7 Računske operacije

7.1 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:

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

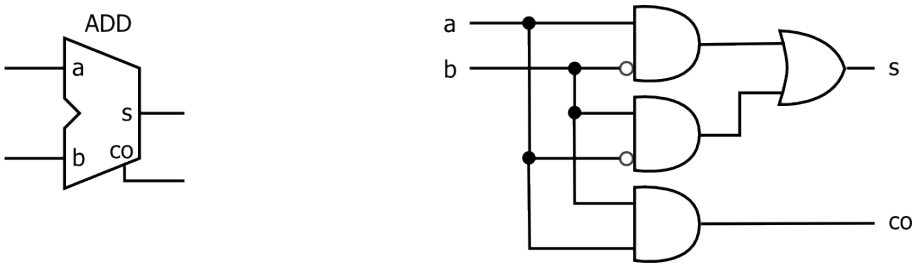
prenos

Pri zadnji kombinaciji smo dobili prenos, ki ga upoštevamo kot dodatno številko. Logično vezje za izračun vsote se imenuje seštevalnik. Slika 7.1 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.

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

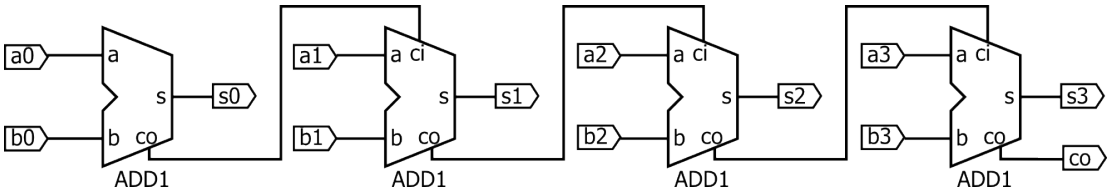
$$\begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 1\,0\,0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,1\,0\,0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

Vezje 4-bitnega seštevalnika naredimo s povezavo enobitnih seštevalnikov z vhodnim in izhodnim prenosom. Seštevalniki ADD1 na sliki 7.2 izračuna vsoto operandov a, b in vhodnega



Slika 7.1: Simbol in logična shema enobitnega seštevalnika.

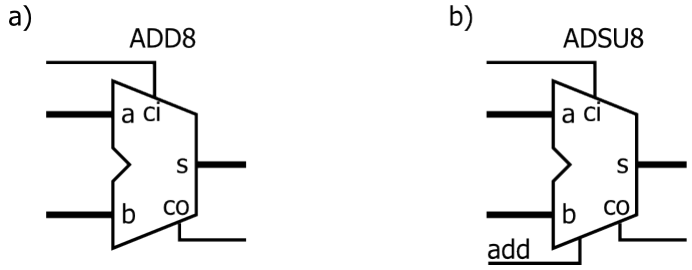
prenosa ci. Vsak izmed seštevalnikov izračuna en bit vsote. Po seštevanju nižjih bitov se izhodni prenos upošteva kot vhod v naslednji seštevalnik. Vsoto sestavljajo biti od $s0$ do $s3$ in izhodni prenos co.



Slika 7.2: Shema 4-bitnega seštevalnika.

Seštevalniki so pogosto uporabljeni elementi, zato dobimo v knjižnici osnovnih gradnikov že pripravljene večbitne seštevalnike. Slika 7.3 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 7.3 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 7.3: Simboli: a) 8-bitni seštevalnik in b) 8-bitni seštevalnik/odštevalnik.

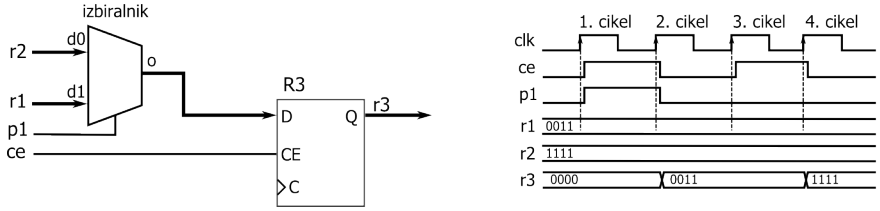
7.2 Mikrooperacije

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.

7.2.1 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 7.4.



Slika 7.4: Vezje za registrski prenos z enim izbirnim signalom in časovni diagram.

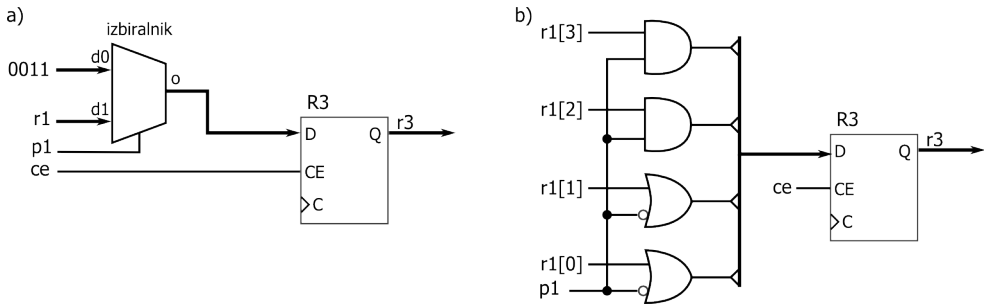
V ciljni register $R3$ se ob aktivnem signalu $ce = 1$ shrani podatek iz vodila $r1$, kadar je $p1 = 1$, sicer pa iz vodila $r2$. Na časovnem diagramu vidimo, da se dejanski prenos podatka izvrši vedno en cikel za tem, ko sta postavljena kontrolna signala ce in $p1$. Mikrooperacijo za izbirni prenos 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 7.5 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 7.5b.

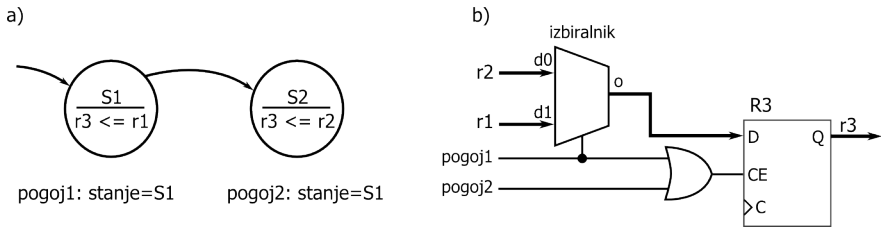
Kadar je $p1 = 0$ bo na izhodu logičnih vrat AND2 vrednost 0, ker velja: $r1[n]$ AND $0 = 0$, na izhodu vrat OR2B pa bo 1, zaradi: $r1[n]$ OR NOT $(0) = 1$. V primeru $p1 = 1$ pa lahko



Slika 7.5: Vezje za izbirni prenos s konstanto: a) z izbiralnikom in b) z logičnimi vrati.

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 7.6a, 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 7.6: 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 7.6b.

7.2.2 Aritmetične mikrooperacije

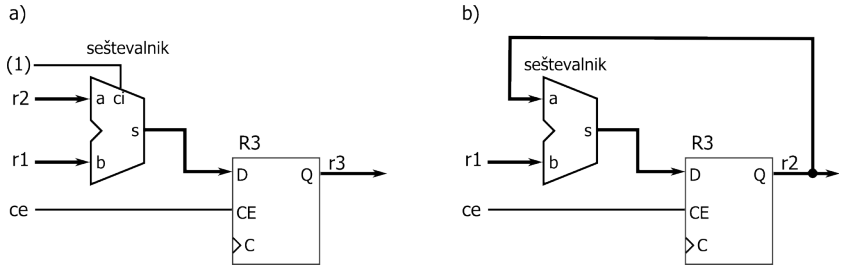
Kadar podatki v registru predstavljajo številsko vrednost, lahko z njimi naredimo računske ali aritmetične mikrooperacije. Osnovne aritmetične mikrooperacije so seštevanje, odštevanje, povečevanje in zmanjševanje vrednosti. Operacijo seštevanja vrednosti iz dveh registrov, ki se shranita v tretji register 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 7.7a. Mikrooperacija v obliki:

$$r2 <= r2 + r1$$

predstavlja akumulator, ki je vezje s povratno zanko, pri katerem enemu registru prištevamo vsebino drugega, kot prikazuje slika 7.7b. 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 $r1$.



Slika 7.7: Vezje za mikrooperacijo seštevanja: a) vsota $r1 + r2 + (1)$ in b) akumulator.

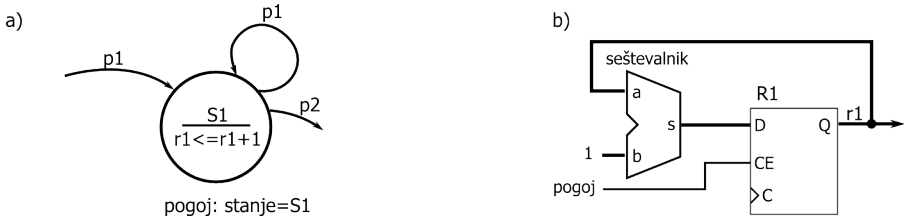
Odštevanje binarnih vrednosti naredimo z uporabo dvojiškega komplementa. Vrednost, ki jo želimo odšteti invertiramo in ji prištejemo 1 (vhodni prenos ci), nato pa kar seštejemo z drugo vrednostjo:

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

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

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

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 7.8.



Slika 7.8: 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.

7.2.3 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:

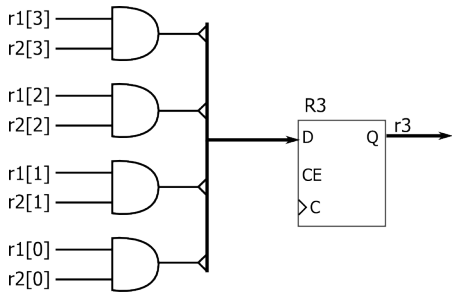
$$r2 <= \text{ NOT } r1$$

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

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

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:

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



Slika 7.9: Izvedba logične mikrooperacije AND.

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

7.2.4 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:

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

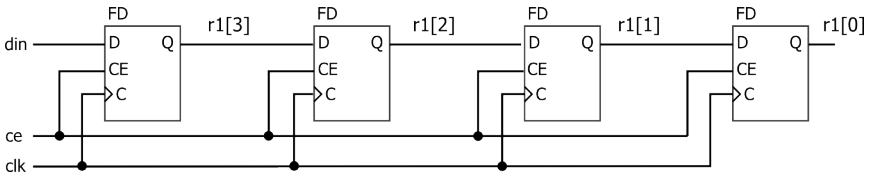
Rezultat je sestavljen iz dveh delov, ki smo ju v zapisu mikrooperacije ločili z dvopičjem; na levi strani je konstanta 0, na desni pa podvektor signala r1. Podobno velja pri pomiku za eno mesto v levo, le da tokrat izpade najvišji bit in dodajamo ničlo na desni strani:

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

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

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

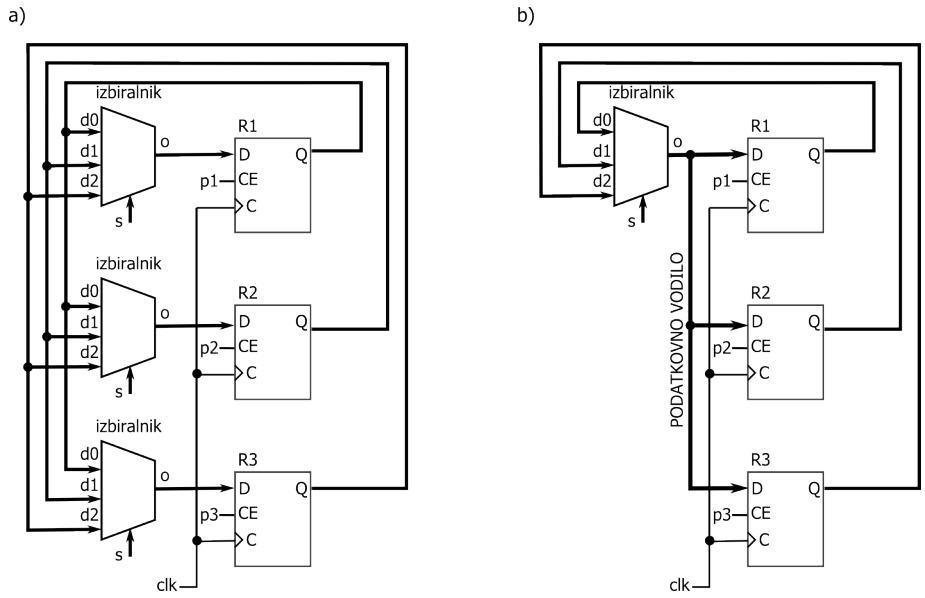


Slika 7.10: Serijski pomikalni register.

7.2.5 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 7.11a 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 7.11: Registrski prenos a) z izbiralniki in b) s skupnim podatkovnim vodilom.

Slika 7.11b prikazuje shemo za prenos podatkov med tremi registri s skupnim vodilom, pri kateri potrebujemo le en izbiralnik.

7.3 Načrtovanje registrskih celic

Na nivoju registrskih prenosov lahko naredimo univerzalne gradnike, ki jim s parametri določimo katero registrsko operacijo naj izvedejo. Osnovni element teh gradnikov so **registrske celice** sestavljene iz kombinacijske logike in flip-flopa D. Če med seboj povežemo n registrskih celic, dobimo n-bitni register z logiko za izvedbo ene ali več mikrooperacij.

Primer 7.1
Registrska celica za izvedbo dveh logičnih mikrooperacij:

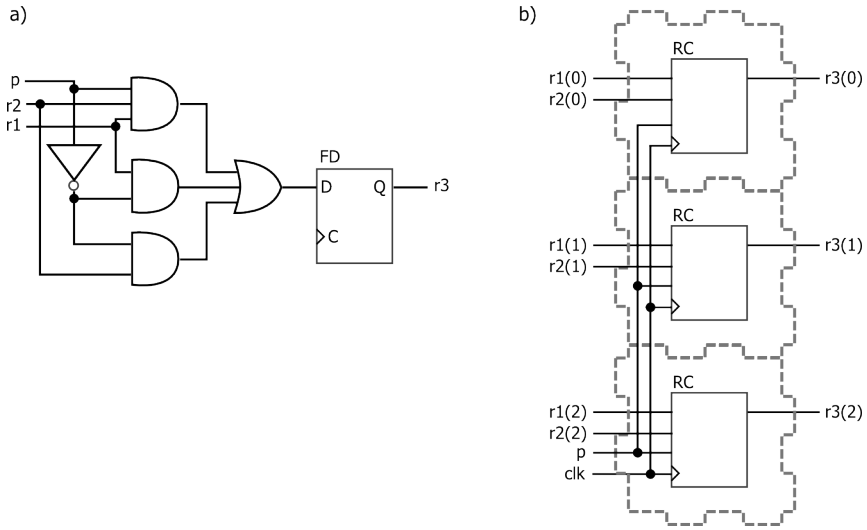
$$\begin{array}{ll} r3 <= r1 \text{ AND } r2 & \mid p = 1 \\ r3 <= r1 \text{ OR } r2 & \mid p = 0 \end{array}$$

Parameter p določa vrsto mikrooperacije; kadar je p=1 se izvede logična AND, pri $p = 0$ pa OR. Za načrtovanje registrske celice privzemimo, da so $r1$, $r2$ in $r3$ enobitni signali. Signal $r3$ ima vrednost 1 v primeru ko so $r1$ in $r2$ in p enaki 1 ali pa ko sta $r1$ ali $r2$ enaka 1 pri $p = 0$, kar zapišemo z enačbo:

$$r3 <= (r1 \text{ AND } r2 \text{ AND } p) \text{ OR } ((r1 \text{ OR } r2) \text{ AND NOT}(p))$$

Enačbo nekoliko preuredimo, da bo primerna za AND - OR izvedbo, kot prikazuje slika 7.12 Z **vzporedno** vezavo registrskih celic dobimo univerzalni gradnik za dve logični mikrooperaciji.

$$r3 <= (r1 \text{ AND } r2 \text{ AND } p) \text{ OR } (r1 \text{ AND NOT}(p)) \text{ OR } (r2 \text{ AND NOT}(p))$$



Slika 7.12: Shema a) registrske celice in b) gradnika za izvedbo operacij AND ali OR.

Primer 7.2
Registrska celica za izvedbo aritmetičnih mikrooperacij z dvema parametroma:

$$\begin{array}{ll} r3 <= r1 & \mid p1 = 0,\ p0 = 0 \text{ (prenesi } r1) \\ r3 <= r1 + r2 & \mid p1 = 0,\ p0 = 1 \text{ (seštej)} \\ r3 <= r1 + (\text{ NOT } r2) & \mid p1 = 1,\ p0 = 0 \\ r3 <= r1 - 1 & \mid p1 = 1,\ p0 = 1 \text{ (zmanjšaj)} \end{array}$$

Osnova aritmetične registrske celice je seštevalnik z dodatno logiko na vhodu. Seštevalnik ima tudi vhodni prenos ci, s katerim lahko naredimo dodatne operacije:

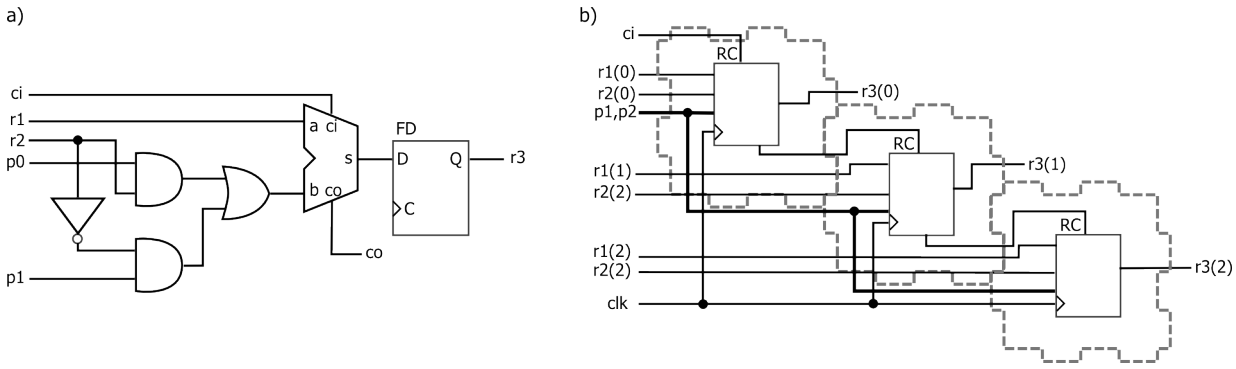
$$\begin{array}{ll} r3 <= r1 + ci & \mid p1 = 0,\ p0 = 0,\ ci = 1 \text{ (povečaj)} \\ r3 <= r1 + r2 + ci & \mid p1 = 0,\ p0 = 1,\ ci = 1 \text{ (seštej s prenosom)} \\ r3 <= r1 + (\text{ NOT } r2) + ci & \mid p1 = 1,\ p0 = 0,\ ci = 1 \text{ (odštej)} \\ r3 <= r1 - 1 + ci & \mid p1 = 1,\ p0 = 1,\ ci = 1 \text{ (prenesi } r1) \end{array}$$

Logično funkcijo za registrsko celico zapišemo v obliki:

$$r3 <= r1 + y + ci$$

$$\begin{array}{ll} y = 0 & \mid p1 = 0,\ p0 = 0 \\ y = r2 & \mid p1 = 0,\ p0 = 1 \\ y = (\text{ NOT } r2) & \mid p1 = 1,\ p0 = 0 \\ y = 1 & \mid p1 = 1,\ p0 = 1 \end{array}$$

Uvedli smo nov signal y, ki predstavlja kombinacijsko logiko z izhodom odvisnim od parametrov.
Z razvijanjem logičnih enačb pridemo do sheme registrske celice, ki jo vidimo na sliki 7.13a. Aritmetične celice povezujemo v večbitno enoto na podoben način kot gradimo večbitni seštevalnik; izhodne prenose (co) nižjih bitov **zaporedno** povežemo z vhodnimi prenosi (ci) višjih bitov, kot vidimo na primeru 7.13b.

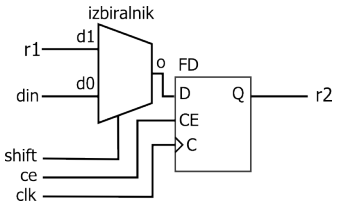


Slika 7.13: Shema a) registrske celice in b) gradnika za izvedbo aritmetičnih operacij.

Primer 7.3
Registrska celica za izvedbo mikrooperacije pomika in prenosa:

$$\begin{array}{ll} r2 <= din \ : \ r1[3..1] & \mid shift = 0 \text{ and } ce = 1 \\ r2 <= r1 & \mid shift = 1 \text{ and } ce = 1 \end{array}$$

Signal $shift$ določa, ali naj se ob pogoju za nalaganje ($ce = 1$) v register $R2$ naloži pomaknjena vrednost ali pa vrednost iz registra $R1$. Registrska celica je sestavljena iz izbiralnika in flip-flopa D, kot prikazuje slika 7.14.



Slika 7.14: Shema registrske celice izvedbo pomika in prenosa.