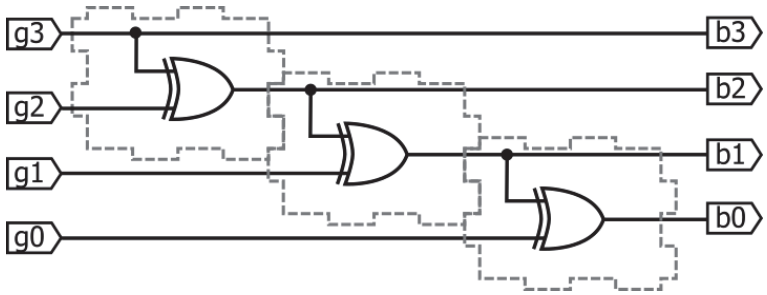
6
Načrtovanje vezij

Načrtovanje vezja ali *sinteza* vezja je postopek, pri katerem pridemo iz opisa delovanja 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 osnovne vezave logičnih gradnikov, izdelavo sinhronih števcev ter postopek načrtovanja sekvenčnih strojev z diagramom stanj.

6.1 Osnovne vezave gradnikov

Zaporedno vezavo logičnih gradnikov naredimo tako, da izhod enega gradnika vežemo na vhod drugega. Primer takšne vezave je dekodirnik 4-bitne Grayeve kode, v katerem so zaporedno vezana tri logična vrata XOR.



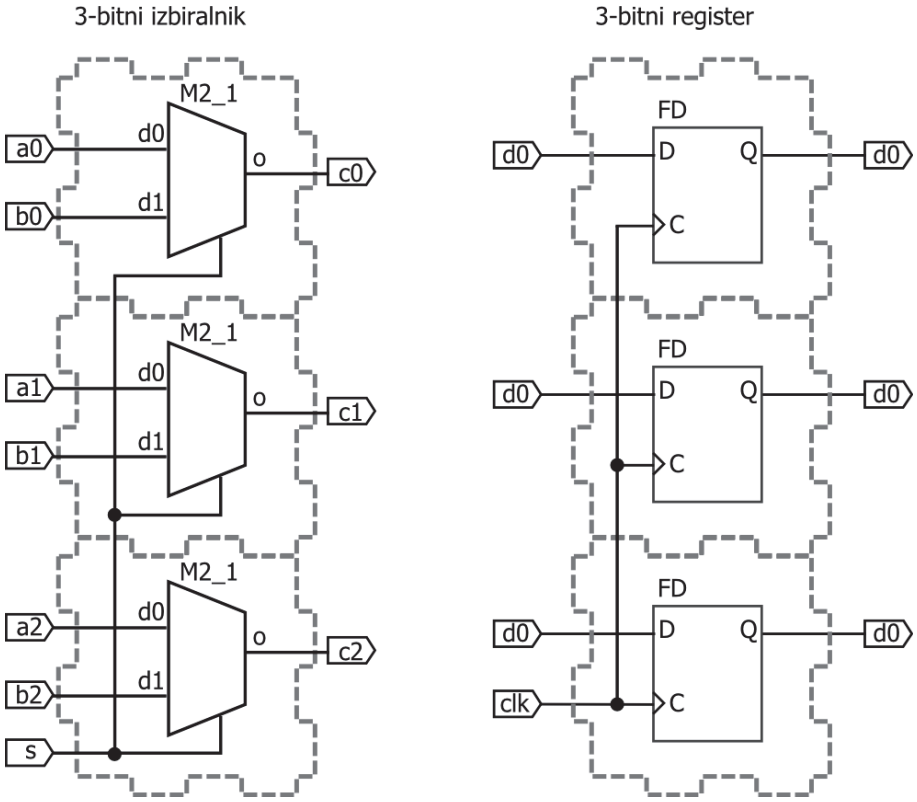
Slika 6.1: Zaporedna vezava logičnih vrat v dekodirniku Grayeve kode.

Število zaporedno vezanih kombinacijskih elementov določa število nivojev vezja (prikazani dekodirnik je 3-nivojsko vezje). Nekateri signali potujejo od vhoda čez vse elemente vezja in

imajo zato velike zakasnitve. Pri zaporedni vezavi se zakasnitve logičnih elementov seštevajo. Teoretično lahko poljubno kombinacijsko funkcijo naredimo z dvonivojskim vezjem, vendar bi marsikdaj potrebovali za izvedbo preveč logičnih elementov. Večnivojska izvedba je v nekaterih vezjih (npr. dekodirniki, seštevalniki) najboljši kompromis med velikostjo in zmogljivostjo vezja.

Z zaporedno vezavo sekvenčnih elementov vplivamo na število urnih ciklov med spremembo vhoda in spremembo na izhodu vezja. Najosnovnejši primer je zaporedna vezava flip-flopov v pomikalni register (slika 5.16). Število nivojev določa število zakasnitvenih ciklov, ne vpliva pa na najvišjo frekvenco ure. Ta je določena z zakasnitvijo znotraj flip-flopa in največjo zakasnitvijo v kombinacijskih elementih med flip-flopi.

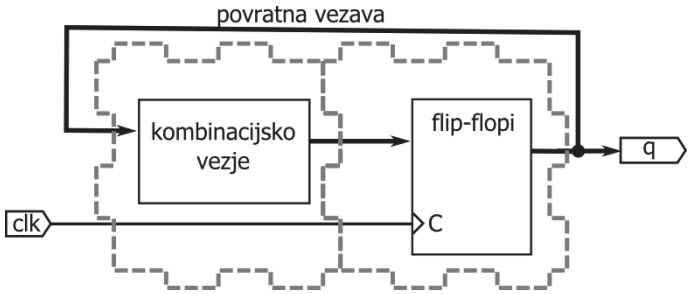
Vzporedno vezavo uporabljamo za izdelavo večbitnih struktur iz osnovnih gradnikov ali pa za hitrejša vezja. Tudi pri vzporedni vezavi so nekateri kontrolni signali gradnikov vezani skupaj, podatkovni vhodi in izhodi pa med seboj niso povezani.



Slika 6.2: Izvedba 3-bitnega izbiralnika in registra z vzporedno vezavo.

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.

Povratna vezava pripelje podatkovni izhod prek enega ali več gradnikov na vhod istega gradnika. Vezja iz logičnih vrat s povratno vezavo smo zasledili pri zgradbi osnovnih sekvenčnih gradnikov. Povratna vezava signalov povzroči ohranjanje notranjega stanja vezja, lahko pa ima tudi neželene učinke, npr. nestabilnost RS-zapaha. Kadar potrebujemo pomnilne elemente, uporabimo že narejene sekvenčne gradnike, zato v praksi nikoli ne povezujemo izključno kombinacijskih elementov v povratni vezavi.

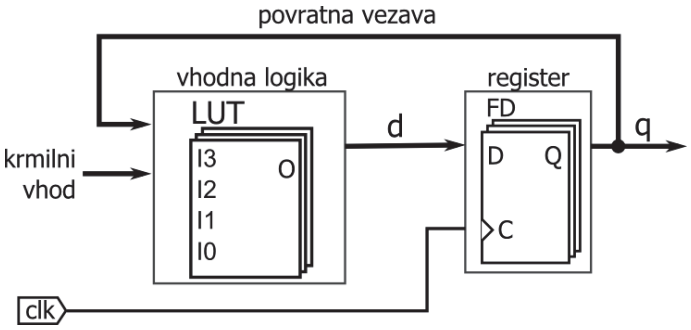


Slika 6.3: Shema sekvenčnega vezja s povratno vezavo.

Osnovno sekvenčno vezje s povratno vezavo je sestavljeno iz kombinacijske logike in flip-flopov, tako da so izhodi flip-flopov prek logike vezani na vhode. Takšna vezava se uporablja za ciklična sekvenčna vezja, kot so števci in sinhroni stroji.

6.2 Sinhrona sekvenčna vezja

Najpreprostejša sinhrona sekvenčna vezja so tista, ki ciklično spreminjajo stanje na izhodu in jih imenujemo števci. Sinhroni števci so narejeni kot digitalna vezja s povratno vezavo, na kateri mora biti vsaj en sekvenčni gradnik (register ali flip-flop). Stanje izhoda se spreminja ob fronti ure, določeno pa je z notranjim stanjem in morebitnimi krmilnimi vhodi.

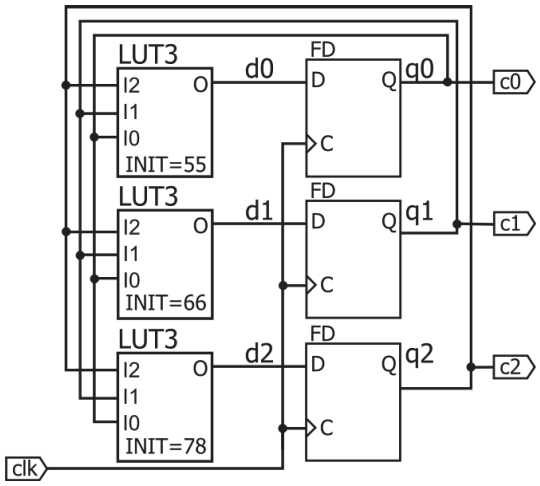


Slika 6.4: Izvedba sinhronega sekvenčnega vezja s povratno vezavo.

Sinhroni števec

Za izvedbo 3-bitnega sinhronega števca potrebujemo 3 podatkovne flip flope in 3-vhodno kombinacijsko vezje iz logičnih vrat ali vpoglednih tabel LUT. Delovanje števca opišemo s pravilnostno tabelo, ki določa vrednosti na vhodih flop-flopov ($d0$ do $d2$) glede na vrednosti trenutnih izhodov ($q0$ do $q2$). Pri binarnem števcu je vrednost v desnem stolpcu kar naslednja binarna kombinacija, zadnja kombinacija 111 pa se preslika v 000.

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 6.5: Tabela in zgradba 3-bitnega sinhronega števca.

Pravilnostna tabela opisuje kombinacijsko logiko na vhodih flip-flopov, ki jo najlažje naredimo s 3-vhodnimi vpoglednimi tabelami (LUT). V posamezne tabele vpišemo vrednost izhodnega stolpca pravilnostne tabele, npr. v tabelo signala $d2$ vpišemo kombinacijo 01111000.

Števec s kontrolnimi vhodi

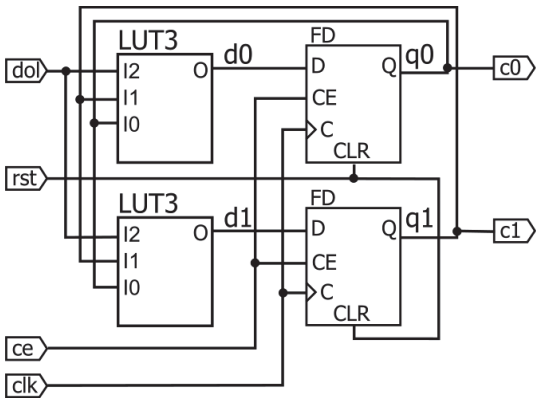
Sinhronemu števcu bomo dodali nekaj kontrolnih vhodov:

- ce – omogoči štetje, kadar je ce=1;
- dol – določa smer štetja; pri dol=1 šteje navzdol, sicer pa navzgor;
- rst – postavi izhod števca na 0.

Nekatere kontrolne signale lahko vežemo direktno na flip-flope, v našem primeru sta to signala ce in rst. Uporabiti moramo izvedbo podatkovnega flip-flopa, ki ima signal za omogočanje ure in reset signal. Signal za določanje smeri štetja pa bo vezan na kombinacijski del vezja kot dodatni vhod v vse vpogledne tabele.

Naredimo primer vezja 2-bitnega števca, ki šteje navzgor in navzdol. V pravilnostni tabeli imamo na levi strani poleg vhodov flop-flopov ($d0$ in $d1$) še signal dol. V prvih štirih vrsticah tabele je signal dol enak 0 in izhodi so enaki kot pri navadnem števcu, v drugih štirih vrsticah pa so izhodi takšni, da se smer štetja obrne.

dol	q1	q0	d1	d0
0	0	0	0	1
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	0

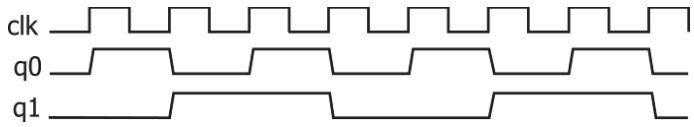


Slika 6.6: Tabela in zgradba 2-bitnega števca s kontrolnimi vhodi.

Števec po modulu in delilnik

Po opisanem postopku naredimo poljubne sinhrone števce, ki štejejo v različne smeri in z različnimi koraki štetja. Največje število bitov, ki ga potrebujemo za predstavitev izhodnega stanja (običajno je to najvišja vrednost števca), določa število izhodnih signalov, flip-flopov in vpogledne tabel.

Nekateri števci ne dajo na izhod vseh binarnih kombinacij, ker njihov cikel oz. modul štetja ni potenca števila 2, na primer števec po modulu 10 šteje od 0 do 9. Takšen števec bi uporabili za štetje enic pri digitalni uri, medtem ko bi za štetje desetic potrebovali števec po modulu 6.



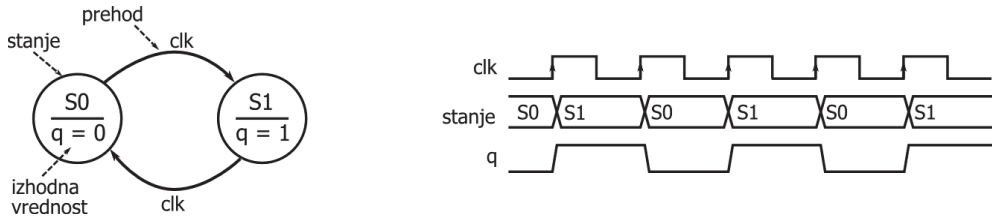
Slika 6.7: Časovni potek izhodov 2-bitnega števca.

Delilnik frekvence je primer vezja, kjer uporabimo binarni števec po določenem modulu z namenom, da zmanjšamo frekvenco ure v nekem delu vezja. Najvišji bit takšnega števca se spreminja s frekvenco, ki je enaka vhodni frekvenci deljeni z modulom štetja. Npr. dvobitni števec po modulu 4 (običajen binarni števec) spreminja izhod $q1$ s frekvenco, ki je 4-krat nižja od vhodne ure. Števec po modulu 1000 (to je 10 bitni števec) pa spreminja izhod $q9$ s 1000-krat nižjo frekvenco.

6.3 Načrtovanje z diagramom stanj

Model sinhronega sekvenčnega vezja lahko predstavimo z *diagramom stanj*. Gre za grafičen opis delovanja sistema, v katerem so prikazana stanja, izhodi in prehodi med stanji. Pri sekvenčnih vezjih je stanje kombinacija vrednosti, ki so shranjene v flip-flopih. Stanja na digramu prikazujemo s krožnicami, znotraj 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 6.8 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.

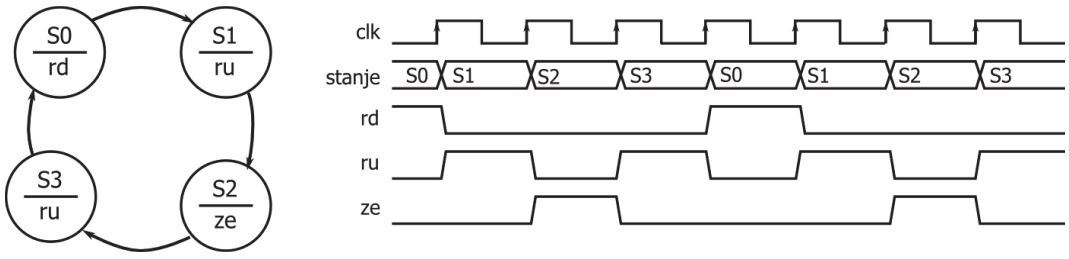


Slika 6.8: Diagram stanj in časovni potek na simulaciji.

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.

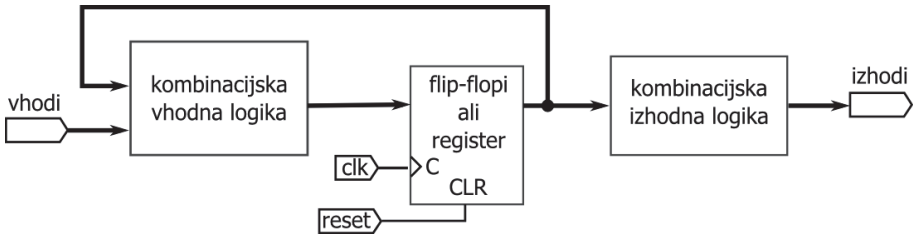
Diagram stanj pogosto uporabljamo pri načrtovanju sekvenčnih vezij, posebej pri izdelavi krmilne logike, ki generira izhodne signale v odvisnosti od vhodov in trenutnega stanja. Ogledali si bomo, kako z diagramom stanj opišemo sekvenčna vezja, ki *generirajo* izhod v danem časovnem zaporedju, in vezja, ki *zaznajo* določeno časovno zaporedje vhodnih signalov.

Narišimo diagram stanj za preprost semafor z izhodi: rd za rdečo luč, ru za rumeno in ze za zeleno. Semafor naj ima štiri stanja, ki se ciklično izmenjujejo v zaporedju: rd, ru, ze, ru.



Slika 6.9: Diagram stanj in časovni diagram za preprost semafor.

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.

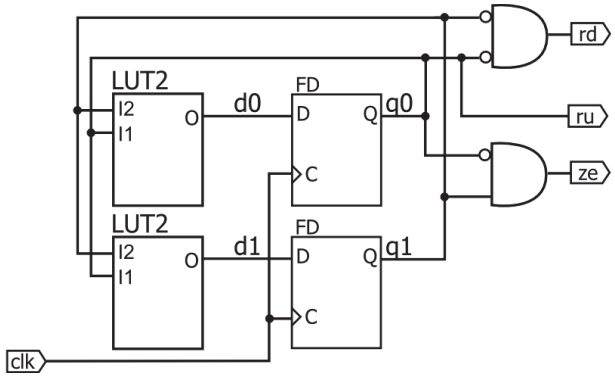


Slika 6.10: Blokovna shema končnega stroja stanj.

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. Vezje pa seveda lahko sintetiziramo tudi sami, in sicer z uporabo postopkov načrtovanja sekvenčnih vezij.

Načrtovanje vezja začnemo s tabelo, v kateri predstavimo prehode med stanji in kodiranje stanj. Za štiri stanja potrebujemo 2-bitni register stanj. Stanje $S0$ predstavlja kombinacija 00, $S1$ kombinacija 01 itn. Na podlagi prehajanja stanj določimo vrednosti na vhodih flop-flopov ($d0$ in $d1$) glede na vrednosti trenutnih izhodov ($q0$ in $q1$). Vrednosti iz tabele zapišemo v vpogledni tabeli LUT, ki predstavljata kombinacijsko vhodno logiko.

stanje	q1	q0	d1	d0	rd	ru	ze
S0	0	0	0	1	1	0	0
S1	0	1	1	0	0	1	0
S2	1	0	1	1	0	0	1
S3	1	1	0	0	0	1	0



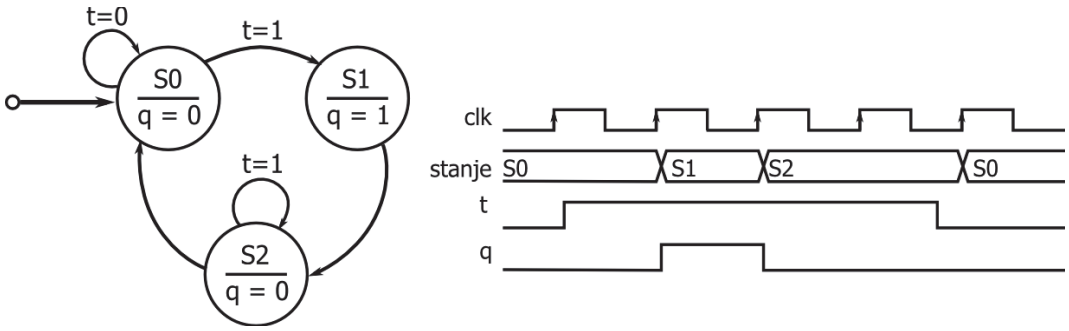
Slika 6.11: Izvedba preprostega semaforja s sekvenčnim vezjem.

Izhodni signali semaforja so odvisni od stanja. Izhodno logiko določimo kot kombinacijsko funkcijo stanja. Signal rd je ena, kadar sta $q0$ in $q1$ na 0; logika za ta signal so vrata AND z negatorji na vhodih. Iz tabele je razvidno, da se signal ru spreminja enako kot signal $q0$, zato zanj ne potrebujemo posebne logike.

6.4 Uporaba diagrama stanj

Oblikovanje impulza

Naredimo diagram vezja za oblikovanje kratkega impulza ob spremembi vhodne vrednosti na 1. Diagram vsebuje tri stanja: $S0$, $S1$ in $S2$, kot prikazuje slika 6.12. 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'.

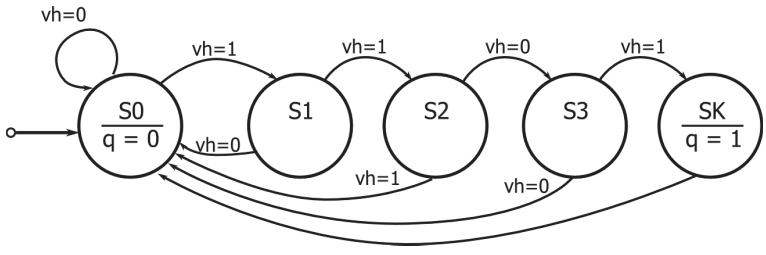


Slika 6.12: Diagram stanj in časovni diagram generatorja impulzov.

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.

Detekcija zaporedja

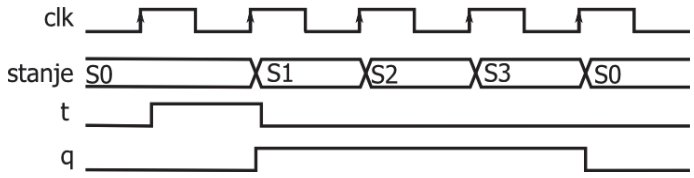
Diagram stanj uporabljamo za opis vezij, ki morajo generirati ali pa se odzivati v določenem zaporedju. Videli smo, da z diagramom lahko opišemo sekvenčno vezje, ki naredi na izhodu neko zaporedje stanj. Diagram na sliki 6.13 pa opravlja obratno nalogo: opazuje vhodni signal vh in ugotavlja, ali se spreminja 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 kakršne koli druge kombinacije se vrnemo v začetno stanje.



Slika 6.13: Diagram stanj za detekcijo zaporedja.

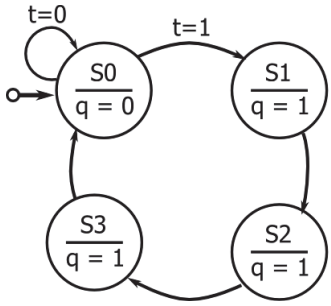
Vezje za oblikovanje daljšega impulza

Naredimo sekvenčni stroj, ki oblikuje na izhodu impulz dolžine treh urinih ciklov, kadar dobi na vhod b vrednost 1.



Slika 6.14: Časovni diagram signalov vezja za oblikovanje impulza.

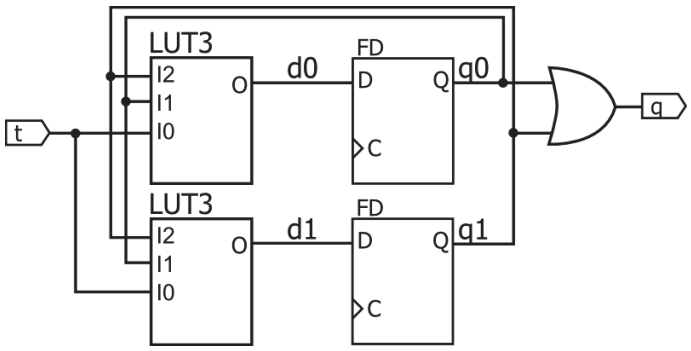
Na sliki 6.15 je diagram stanj vezja, ki vsebuje štiri stanja. Pri izdelavi tabele za vhodno in izhodno logiko moramo stanjem prirediti binarne kombinacije.



		vhod		izhod
	q1 q0 t	d1	d0	q
S0	0 0 0	0	0	0
	0 0 1	0	1	0
S1	0 1 0	1	0	1
	0 1 1	1	0	1
S2	1 0 0	1	1	1
	1 0 1	1	1	1
S3	1 1 0	0	0	1
	1 1 1	0	0	1

Slika 6.15: Diagram stanj in tabela prehajanja stanj.

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. Vhodna logika sekvenčnega vezja je narejena z dvema tabelama, izhod pa z logičnimi vrati OR.



Slika 6.16: Izvedba sekvenčnega stroja za oblikovanje impulza.

Naloge

1. Koliko flip-flopov vsebuje števec s signalom za omogočanje, ki šteje od 0 do 7?

2. Nariši diagram stanj sekvenčnega vezja za zaznavanje avtomobilov, ki zapeljejo na parkirišče. Ob vhodu sta dva senzorja: S1 in S2. Sekvenčno vezje naj za en cikel postavi izhod na 1, kadar se najprej aktivira S1, nato pa S2.