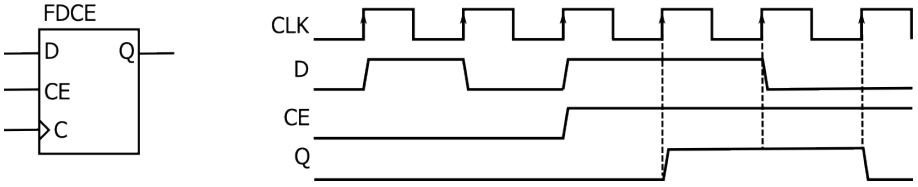


Slika 3.20: a) tabela in zgradba flip-flopa T, b) flip-flop s stalnim preklapljanjem izhoda.

Pravilnostna tabela določa logično funkcijo na vhodu flip-flopa, ki je v našem primeru XOR. Če potrebujemo flip-flop, ki stalno preklaplja izhod, uporabimo inverter med izhodom in vhodom flip-flopa (slika 3.20b).

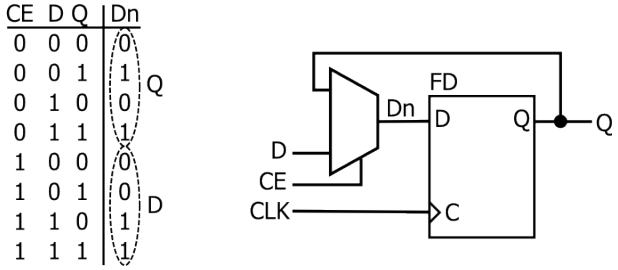
3.4.2 Register

Register je podatkovni pomnilni gradnik s signalom za omogočanje CE. Kadar je CE na 0, bo izhod registra ohranjal vrednost, kot prikazuje slika 3.21.



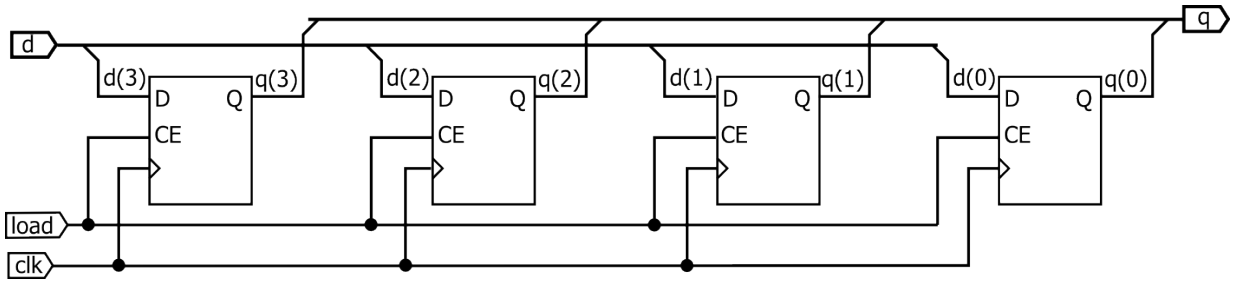
Slika 3.21: Simbol in časovni diagram flip-flopa, ki ohranja stanje izhoda.

Ohranjanje vrednosti izhoda opisuje enačba: $Q(t+1) = Q(t)$. Delovanje registra opišemo s tabelo v kateri so na levi strani vse kombinacije vhodov CE, D in stanja Q, na desni strani pa je signal Dn, ki določa naslednje stanje $Q(t+1)$. Tabela je identična pravilnostni tabeli izbiralnika 2-1, ki predstavlja logiko na vhodu flip-flopa.



Slika 3.22: Tabela in zgradba enobitnega registra.

Register, ki shranjuje večbitne besede naredimo z vzporedno vezavo flip-flopov FDCE. Slika 3.23 prikazuje logično shemo 4-bitnega registra z uro in signalom $Load$ za nalaganje nove vrednosti.

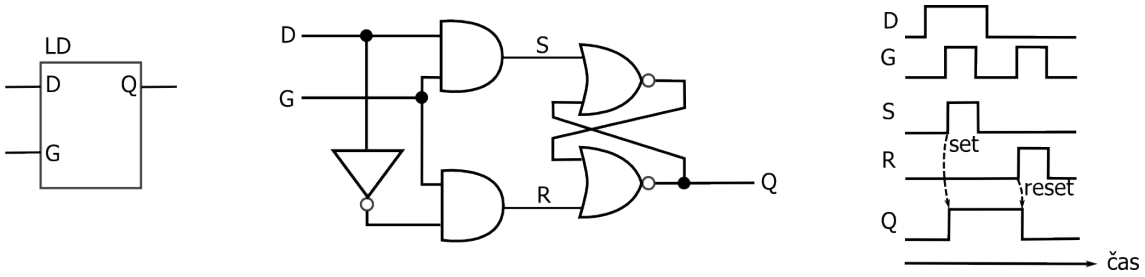


Slika 3.23: Shema 4-bitnega registra.

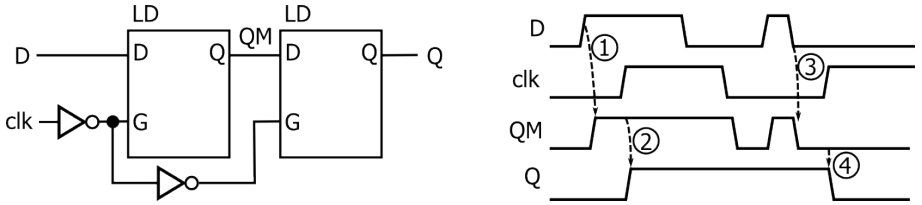
3.4.3 Izvedba flip-flopov

Za izvedbo so najbolj enostavni asinhroni pomnilni gradniki oz. zapahi. Zapah z oznako SR ima dva vhoda: S (angl. Set) in R (angl. Reset). Zapah ohranja vrednost na izhodu, kadar sta oba vhoda na 0. Kadar je vhod S na 1, se izhod postavi na 1, kadar je R na 1 pa se izhod postavi na 0. V primeru, ko sta oba vhoda postavljena na 1, pa zapah SR ni stabilen. V praksi to pomeni, da ne moremo predvideti stanja v katero se bo postavil, možno je celo da izhod stalno preklaplja (oscilira) med logično 0 in 1. Osnovni stabilen gradnik je zapah D, prikazan na sliki 3.24.

Zapah D ima podatkovni vhod D, kontrolni vhod (ali vrata) za omogočanje G (angl. Gate) in izhod Q. Kadar je $G = 1$, se stanje iz podatkovnega vhoda prenese na izhod. Pravimo, da je zapah transparenten in vse spremembe vhoda z majhno zakasnitvijo prenaša na izhod. Ko gre signal G na 0, se stanje izhoda zapahne in ohranja zadnjo vrednost. Podatkovni flip-flop lahko naredimo s povezavo dveh zapahov:



Slika 3.24: Zapah D: simbol, logična shema in časovni diagram.



Slika 3.25: Izvedba flip-flopa D iz dveh zapahov.

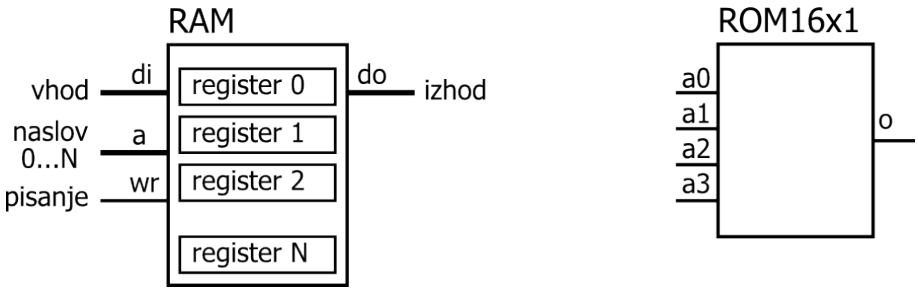
Slika 3.25 prikazuje izvedbo flip-flopa in časovni potek signalov. Prvi zapah je transparenten, ko je signal ure (clk) na 0, ob prehodu ure na 1 pa zapahne zadnjo vrednost (1). Obenem postane transparenten drugi zapah, ki prenese vrednost na izhod (2). Če pride do spremembe na podatkovnem vhodu, ko je ura na 0, se sprememba sicer prenese na izhod prvega zapaha (3), ne pa na tudi na izhod vezja. Do spremembe na izhodu vezja pride šele ob naslednji naraščajoči fronti ure (4), tako da dobi izhod vrednost iz podatkovnega vhoda tik pred fronto ure.

3.5 Pomnilnik

Pomnilniki so elementi za shranjevanje podatkov in ukazov, ki jih npr. obdelujejo računalniški digitalni sistemi. Pomnilnik shrani več podatkovnih besed, vendar v posameznem ciklu zapisujemo ali beremo le eno besedo naenkrat. Elektronske polprevodniške pomnilnike označujemo s kratico RAM (angl. Random Access Memory), kar pomeni da imajo enak čas za dostop do naključno izbrane pomnilniške besede. Oznaka izhaja iz primerjave s pomnilniki, pri katerih so besede zapisane na magnetnem traku in je čas dostopa do besede odvisen od trenutnega položaja elektromagnetne glave na traku.
Signale na priključkih pomnilnika RAM razdelimo na:

- podatkovno vodilo (ang. *data*), ki je lahko ločeno na podatkovni vhod (datain) in podatkovni izhod (dataout),
- naslovno vodilo (ang. *address*) s katerim izberemo posamezno besedo in
- krmilne signale, ki določajo operacije (branje, pisanje, mirovanje).

Pomnilne celice si lahko predstavljamo kot registre. Vsak izmed registrov ima svoj naslov v obliki binarnega števila med 0 in N. Na naslovno vodilo postavimo vrednost s katero izberemo



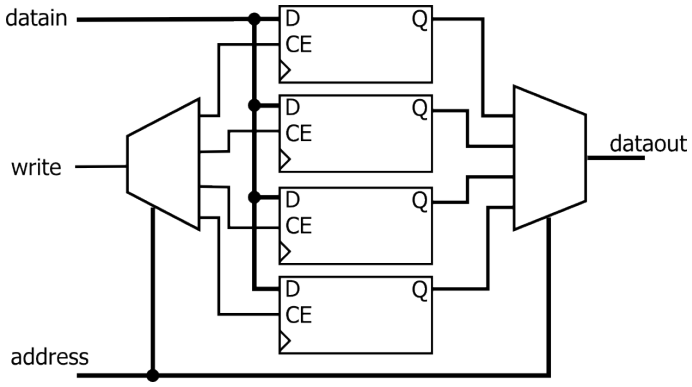
Slika 3.26: Blokovna shema pomnilnika RAM in ROM

enega izmed registrov, s krmilnimi signali pa določimo ali bomo v register naložili novo besedo, ali pa bomo brali iz registra.

Pomnilniki z oznako ROM (angl. Read Only Memory) imajo vnaprej zapisano vsebino, ki jo preberemo tako, da nastavimo naslovne signale. Slika 3.26 prikazuje simbol pomnilnika ROM s štirimi naslovnimi signali in enim izhodom. Takšen pomnilnik, v katerega lahko shranimo 16 bitov, najdemo v celicah programirljivega vezja.

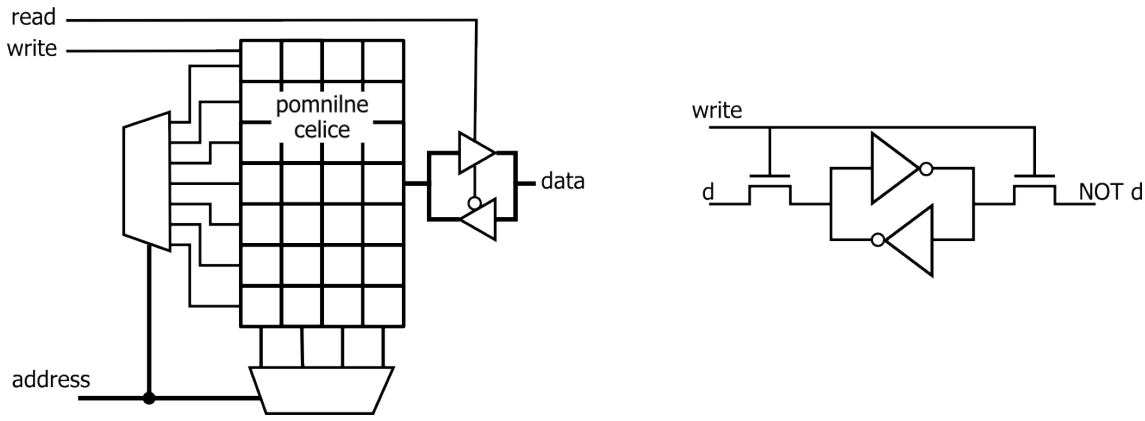
3.5.1 Izdelava pomnilnikov

Pomnilnik je mogoče sestaviti iz elementov, ki smo jih do sedaj spoznali: registrov, binarnega dekodirnika in izbiralnika, kot prikazuje slika 3.27. Registri imajo podatkovne vhode vezane skupaj. Vsakemu registru dodelimo naslov *address* in preko binarnega dekodirnika omogočimo vpis besede v naslovljeni register, kadar je aktiven krmilni signal *write*. Branje poteka tako, da nastavimo naslov in izbrana beseda se preko izbiralnika prenese na podatkovni izhod.



Slika 3.27: Izvedba pomnilnika RAM iz registrov.

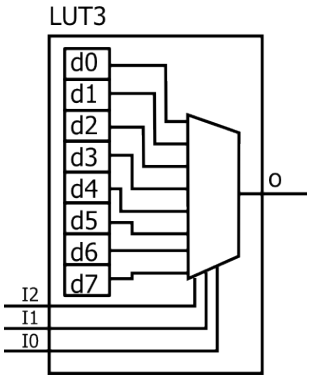
Pomnilnik iz registrov potrebuje za shranjevanje besed veliko elektronskih elementov, ker imajo registri iz flip-flopov D kompleksno zgradbo. V praksi zato uporabljamo učinkovitejše in cenejše pomnilnike sestavljene iz matrike pomnilnih celic. Slika 3.28 prikazuje zgradbo statičnega pomnilnika RAM in posamezne pomnilne celice.



Slika 3.28: Izvedba statičnega pomnilnika in pomnilna celica.

Razdeljevalniki poskrbijo za izbiro pomnilnih celic, ki so razporejene v matriko. Vsaka celice je sestavljena iz dveh inverterjev za shranjevanje podatka in transistorjev za vpis nove vrednosti. Za vsak bit potrebujemo skupaj 6 transistorjev, kar je precej manj kot pri izvedbi s flip-flopom D. Še bolj učinkoviti so dinamični pomnilniki, ki hranijo zapis v enem samem transistorju, vendar imajo zahtevnejšo krmilno logiko.

V programirljivih vezjih uporabljamo za izvedbo pomnilnika ROM vpogledno tabelo, kot prikazuje slika 3.29.



Slika 3.29: Pomnilnik ROM je narejen kot vpogledna (angl. Look-Up) tabela LUT.

Naloge

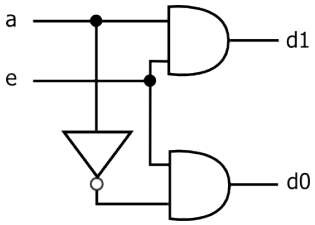
1. Kakšen bo izhod vezja, ko je na vhodu $a = 1, e = 1$?

   (a) $d0 = 0, d1 = 0$

   (b) $d0 = 1, d1 = 0$

   (c) $d0 = 0, d1 = 1$

   (d) $d0 = 1, d1 = 1$



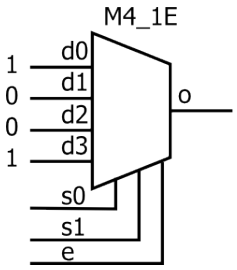
2. Na podatkovnih vhodih izbiralnika so konstantne vrednosti, kot prikazuje slika. Pri kateri kombinaciji kontrolnih vhodov bo izhod enak 1?

   (a) $s0 = 0, s1 = 0, e = 0$

   (b) $s0 = 0, s1 = 0, e = 1$

   (c) $s0 = 0, s1 = 1, e = 1$

   (d) $s0 = 1, s1 = 0, e = 1$



3. Kateri pomnilni gradnik predstavlja časovni diagram?

   (a) zapah D

   (b) zapah SR

   (c) flip-flop D prožen na prednjo fronto

   (d) flip-flop D prožen na zadnjo fronto