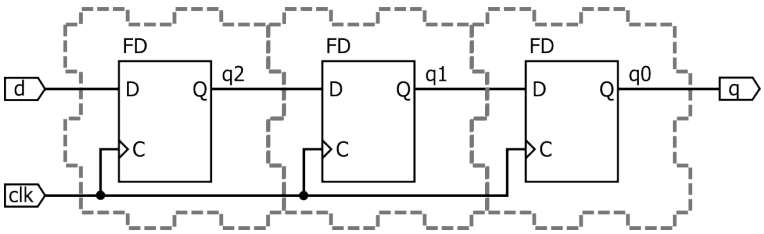
4
Načrtovanje vezij

Načrtovanje vezja ali *sinteza* vezja je postopek, pri katerem iz opisa delovanja pridemo do zgradbe vezja. Digitalno vezje naredimo s povezavo osnovnih kombinacijskih in sekvenčnih gradnikov. Vezje, ki vsebuje vsaj en sekvenčni gradnik, imenujemo sekvenčno vezje. Spoznali bomo postopek načrtovanja z gradniki, opis in načrtovanje z diagramom stanj in načrtovanje vezij z registri.

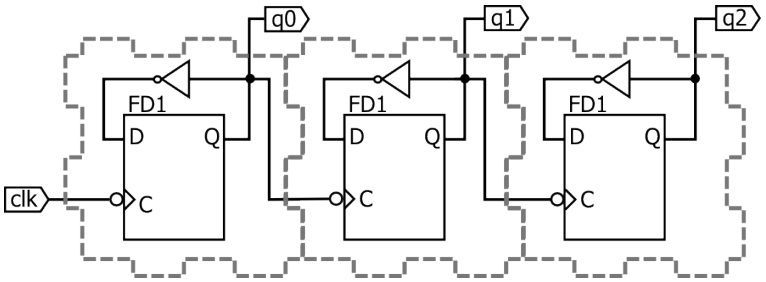
4.1 Načrtovanje z gradniki

Povezovanje digitalnih gradnikov lahko primerjamo z zlaganjem kock. Simboli digitalnih gradnikov imajo vhodne signale na levi in izhodne na desni strani. *Zaporedno vezavo* naredimo tako, da postavimo več gradnikov v vrsto in izhodne signale povežemo z vhodnimi:



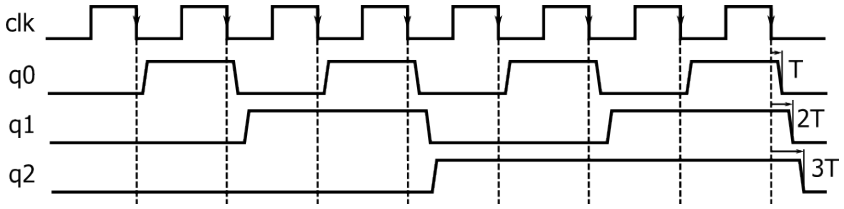
Slika 4.1: Zaporedna vezava podatkovnih flip-flopov.

Značilnost zaporedne vezave je zakasnitev pri prenosu podatkov skozi vezje. Kontrolni signali so vezani na vse gradnike (npr. ura), podatki pa prehajajo zaporedno čez gradnike in so zakasnjeni. Pomikalni register s slike 4.1 ima zakasnitev med vhodom in izhodom 3 urne cikle.



Slika 4.2: Zgradba 3-bitnega serijskega števca.

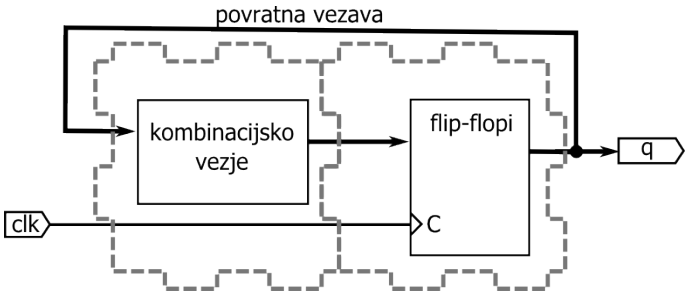
Iz zaporedno vezanih preklopnih flip-flopov naredimo serijski števec, kot prikazuje slika 4.2. Gradniki so vezani tako, da izhod posameznega flip-flopa krmili uro naslednjega. Na časovnem diagramu vidimo, da predstavlja zaporedje izhodnih signalov $q2$, $q1$ in $q0$ binarno zaporedje, ki se spreminja ob ciklih vhodne ure. V realnem vezju je izhod prvega flip-flopa $q0$ za čas T zakasnjen za fronto ure. Ta signal krmili naslednji flip-flop, katerega izhod ima že dvojno zakasnitev (2T) glede na vhodno uro, izhod tretjega pa trojno zakastnitev (3T).



Slika 4.3: Časovni potek signalov 3-bitnega serijskega števca.

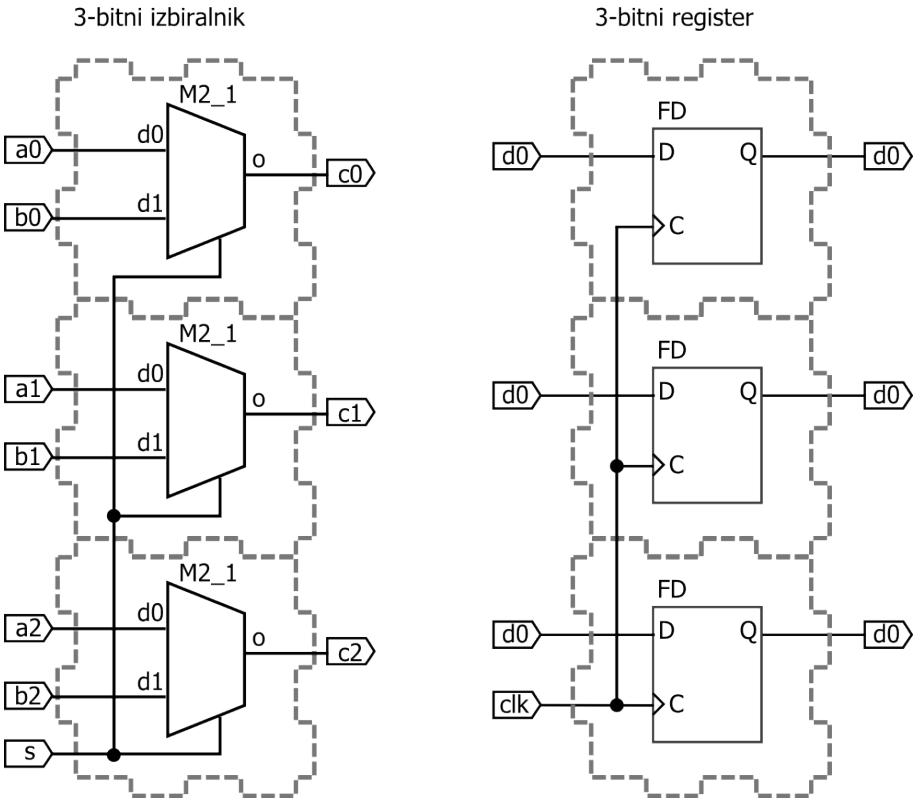
Števec pogosto uporabljamo kot generator zaporedja binarnih vrednosti. Če opazujemo vse izhode hkrati, nam različne zakasnitve preklopov povzročajo težave pri dekodiranju vrednosti in omejujejo navišjo frekvenco delovanja števca. Zaradi tega števce raje načrtujemo v obliki sinhronega vezja, kjer so vsi flip-flopi vezani na isto uro.

Povratna vezava pripelje podatkovni izhod preko enega ali več gradnikov na vhod istega gradnika. Na sliki 4.4 je povratna vezava med izhodom flip-flopov in kombinacijsko logiko pred vhodom flip-flopov, ki se uporablja za ciklična sekvenčna vezja (npr. števci).



Slika 4.4: Shema sekvenčnega vezja s povratno vezavo.

Vzporedno vezavo uporabljamo za izdelavo večbitnih struktur iz osnovnih gradnikov ali pa za hitrejša vezja. Tudi pri vzporedni vezavi so nekateri kontrolni signali gradnikov vezani skupaj, podatkovni vhodi in izhodi pa med seboj niso povezani.

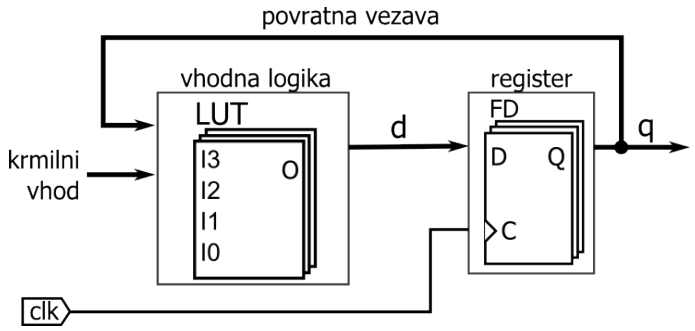


Slika 4.5: Izvedba 3-bitnega izbiralnika in registra z vzporedno vezavo.

Primer vzporedne vezave je na sliki 4.5. Značilnost vzporedne vezave je, da ne poveča zakasnitev signalov v primerjavi s posameznim gradnikom. V splošnem pričakujemo, da imajo večja vezja večjo zakasnitev, ker morajo signali potovati med zaporednimi gradniki ali logičnimi vrati. Vsak element prispeva nekaj zakasnitve, ki se sešteva na poti med signalnimi vhodi in izhodi vezja. Pri vzporedni vezavi pa se zakasnitve ne seštevajo, ker so podatkovni signali na posameznih gradnikih neodvisni od drugih.

4.1.1 Sinhroni števec

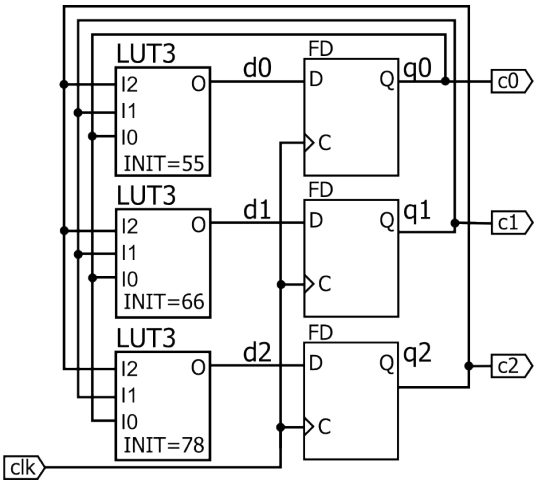
Števci spadajo med sekvenčna vezja, ki ciklično spreminjajo stanja na izhodu. Sinhroni števci so narejeni kot digitalna vezja s povratno vezavo, na kateri mora biti vsaj en sekvenčni gradnik (register ali flip-flop).



Slika 4.6: Izvedba sinhronega sekvenčnega vezja s povratno vezavo.

Za izvedbo 3-bitnega sinhronega števca potrebujemo 3 podatkovne flip flope in 3-vhodno kombinacijsko vezje iz logičnih vrat ali tabel LUT:

q2	q1	q0	d2	d1	d0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0



Slika 4.7: Tabela in zgradba 3-bitnega sinhronega števca.

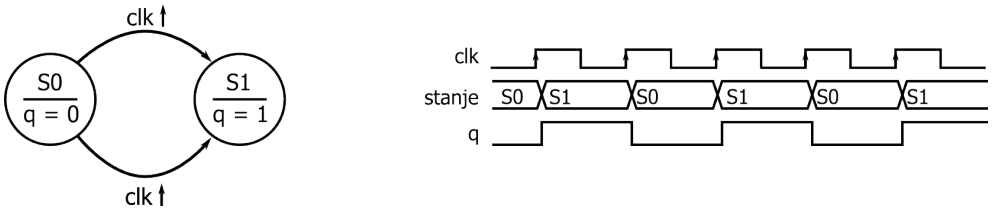
V posamezne vpogledne tabele LUT vpišemo vrednost izhodnega stolpca pravilnostne tabele. Npr. v tabelo signala $d2$ vpišemo INIT=78, kar predstavlja šestnajstiško vrednost kombinacije 01111000, ki jo preberemo od spodaj navzgor.

Vhodno kombinacijsko vezje opišemo s pravilnostno tabelo, ki določa vrednosti na vhodu flip-flopov (di) ob vseh kombinacijah stanja in vhodov sekvenčnega vezja.

4.2 Načrtovanje z diagramom stanj

Sekvenčna vezja s povratno vezavo predstavimo z *diagramom stanj*. Diagram stanj je grafičen opis delovanja vezja, v katerem so prikazana stanja vezja in pogoji za prehod med stanji. Stanje je kombinacija vrednosti, ki so shranjene v flip-flopih. Stanja prikazujemo s krožnicami v katerih je simbolično ime stanja, npr. $S0$, $S1$..., prehode med stanji pa s puščicami. Na puščice zapišemo pogoje za prehod v katerih nastopajo vhodi v sekvenčno vezje. V vsakem stanju zapišemo vrednost izhodnih signalov pod imenom stanja.

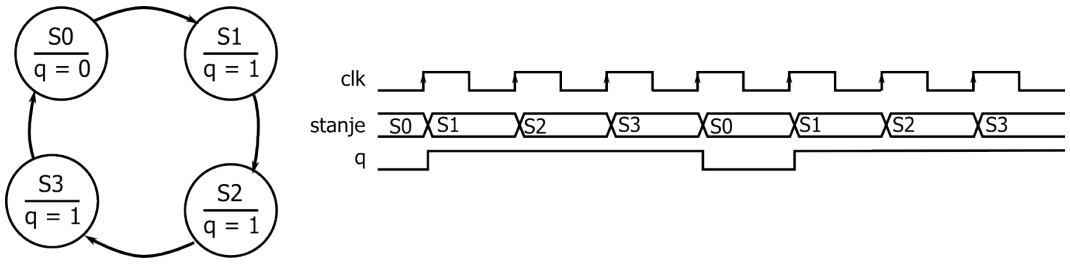
Na sliki 4.8 je preprost diagram z dvema stanjema: S0 in S1. V stanju S0 je izhod q enak 0. Ob prednji fronti ure se izvrši prehod v stanje S1, v katerem se postavi q na vrednost 1. Ob naslednji prednji fronti gremo spet nazaj v S0.



Slika 4.8: Diagram stanj in časovni potek na simulaciji.

S sledenjem prehodov v diagramu stanj razberemo časovni potek signalov v sekvenčnem vezju. Na izhodu q dobimo periodičen signal, ki je dvakrat počasnejši od vhodne ure. Prehod med stanji sinhronega vezja je vedno pogojen s fronto ure, zato pri risanju diagrama izpustimo pogoj za fronto ure pri prehodih stanj. Če je puščica brez oznake, pomeni da se prehod v naslednje stanje izvrši ob nasledji fronti ure.

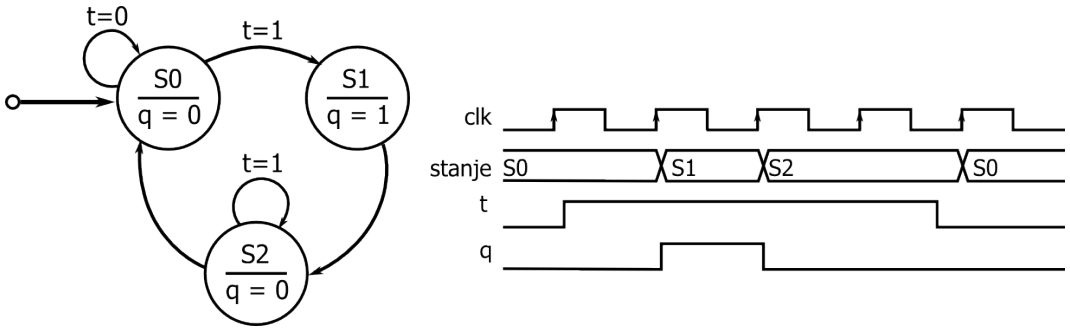
Narišimo diagram stanj vezja, ki generira periodičen izhod, tako da je en urni cikel na 0 in tri cikle na vrednosti 1. Vezje vsebuje štiri stanja, ki se ciklično izmenjujejo, kot prikazuje slika 4.9.



Slika 4.9: Diagram stanj in časovni diagram za 3 cikle dolge impulze na izhodu.

4.2.1 Generator impulzov

Naredimo diagram vezja za generiranje kratkega impulza ob spremembi vhodne vrednosti na 1. Diagram vsebuje tri stanja: $S0$, $S1$ in $S2$, kot prikazuje slika 4.10. Prvo stanje $S0$ je s puščico označeno kot začetno stanje v katerega se postavi vezje ob resetu. Če je vezje v prvem stanju in vhodni signal t na 0, ostaja v tem stanju, kar je označeno s polkrožno puščico. Ob spremembi vhoda t na 1 in fronti ure se stanje spremeni v $S1$ in izhod q gre na 1. Ob naslednji fronti ure gre v stanje $S2$, kjer je toliko časa, dokler se vhodni signal ne postavi na '0'.

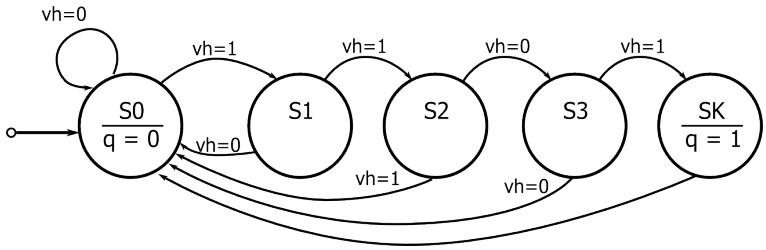


Slika 4.10: Diagram stanj in časovni diagram generatorja impulzov.

Generator kratkih impulzov je uporaben za oblikovanje signala, ki ga dobimo ob pritisku tipke. Z njim spremenimo poljubno dolg vhodni impulz v kratek izhodni impulz.

4.2.2 Detekcija zaporedja

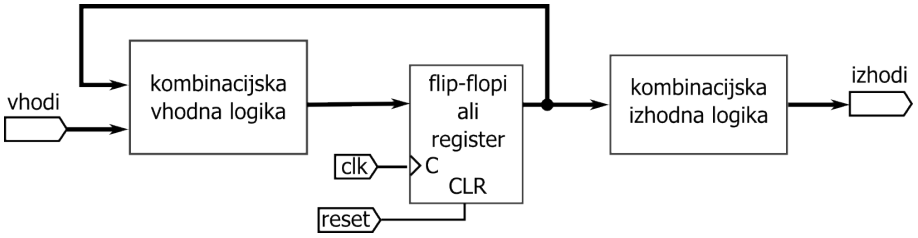
Diagram stanj uporabljamo za opis vezij, ki morajo generirati ali pa se odzivati v določenem zaporedju. Videli smo, da z diagramom lahko opišemo sekvenčno vezje, ki naredi na izhodu neko zaporedje stanj. Diagram na sliki 4.11 pa opravlja obratno nalogo: opazuje vhodni signal vh in ugotavlja ali se spreminja po določenem zaporedju. Če se vhod ob zaporednih frontah ure postavi na vrednosti 1, 1, 0 in 1, prehaja vezje skozi vsa stanja do končnega, kjer postavi izhod q na 1. V primeru kakršnekoli druge kombinacije se vrnemo v začetno stanje.



Slika 4.11: Diagram stanj za detekcijo zaporedja.

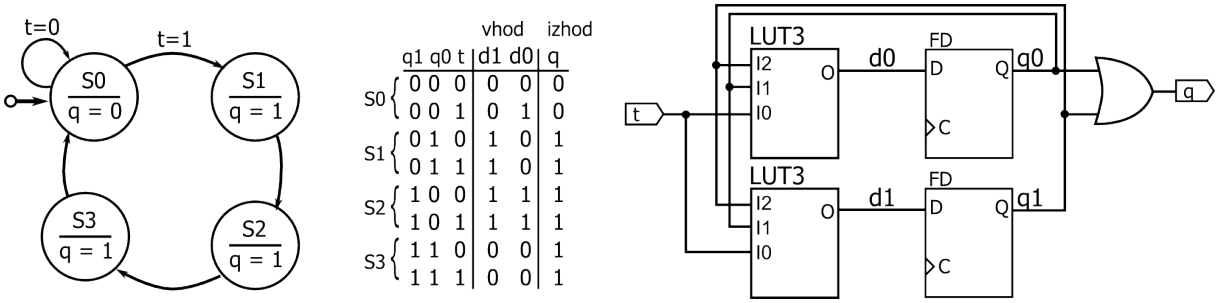
4.2.3 Sinteza sekvenčnega vezja

Na osnovi diagrama stanj naredimo vezje, ki ga imenujemo končni stroj stanj (angl. Finite State Machine) ali avtomat. Vezje je v splošnem sestavljeno iz: vhodne logike, flip-flopov ali registra in izhodne logike, kot prikazuje slika 4.12.



Slika 4.12: Blokovna shema končnega stroja stanj.

Naredimo sekvenčni stroj, ki generiran na izhodu impulz dolžine treh urinih ciklov, kadar dobi na vhod b vrednost 1. Na sliki 4.13 diagram stanj vezja, ki vsebuje štiri stanja. Pri izdelavi tabele za vhodno in izhodno logiko moramo stanjem prirediti binarne kombinacije.



Slika 4.13: Diagram stanj, tabela in izvedba s sekvenčnim vezjem.

Stanje $S0$ je predstavljeno s kombinacijo 00 na izhodu flip-flopov, stanje $S1$ je kombinacija 01. $S2$ je 10 in S3 je 11. Vezje ima še enobitni vhod t, ki da skupaj z dvobitnim stanjem $2^3 = 8$ kombinacij v pravilnostni tabeli. Na podlagi diagrama stanj določimo za vsako kombinacijo vrednosti na vhodu v flip-flope (naslednje stanje) in vrednosti izhoda vezja. Na sliki 4.13 je izvedba sekvenčnega vezja, pri katerem je vhodna logika narejena z dvema tabelama, izhod pa z logičnimi vrati OR.

Iz grafičnega opisa diagrama stanj je možno avtomatsko narediti sekvenčno vezje. Programska oprema za računalniško podprto načrtovanje najprej določi število flip-flopov na podlagi števila stanj in načina kodiranja stanj. Nato določi optimalno izvedbo kombinacijske vhodne logike in dekodirnik za izhodne signale.

4.3 Načrtovanje z registri

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. Mikrooperacije se izvedejo v enem ciklu na vseh podatkovnih bitih hkrati. Rezultat operacije se lahko shrani nazaj v register ali pa prenese v naslednji register. Spoznali bomo štiri vrste mikrooperacij: prenos, aritmetične, logične in mikrooperacije pomika.

4.3.1 Registrski prenos

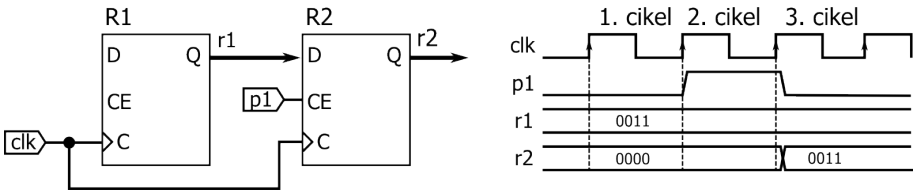
Vzemimo dva registra $R1$ in $R2$, ki imata izhodna signala $r1$ in $r2$ z enakim številom podatkovnih bitov. Prenos podatka iz enega v drug register zapišemo z izrazom:

$$r2 \leftarrow r1$$

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. Podatki se prenesejo ob fronti ure, na katero sta vezana oba registra. Če ni dodatnih pogojev se prenos izvede v vsakem urnem ciklu. Registrski prenos s pogojem zapišemo:

$$r2 \leftarrow r1 \mid pogoj$$

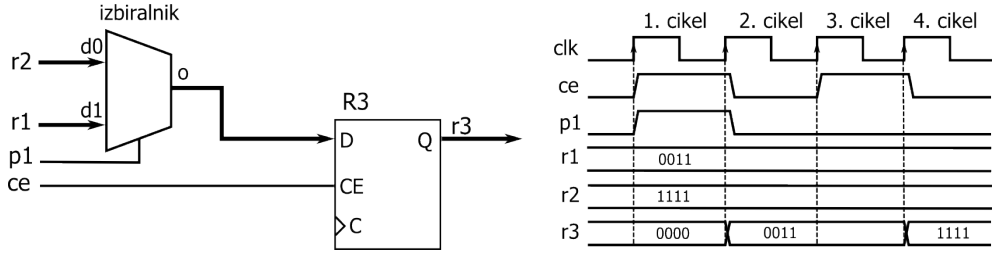
Pogoj je lahko vrednost binarnega kontrolnega signala ali poljuben logični izraz. Slika 4.14 prikazuje vezje za pogojni registrski prenos, ki prenese podatek v drugi register kadar je vrednost kontrolnega signala $p1$ enaka 1.



Slika 4.14: Vezje za pogojni prenos podatkov med dvema registroma.

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 4.14 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.

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 4.15.



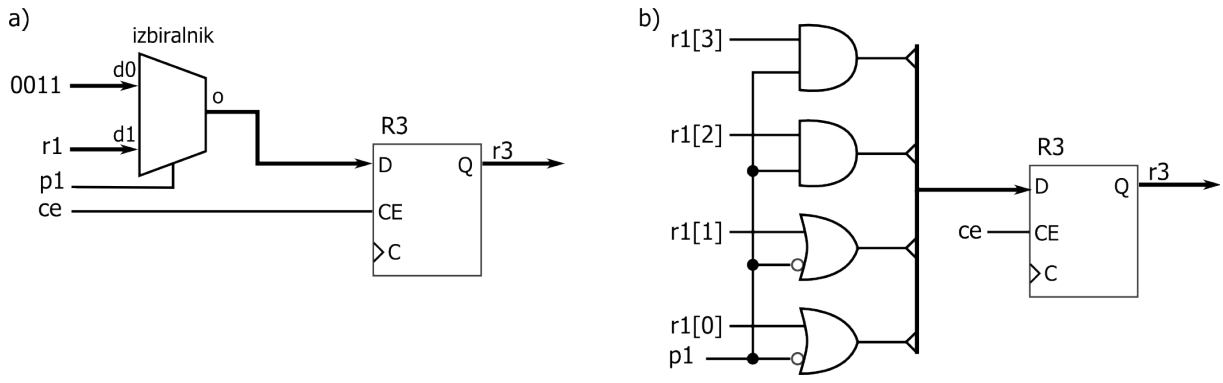
Slika 4.15: Vezje za registrski prenos z izbirnim signalom in časovni diagram.

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:

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

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

V primeru, ko je na enem izmed vhodov konstantna vrednost dobimo še nekoliko manjše vezje. Slika 4.16 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 4.16b.



Slika 4.16: 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 $r[n], n = 0..3$

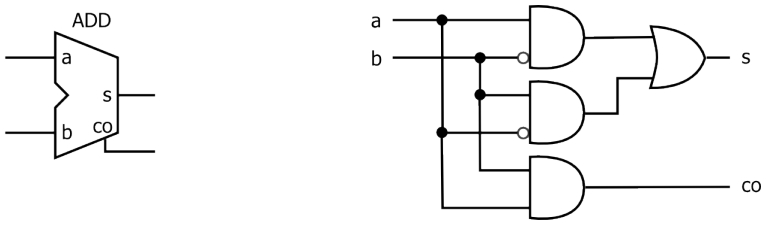
4.3.2 Aritmetične mikrooperacije

Osnovna aritmetična operacija je seštevanje. Naredimo izračun vsote 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 4.17 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).

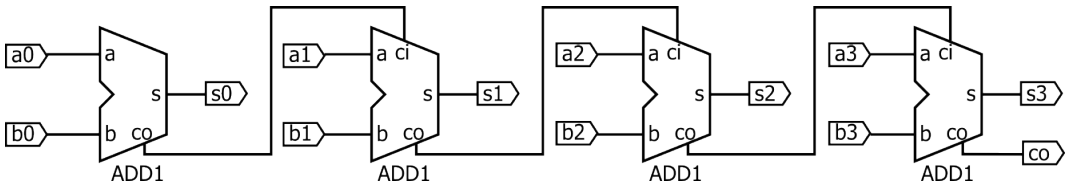


Slika 4.17: Simbol in logična shema enobitnega seštevalnika.

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:

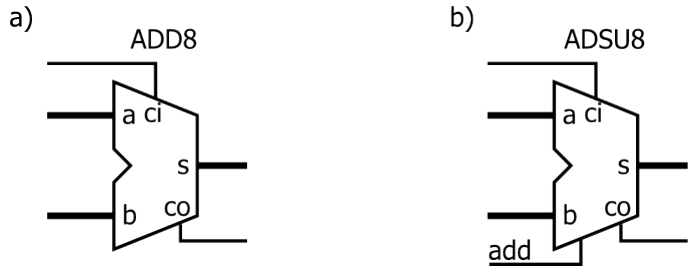
$$\begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 1\,0\,0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\,0\,0\,0\,1 \\ \hline 0\,1\,0\,0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

Vezje 4-bitnega seštevalnika naredimo s povezavo enobitnih seštevalnikov z vhodnim in izhodnim prenosom. Seštevalniki ADD1 na sliki 4.18 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 4.18: Shema 4-bitnega seštevalnika.

Seštevalniki so pogosto uporabljeni elementi, zato dobimo v knjižnici osnovnih gradnikov že pripravljene večbitne seštevalnike. Slika 4.19 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.

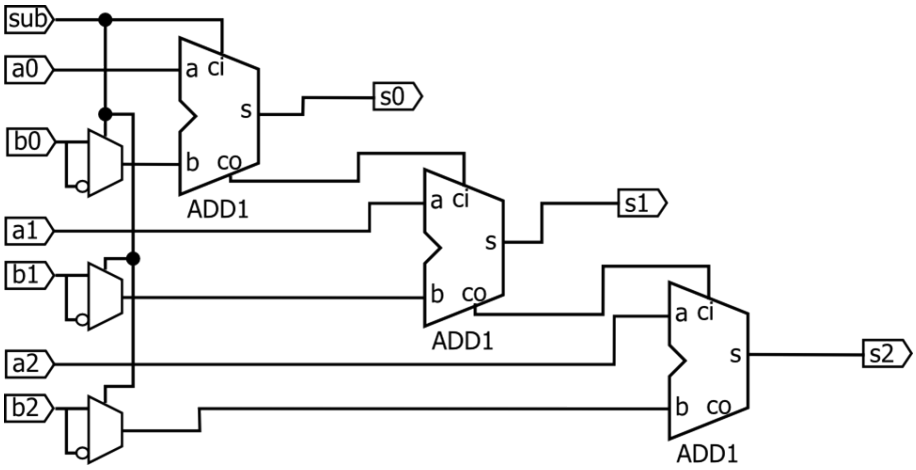


Slika 4.19: Simboli: a) 8-bitni seštevalnik in b) 8-bitni seštevalnik/odštevalnik.

Enak simbol se uporablja tudi za splošno aritmetično enoto, ki izvaja različne računske operacije nad večbitnimi signali. Slika 4.19 b) prikazuje simbol 8-bitne seštevalno/odštevalne enote ADSU8. Dodatni vhod add določa ali enota izvaja operacijo seštevanja ($add = 1$) ali pa odštevanja ($add = 0$). Odštevanje binarnih števil naredimo s pomočjo dvojiškega komplementa, tako da odštevanec negiramo in naredimo vsoto z vhodnim prenosom:

$$\begin{array}{r} 0\,0\,1\,1 \\ -\;0\,0\,0\,1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\;1\,1\,1\,0 \\ \hline 1 \\ 0 \end{array} \begin{array}{l} \text{negacija} \\ \text{prenos} \end{array} \qquad \begin{array}{r} 0\,0\,1\,1 \\ +\;1\,1\,1\,0 \\ \hline 1\,1\,1\,1 \\ 0\,0\,1\,0 \end{array}$$

Slika 4.20 prikazuje izvedbo 3-bitnega gradnika za izračun vsote ali razlike. Kadar je signal sub na 0, bo vezje izračunalo vsoto vhodov a in b. Kadar je sub enak 1, pa bodo prišle na vhod seštevalnikov negirane vrednosti b, na vhod prvega pa še prenos in rezultat bo razlika vrednosti.

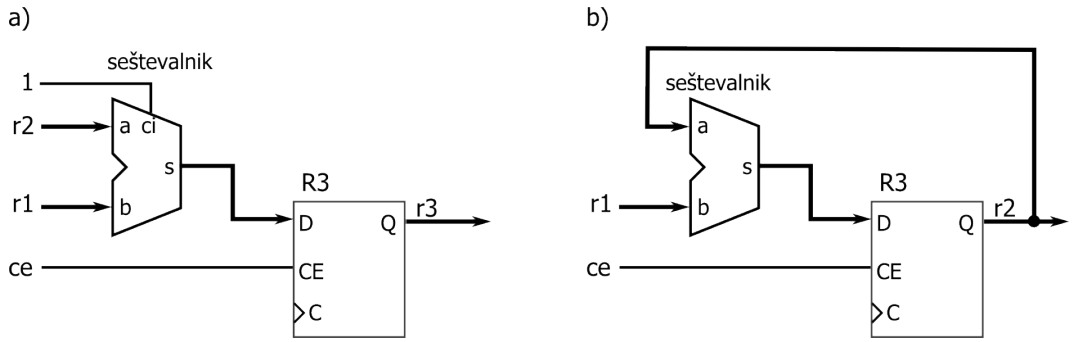


Slika 4.20: Izvedba 3-bitnega seštevalno-odštevalnega gradnika.

Aritmetično mikrooperacijo seštevanja vrednosti iz dveh registrov, ki se shranita v tretji register zapišemo:

$$r3 \leftarrow r1 + r2$$

Za izvedbo te operacije potrebujemo 3 registre in kombinacijsko vezje, ki izvaja operacijo seštevanja, kot je npr. paralelni seštevalnik, kot prikazuje slika 4.21a.



Slika 4.21: Vezje za mikrooperacijo seštevanja: a) vsota $r1 + r2 + 1$ in b) akumulator.

Mikrooperacija v obliki:

$$r2 \leftarrow r2 + r1$$

predstavlja akumulator, ki je vezje s povratno zanko, pri katerem enemu registru prištevamo vsebino drugega, kot prikazuje slika 4.21b. 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:

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

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

$$r2 \leftarrow r1 + 1, \quad r4 \leftarrow r3 - 1$$

Kadar prištevamo ali odštevamo vrednost k istemu signalu, dobimo dvojiški števec. 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.

4.3.3 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:

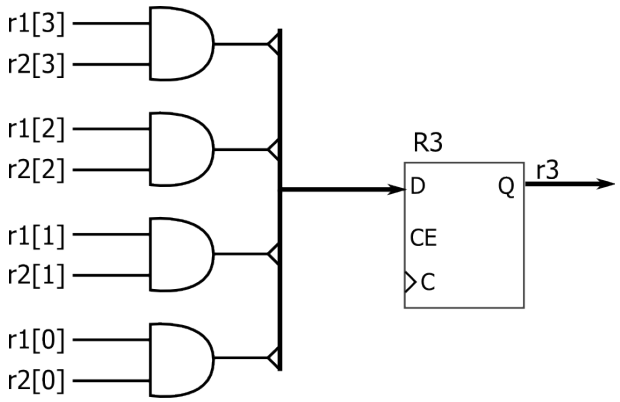
$$r2 \leftarrow \text{ NOT } r1$$

$$r3 \leftarrow r1 \text{ AND } r2$$

$$r3 \leftarrow r1 \text{ OR } r2$$

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:

$$r3 \leftarrow r1 \text{ AND } 00001111$$



Slika 4.22: 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 4.22 prikazuje vezje za izvedbo 4-bitne logične mikrooperacije, ki vsebuje štiri paralelno vezana logična vrata AND.

4.3.4 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:

$$r2 \leftarrow 0 \ : \ r1[3..1]$$

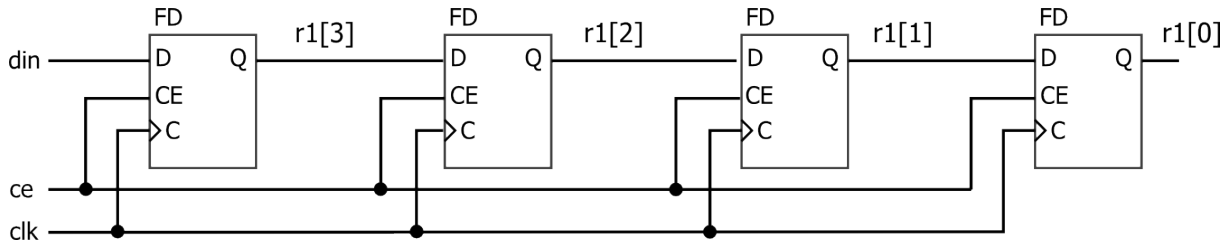
Rezultat je sestavljen iz dveh delov, ki smo ju v zapisu mikrooperacije ločili z dvopičjem; na levi strani je konstanta 0, na desni pa podvektor signala r1. Podobno velja pri pomiku za eno mesto v levo, le da tokrat izpade najvišji bit in dodajamo ničlo na desni strani:

$$r2 \leftarrow r1[2..0] \ : \ 0$$

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

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



Slika 4.23: Serijski pomikalni register.