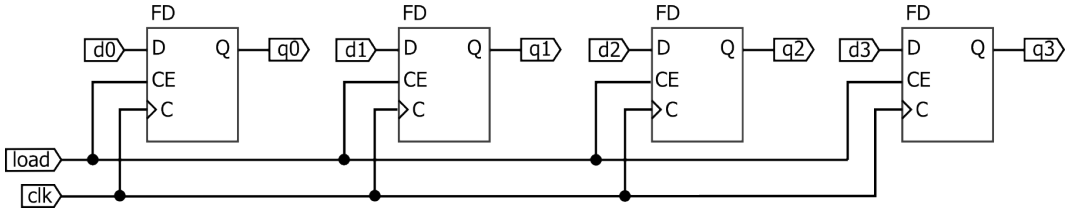
6 Načrtovanje vezij z registri

6.1 Operacije z registri

Digitalni sistemi vsebujejo kombinacijske in sekvenčne logične gradnike. Osnovni sekvenčni gradniki so flip-flopi in registri. Registri shranjujejo podatke, ki jih kombinacijsko vezje obdeluje. Digitalni sistem lahko začnemo načrtovati tako, da ga razdelimo na *podatkovni podsistem* in *kontrolni podsistem*. Podatkovni podsistem ali podatkovno pot (angl. datapath) tvorijo kombinacijski gradniki in zbirka registrov preko katerih potujejo in se preoblikujejo podatki. Pravimo, da podatkovni podsistem izvaja operacije obdelave podatkov. Kontrolni podsistem pa določa zaporedje izvrševanja operacij.

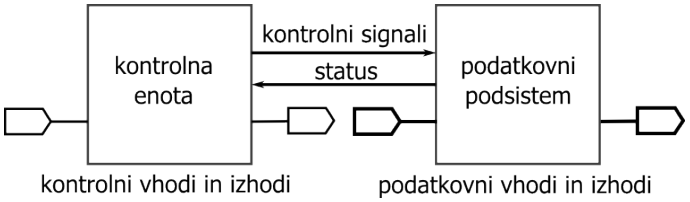
V sinhronem digitalnem sistemu vsi registri spreminjajo izhode ob fronti sistemske ure. Kadar ne želimo, da register prenaša podatke na izhod ob vsaki fronti ure, uporabimo registre s signalom za omogočanje. Ta signal predstavlja kontrolni vhod v register, ki ga včasih imenujemo tudi signal za nalaganje (angl. load) podatkov, kot prikazuje slika 6.1.



Slika 6.1: 4-bitni register s kontrolnim vhodom za nalaganje podatkov.

Pri načrtovanju digitalnega sistema si pomagamo z razdelitvijo sistema na podsisteme ali module. Podsistemi so med seboj povezani s signali za prenos podatkov in kontrolnimi signali. Slika 6.2 prikazuje delitev digitalnega sistema na podatkovni in kontrolni podsistem (ali krajše

kontrolno enoto). Kontrolni signali so binarni signali, ki aktivirajo različne operacije obdelave podatkov. Kontrolna enota pošilja zaporedje kontrolnih signalov v podatkovni podsistem in iz njega dobiva informacijo o stanju (status). Od zunaj dobiva kontrolne vhode in proizvaja kontrolne izhode za interakcijo z drugimi deli sistema. Podatkovni podsistem ima zunanje podatkovne vhode in izhode.



Slika 6.2: Povezava med kontrolno enoto in podatkovno potjo.

Podatkovni podsistem določajo registri in operacije nad binarnimi podatki, ki jih hranijo registri. Primeri operacij so: naloži, seštej ali odštej, prištej ali pomakni podatek. Registri so osnovne komponente digitalnih sistemov. Prenos podatkov med registri in obdelavo podatkov imenujemo **registrski prenos** ali z angleško kratico RT (Register Transfer). Gradniki RT vezij so registri, ki izvajajo eno ali več elementarnih operacij. Elementarne operacije ali **mikrooperacije** so npr. nalaganje podatka v register ali prištevanje k vrednosti v registru. Mikrooperacija se običajno izvede paralelno na vseh podatkovnih bitih v enem urinem ciklu. Rezultat operacije se lahko shrani nazaj v register ali pa prenese v naslednji register.

Vzemimo dva registra $R1$ in $R2$, ki imata izhodna signala $r1$ in $r2$ z enakim številom podatkovnih bitov. Prenos podatka iz enega v drug register zapišemo z operacijo:

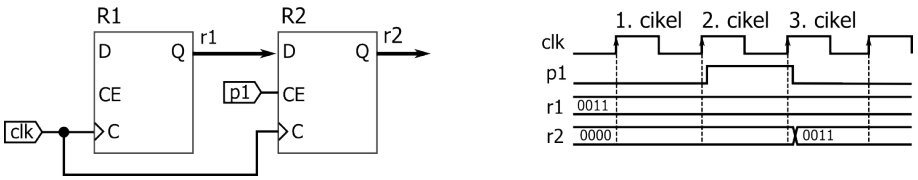
$$r2 <= r1$$

Prenos podatka smo označili s puščico $<=$. Register $R1$ predstavlja izvor podatkov, register $R2$ pa ponor ali ciljni register. Vsebina registra, ki je izvor podatkov, se pri prenosu ne spremeni, spremeni se le podatek v ciljnem registru. Vezje vsebuje dva registra in povezavo med podatkovnim izhodom izvora in vhodom ciljnega registra. Običajno ne želimo, da se prenos izvrši ob vsaki naraščajoči fronti ure, pri kateri registra prenašata vrednost na izhod.

Če želimo, da se podatek prenese le ob določenem pogoju, zapišemo registrski prenos s pogojem:

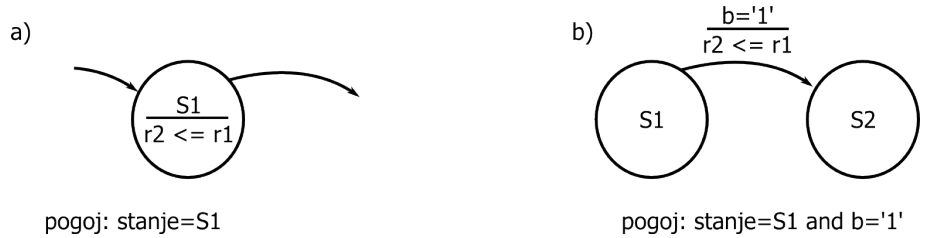
$$r2 <= r1 \mid pogoj$$

Pogoj je lahko vrednost nekega binarnega kontrolnega signala. Registrski prenos se bo na primer izvršil, kadar bo vrednost kontrolnega signala $p1$ enaka 1. Na tem mestu bi lahko zapisali poljuben enostaven ali sestavljen pogoj. Slika 6.3 prikazuje blokovno shemo vezja za pogojni prenos podatka med dvema registroma. Registra sta vezana na skupni urin signal clk. Prenos podatka v register $R2$ se izvrši ob aktivnem signalu $p1$, ki je vezan na kontrolni vhod za nalaganje tega registra.



Slika 6.3: Vezje za pogojni prenos podatkov med dvema registroma in časovni diagram.

V vezju so običajno vsi signali sinhronizirani z uro, zato tudi pričakujemo da se pogoj $p1$ aktivira ob naraščajoči fronti ure. V časovnem diagramu na sliki 6.3 se $p1$ postavi na 1 v drugem urinem ciklu. Prenos podatka pa se naredi ob naraščajoči fronti 3. urinega cikla. V sinhronih vezjih vedno velja, da se registrski prenos izvrši ob enem ciklu kasneje kot je izpolnjen pogoj. Razlog za takšno delovanje so zakasnitve, ki jih ima vsako realno vezje. Če se pogoj aktivira ob nekem urinem ciklu, bo zaradi zakasnitev vrednost kontrolnega signala postavljena na 1 nekoliko za fronto ure, drugi register pa bo sprejel novo vrednost šele ob naslednji naraščajoči fronti.



Slika 6.4: Pogojni registrski prenos: a) v stanju S1 ali b) ob prehodu iz S1 v S2.

Pogojni prenos podatkov med dvema registroma lahko definiramo v razširjeni obliki diagrama stanj. Pogoj za prenos je kar stanje v katerem je zapisana operacija prenosa 6.4a ali pa izvorno stanje in pogoj za prehod, če naj se prenos izvrši ob prehodu 6.4b.

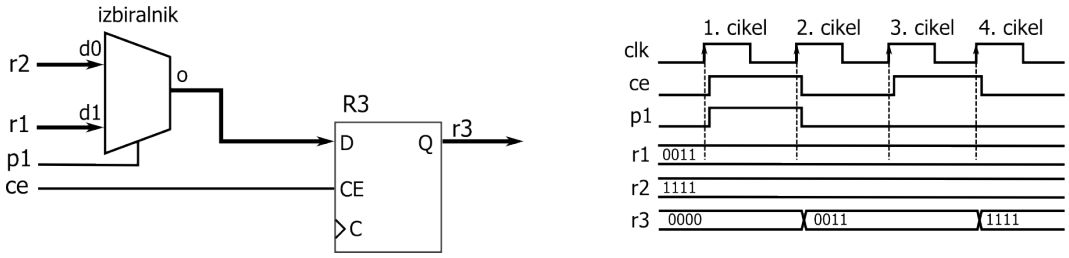
6.2 Mikrooperacije

Mikrooperacija je elementarna operacija nad podatki, katere rezultat se shrani v register ali pomnilnik. Mikrooperacije so osnovni elementi opisa digitalnih vezij na nivoju registrskih prenosov (angl. RTL) in omogočajo zelo kompakten opis vezja v jeziku HDL ali pa v razširjeni obliki diagrama stanj. Primerne so tudi za shematski opis podatkovne poti z vnaprej pripravljenimi gradniki. V digitalnih sistemih ločimo 4 vrste mikrooperacij:

- *registrski prenos* je prenos podatka iz enega v drug register,
- *aritmerična mikrooperacija* je računska operacija nad podatki v registrih,
- *logična mikrooperacija* izvrši logično operacijo nad biti v registru in
- *mikrooperacija pomika*, ki predstavlja pomik bitov v registru.

6.2.1 Registrski prenos

Osnovni prenos podatka med dvema registroma in pogojni prenos smo že spoznali. Nadgradnja osnovnega prenosa je izbirni prenos, kjer imamo več izvorov iz katerih naj se podatek shrani v register. Osnovno vezje je sestavljeno iz registra in izbiralnika, kot prikazuje slika 6.5.



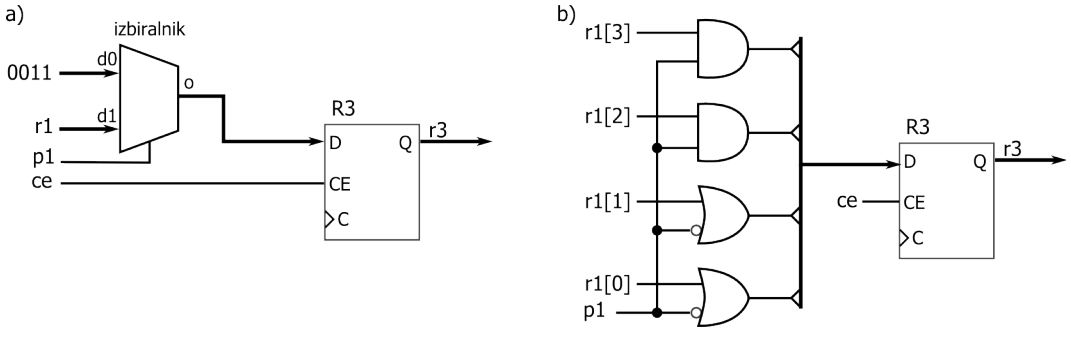
Slika 6.5: Vezje za registrski prenos z enim izbirnim signalom in časovni diagram.

V ciljni register $R3$ se ob aktivnem signalu $ce = 1$ shrani podatek iz vodila $r1$, kadar je $p1 = 1$, sicer pa iz vodila $r2$. Na časovnem diagramu vidimo, da se dejanski prenos podatka izvrši vedno en cikel za tem, ko sta postavljena kontrolna signala ce in $p1$. Mikrooperacijo za izbirni prenos s pogojem $p1$ zapišemo:

$$r3 <= r1 \mid p1 = 1 \text{ and } ce = 1$$

$$r3 <= r2 \mid p1 = 0 \text{ and } ce = 1$$

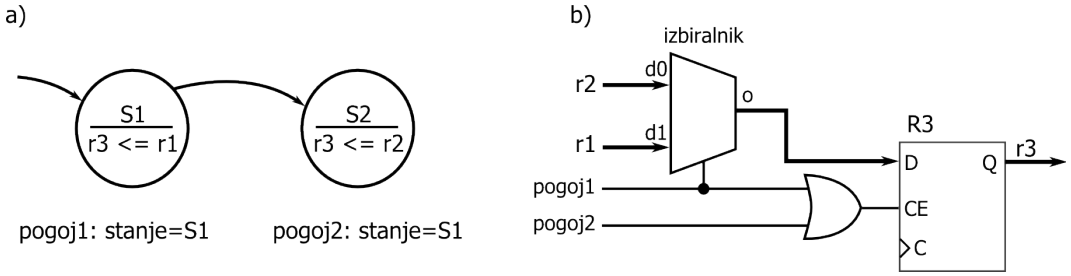
V primeru, ko je na enem izmed vhodov konstantna vrednost dobimo še nekoliko manjše vezje. Slika 6.6 prikazuje shemo vezja za izbirni prenos s 4 bitnim registrom $R3$, ki naloži vrednost $r1$ ob pogoju $p1 = 1$ in konstantno vrednost 0011 ob pogoju $p1 = 0$. Zaradi konstantnega vhoda v 4-bitni izbiralnik se le-ta poenostavi v 4 logična vrata, kot prikazuje slika 6.6b.



Slika 6.6: Vezje za izbirni prenos s konstanto: a) z izbiralnikom in b) z logičnimi vrati.

Kadar je $p1 = 0$ bo na izhodu logičnih vrat AND2 vrednost 0, ker velja: $r1[n]$ AND $0 = 0$, na izhodu vrat OR2B pa bo 1, zaradi: $r1[n]$ OR NOT $(0) = 1$. V primeru $p1 = 1$ pa lahko na podoben način dokažemo, da pride v obeh primerih na izhod vrat kar vrednost iz vhoda $r[n], n = 0..3$

Izbirni registrski prenos naredimo v razširjeni obliki diagrama stanj preprosto tako, da na več mestih (npr. v različnih stanjih) določimo pogojni registrski prenos v isti register. Izsek iz diagrama stanj na sliki 6.7a, pri katerem v vsakem od stanj določimo vrednost signala $r3$, predstavlja primer izbirnega prenosa. Signal $r3$ dobi vrednost signala $r1$, ko smo v prvem stanju (pogoj je $stanje = S1$). Ko smo v drugem stanju pa signal $r3$ dobi vrednost signala $r2$.



Slika 6.7: Izbirni registrski prenos a) v diagramu stanj in b) shema vezja.

V splošnem imamo več stanj in v vseh ostalih se vrednost signala $r3$ ohranja, zato bo ta signal v vezju izhod iz registra $R3$, ki ima tudi kontrolni vhod za nalaganje nove vrednosti. Nova vrednost se naloži, kadar je izpolnjen prvi ali drugi pogoj, zato so v vezju vrata OR, kot prikazuje slika 6.7b.

6.2.2 Aritmetične mikrooperacije

Kadar podatki v registru predstavljajo številsko vrednost, lahko z njimi naredimo računske ali aritmetične mikrooperacije. Osnovne aritmetične mikrooperacije so seštevanje, odštevanje, povečevanje in zmanjševanje vrednosti. Operacijo seštevanja vrednosti iz dveh registrov, ki se shranita v tretji register zapišemo:

$$r3 <= r1 + r2$$

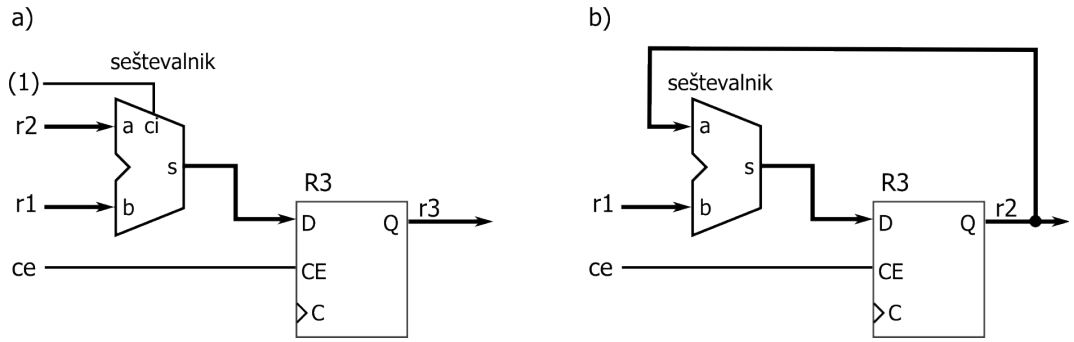
Za izvedbo te operacije potrebujemo 3 registre in kombinacijsko vezje, ki izvaja operacijo seštevanja, kot je npr. paralelni seštevalnik, kot prikazuje slika 6.8a. Mikrooperacija v obliki:

$$r2 <= r2 + r1$$

predstavlja akumulator, ki je vezje s povratno zanko, pri katerem enemu registru prištevamo vsebino drugega, kot prikazuje slika 6.8b. V praksi potrebuje akumulator vsaj še asinhroni reset signal ali pa dodatno izbirno logiko za nalaganje začetne vrednosti h kateri se ob naslednjih ciklih prišteva signal $r1$.

Odštevanje binarnih vrednosti naredimo z uporabo dvojiškega komplementa. Vrednost, ki jo želimo odšteti invertiramo in ji prištejemo 1 (vhodni prenos ci), nato pa kar seštejemo z drugo vrednostjo:

$$\text{namesto: } r3 <= r2 - r1, \quad \text{uporabimo: } r3 <= r2 + (\text{ NOT } r1 + 1)$$

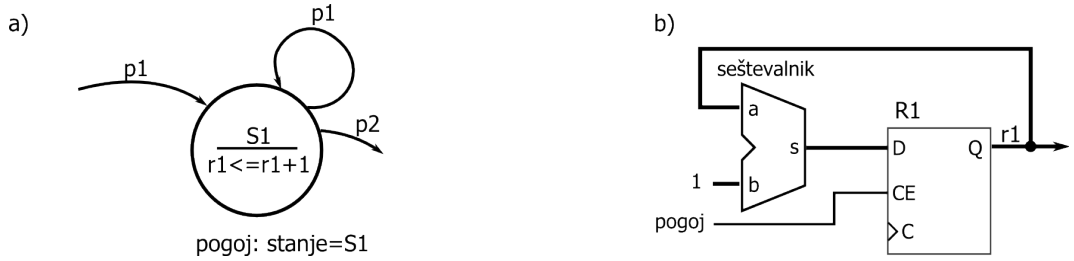


Slika 6.8: Vezje za mikrooperacijo seštevanja: a) vsota $r1 + r2 + (1)$ in b) akumulator.

Povečevanje (angl. increment) in zmanjševanje (angl. decrement) sta aritmetični mikrooperaciji, pri katerih je eden izmed vhodov konstantna vrednost, najpogosteje kar vrednost 1:

$$r2 <= r1 + 1, \;\; r4 <= r3 - 1$$

Kadar prištevamo ali odštevamo vrednost k istemu signalu, dobimo števec. V razširjenem diagramu stanj naredimo števec tako, da v nekem stanju registru prištevamo konstantno vrednost, kot prikazuje slika 6.9.



Slika 6.9: Števec: a) v razširjenem diagramu stanj in b) shema vezja.

Množenje in deljenje ne spadata med osnovne mikrooperacije, ker jih lahko naredimo z zaporedjem osnovnih operacij, npr. množenje je narejeno kot zaporedno seštevanje in pomikanje vrednosti. Kadar potrebujemo v vezju hiter množilnik ali delilnik, ki ga razvijemo v obliki paralelnega kombinacijskega vezja. Ko se signali prenesejo do izhoda takšnega paralelnega kombinacijskega vezja, jih lahko ob fronti ure shranimo v register in in v tem primeru vezje predstavlja mikrooperacijo.

6.2.3 Logične mikrooperacije

Logične mikrooperacije so uporabne za manipulacijo s posameznimi biti v registru, saj se logična operacija izvaja nad vsakim bitom posebej. Poglejmo si osnovne tri mikrooperacije, invertiranje, logično AND in OR operacijo:

$$r2 <= \text{ NOT } r1$$

$$r3 <= r1 \text{ AND } r2$$

$$r3 <= r1 \text{ OR } r2$$

Vezje logičnih mikrooperacij vsebuje parallelno vezana logična vrata in register. Invertiranje je npr. uporabno pri izvedbi mikrooperacije odštevanja z uporabo dvojiškega komplementa. Logični mikrooperaciji AND in OR pa sta uporabni za maskiranje bitov. Če imamo npr. 8 bitno vrednost $r1$ in bi želeli dobiti rezultat pri katerem so zgornji 4 biti postavljeni na 0, spodnji pa enaki spodnjim bitom signala $r1$, naredimo operacijo:

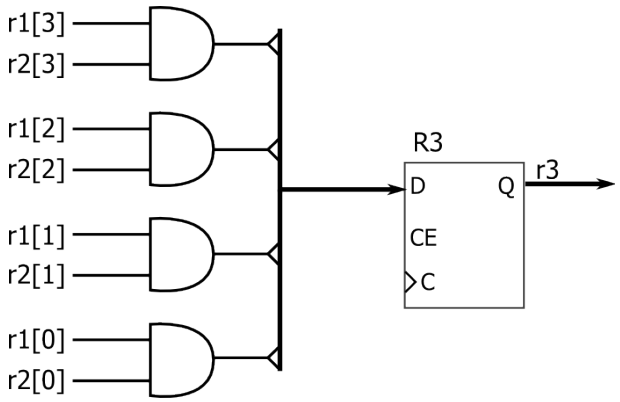
$$r3 <= r1 \text{ AND } 00001111$$

Operacija se izvrši nad istoležnimi biti:

$r1$ (podatek)	10100101
$r2$ (maska)	00001111
$r3$ (rezultat)	00000101

Kadar izvajamo maskiranje z operacijo AND se pobrišejo vsi biti, ki imajo v maski vrednost 0. Obratno je pri maskiranju z operacijo OR, kjer se vsi biti ki so v maski postavljeni na 1 tudi na rezultatu postavijo na 1.

Slika 6.10 prikazuje vezje za izvedbo 4-bitne logične mikrooperacije, ki vsebuje štiri paralelno vezana logična vrata AND.



Slika 6.10: Izvedba logične mikrooperacije AND.

6.2.4 Mikrooperacije pomika

Z mikrooperacijami pomika spreminjamo mesta posameznih bitov v registru. Osnovni mikrooperaciji sta pomik vseh bitov za eno mesto v desno in pomik vseh bitov za eno mesto v levo. Pomik za eno mesto v desno predstavlja deljenje številske vrednosti z 2, pomik v levo pa množenje vrednosti z 2.

Poglejmo si primer pomika 4 bitne vrednosti $r1$ za eno mesto v desno:

$r1$ (podatek)	1010
$r2$ (pomaknjena vrednost)	0101

Pomaknjena vrednost ima najvišji bit vedno postavljen na 0, nato pa sledijo zgornji trije biti vhodne vrednosti (biti z indeksi 3 do 1), najnižji bit pa pri pomikanju izpade. Enačba mikrooperacije za pomik 4-bitne vrednosti r1 za eno mesto v desno je:

$$r2 <= 0 \ : \ r1[3..1]$$

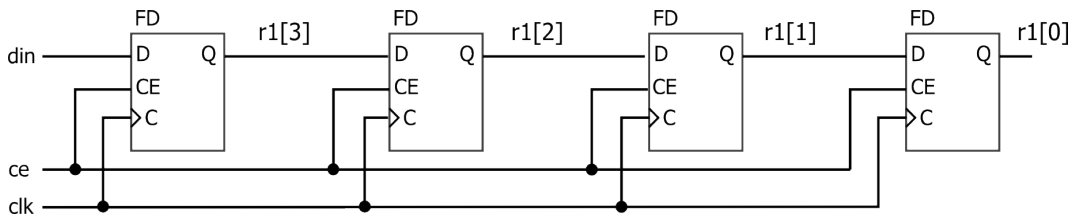
Rezultat je sestavljen iz dveh delov, ki smo ju v zapisu mikrooperacije ločili z dvopičjem; na levi strani je konstanta 0, na desni pa podvektor signala r1. Podobno velja pri pomiku za eno mesto v levo, le da tokrat izpade najvišji bit in dodajamo ničlo na desni strani:

$$r2 <= r1[2..0] \ : \ 0$$

Vezje za osnovni mikrooperaciji pomika je zelo preprosto, saj vsebuje le register, ki ima na vhodu ustrezno povezane bite in konstanto 0. Vezje za pomikanje pri katerem je izvor in cilj isti register se imenuje pomikalni register. Pri pomikalnem registru moramo poskrbeti, da v njega na nek način naložimo podatek. Glede na način nalaganja ločimo paralelne in serijske pomikalne registre.

Slika 6.11 prikazuje 4-bitni serijski pomikalni register, ki izvaja naslednjo operacijo:

$$r1 <= din \ : \ r1[3..1] \mid ce = 1$$



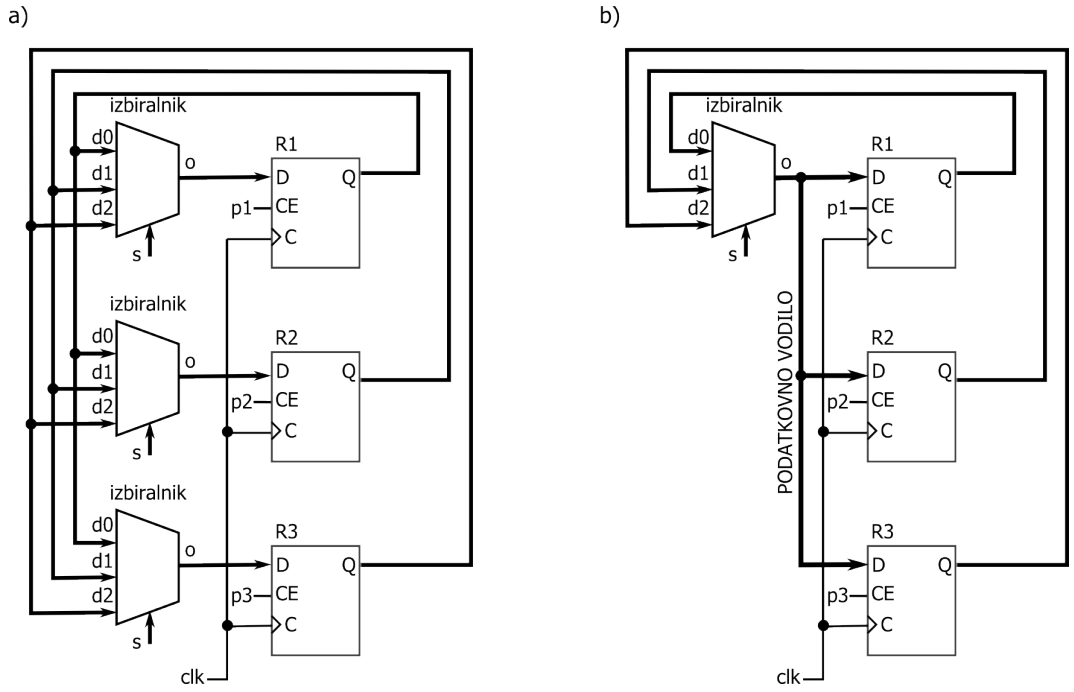
Slika 6.11: Serijski pomikalni register.

6.2.5 Registrski prenos s skupnim vodilom

V digitalnem sistemu imamo običajno veliko registrov med katerimi prenašamo podatke. Povezave med različnimi pari registrov lahko naredimo z izbirnim prenosom, ki ga v vezju predstavlja izbiralnik pred vsakim registrom. Takšno vezje ima veliko logičnih elementov in medsebojnih povezav.

Bolj učinkovito rešitev prenosa podatkov med več registri predstavlja skupno **podatkovno vodilo**. Vodilo je podatkovna prenosna pot za različne mikrooperacije. S kontrolnimi signali izberemo en izvor in enega ali več ciljnih registrov, v katere se prenese podatek. Slika 6.12a prikazuje prenos podatkov med tremi različnimi registri: $R1$, $R2$ in $R3$. S signali s izberemo

register, ki je izvor podatkov, signali $p1$, $p2$ in $p3$ pa so kontrolni vhodi za nalaganje podatka v ciljni register.



Slika 6.12: Registrski prenos a) z izbiralniki in b) s skupnim podatkovnim vodilom.

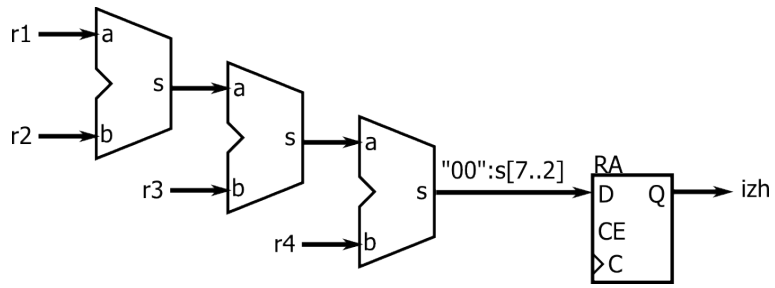
Slika 6.12b prikazuje shemo za prenos podatkov med tremi registri s skupnim vodilom, pri kateri potrebujemo le en izbiralnik.

6.3 Podatkovni podsistem

Obdelavo podatkov v digitalnem sistemu izvaja podatkovni podsistem v katerem so registri in kombinacijska logika (registrske mikrooperacije). Vzemimo primer vezja za izračun povprečja štirih vrednosti, ki ga lahko najdemo v aplikaciji za digitalno obdelavo video slike. VPodatkovni del vezja naredimo iz treh seštevalnikov in registra, kot prikazuje slika 6.13.

Če imamo hkrati na voljo vse štiri vrednosti vhodnih podatkov (signali $r1$, $r2$, $r3$ in $r4$), dobimo rezultat v enem urinem ciklu. Tudi v primeru, ko potrebujemo več ciklov, da na vhode pripeljemo vse vrednosti, je vezje hitrejše od mikroprocesorja, saj ne potrebujemo ciklov za shranjevanje vmesnih rezultatov, pa tudi pomikanje je narejeno le z ustreznimi povezavami. Takšno vezje lahko pri manjši frekvenci ure obdela hiter tok video podatkov v realnem času.

Slabost porazdeljene obdelave podatkov je, da potrebujemo več računskih enot in s tem večje vezje, ter da moramo za spremembo algoritma razviti novo vezje. Kadar uporabljamo programirljive digitalne sisteme z vezji CPLD ali FPGA, je tudi takšna rešitev dovolj prilagodljiva, ker omogoča poljubno spreminjanje in nalaganje novega vezja. Glavna slabost je nekoliko višja



Slika 6.13: Izračun povprečja štirih vrednosti s porazdeljeno obdelavo podatkov.

cena v primerjavi z mikroprocesorji in omejitev velikosti porazdeljenega vezja, ki pa jo nekoliko kompenzira hiter razvoj tehnologije programirljivih vezij.