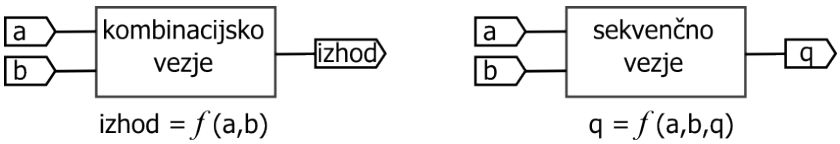
4
Sekvenčna vezja

4.1 Pomnilni gradniki

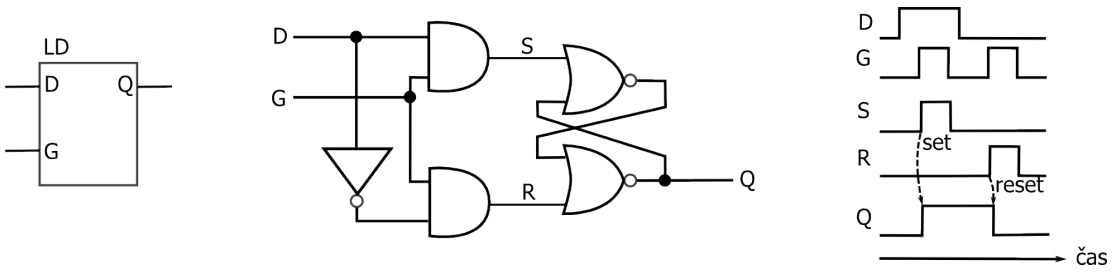
Do sedaj smo obravnavali kombinacijska vezja in gradnike, pri katerih je izhod odvisen le od trenutnega stanja na vhodu. Delovanje teh gradnikov predstavimo s preslikovalno tabelo ali pa opišemo z logično funkcijo, npr. $izhod = f(a, b)$. Pomnilni elementi pa spadajo med *sekvenčna vezja*, pri katerih je izhod odvisen od trenutne kombinacije na vhodih in preteklih vrednosti. Sekvenčna vezja omogočajo shranjevanje stanj in so narejeni iz logičnih vrat s povratno vezavo.



Slika 4.1: Kombinacijski in sekvenčni gradniki.

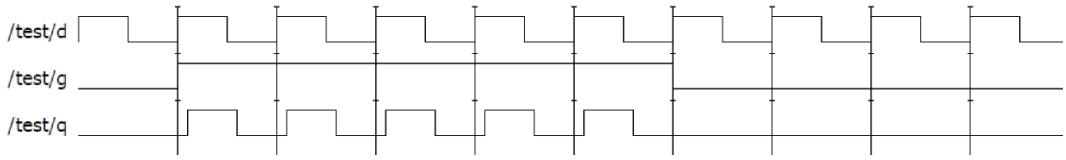
Najbolj enostaven pomnilni element je zapah (angl. latch) z oznako SR, ki ima dva vhoda: S (angl. Set) in R (angl. Reset). Zapah ohranja vrednost na izhodu, kadar sta oba vhoda na 0. Kadar je vhod S na 1, se izhod postavi na 1, kadar je R na 1 pa se izhod postavi na 0. V primeru, ko sta oba vhoda postavljena 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. Zapah SR izboljšamo tako, da z dodatno logiko na vhodu poskrbimo za pravilno krmiljenje S in R signalov, pri kateri ne pride nikoli do kombinacije $R = 1$ in $S = 1$.

Slika 4.2 prikazuje zapah D, ki je eden od osnovnih uporabnih pomnilnih elementov. 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



Slika 4.2: Zapah D: simbol, logična shema in časovni diagram.

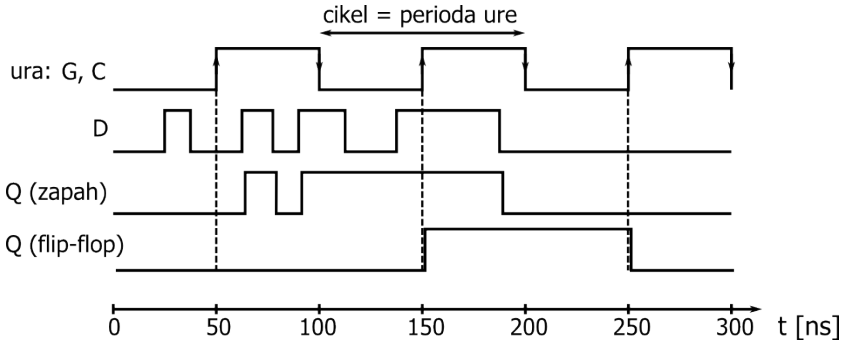
transparenten in vse spremembe vhoda z majhno zakasnitvijo prenaša na izhod, kot prikazuje slika 4.3. Ko gre signal G na 0, se stanje izhoda zapahne in ohranja zadnjo vrednost.



Slika 4.3: Časovni diagram zapaha pri hitrih spremembah vhoda.

Transparentnost zapaha in ohranjanje zadnjega stanja izkoriščajo nekatera programirljiva vezja za način delovanja z nizko porabo (angl. power save). Vezje troši tok, kadar digitalni signali spreminjajo stanja. Če uporabimo na vhodih zapahe, jih lahko zapahnemo takrat kadar nas spremembe vhodov ne zanimajo in s tem precej zmanjšamo porabo vezja. V družini vezij CPLD Coolrunner-II, se takšen zapah na vhodu imenuje Data Gate.

V sekvenčnih vezjih imamo običajno en signal, ki določa kdaj naj se izhodi spremenijo. Takšen signal je v obliki periodičnih impulzov in ga označujemo z imenom ura (angl. clock, clk). Ura je običajno signal, ki se v vezju najhitreje spreminja. Pogosto so vsi pomnilni elementi sekvenčnega vezja vezani na isto uro. Takšno vezje imenujemo **sinhrono** sekvenčno vezje. Sinhrona vezja so za analizo in obravnavo najbolj enostavna.

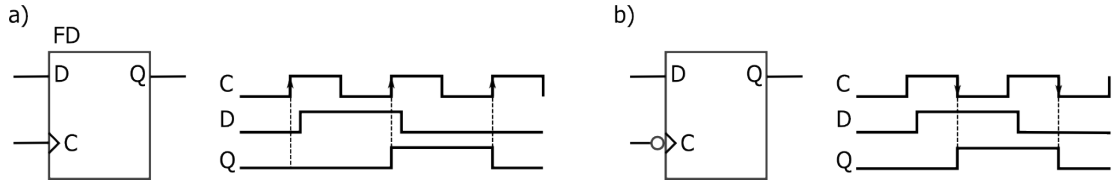


Slika 4.4: Proženje na nivo ali na fronto ure.

Signal ure v digitalnem vezju je v obliki periodičnih pravokotnih impulzov, kot prikazuje slika 4.4. Glavni parameter je perioda, ki predstavlja časovni interval med sosednjimi impulzi.

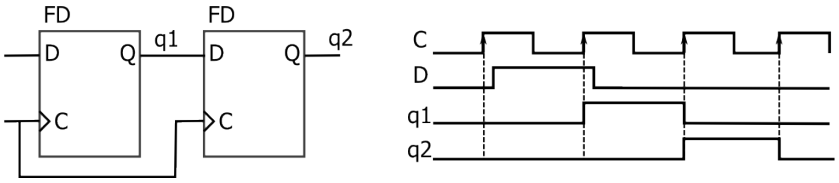
Časovni interval ene periode v katerem se stanje spremeni iz nizkega v visoko (ali obratno), imenujemo urin cikel. Včasih je pomembno tudi razmerje med trajanjem visokega in nizkega cikla, ki ga podajamo v odstotkih. Obratna vrednost periode je frekvenca, v primeru s slike velja: $f = 1/100\,\mathrm{ns} = 10\,\mathrm{MHz}$. 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.

Pomnilni elementi, ki spreminjajo izhod le ob fronti ure, se imenujejo **flip-flopi**. Pravimo, da so zapahi nivojsko proženi, flip-flopi pa robno proženi gradniki. Zaradi različnih zakasnitev pri potovanju signalov preko logičnih vrat se na signalih lahko pojavljajo občasni hitri preklopi med stanji. Pri prehodu skozi zapah se v času, ko je signal G na 1 preklopi prenašajo na izhod, pri prehodu skozi robno proženi flip-flop pa ne, kot prikazuje slika 4.4.



Slika 4.5: Simbol in časovni diagram za flip-flop D: a) na prednjo fronto in b) na zadnjo fronto ure.

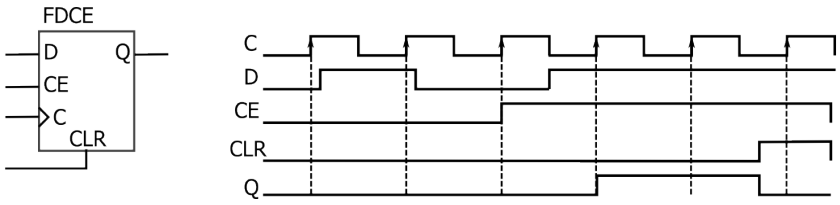
Med flip-flopi je najbolj osnoven gradnik flip-flop D, ki ima podatkovni vhod D, vhod za uro C in izhod Q. Poznamo flip-flope, ki so proženi na prednjo ali zadnjo fronto ure. Oba gradnika sta s tipičnimi časovnimi diagrami prikazana na sliki 4.5. Na časovnem diagramu vidimo, da flip-flop D zakasni spremembo na vhodu za en cikel ure. Če vežemo drug za drugim 2 flip-flopa, bomo zakasnili signal za 2 cikla, kot prikazuje slika 4.6, itn. Takšna vezava je osnova za izvedbo elementov, ki jih imenujemo pomikalni registri.



Slika 4.6: Zakasnitev signalov pri prehodu čez zaporedne flip-flope.

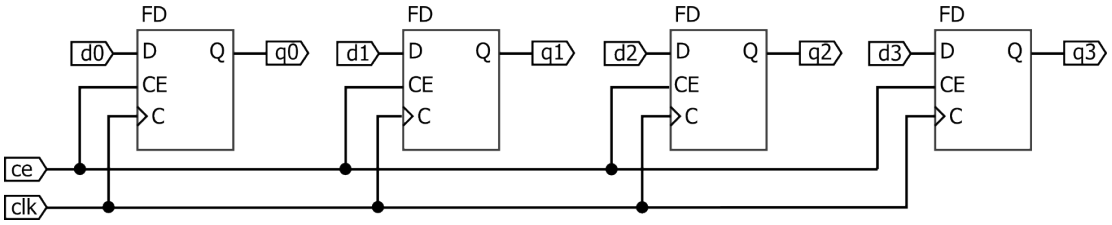
Pogosto potrebujemo še kakšne dodatne kontrolne signale: npr. FDCE na sliki 4.7 ima 2 dodatna vhoda. S kontrolnim signalom CE (angl. Clock Enable) določamo ali flip-flop normalno vzorči vhod ob fronti ure, ali pa se, kadar je $CE = 0$, vzorčenje ob fronti ne izvede. Z aktiviranjem kontrolnega vhoda CLR (angl. clear) pa postavimo izhod na 0, kot prikazuje diagram na sliki 4.7.

Kontrolni signal CLR deluje kot reset pri zapahu; ko ga postavimo na 1, gre izhod flip-flopa takoj na 0, ne glede na fronto ure. Pravimo, da ima flip-flop asinhroni reset, ker se izhod spremeni asinhrono glede na uro. Ko gre signal CLR nazaj na 0, začne flip-flop delovati kot običajno in izhod se lahko spremeni na 1 šele ob naslednji prednji fronti ure.



Slika 4.7: Flip-flop D s signalom za omogočanje (CE) in asinhronim reset (CLR).

Posamezen flip-flop shrani en podatkovni bit. Če jih povežemo več vzporedno, tako da imajo skupne kontrolne signale, dobimo register. Slika 4.8 prikazuje logično shemo 4-bitnega registra z uro in signalom za omogočanje. Register je osnovni gradnik vezja, ki omogoča shranjevanje večbitnega podatka.



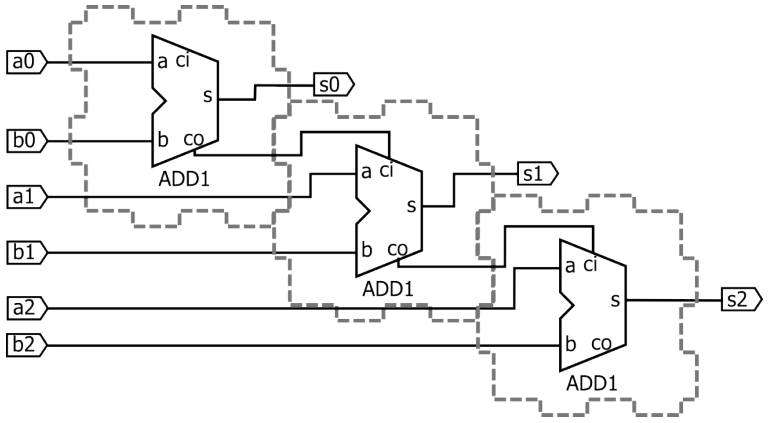
Slika 4.8: Shema 4-bitnega registra.

4.2 Sestavljanje gradnikov

V procesu izdelave digitalnega vezja sestavljamo vnaprej pripravljene gradnike v končno vezavo. Gradniki morajo spoštovati statični red signalov in jih povezujemo tako, da posamezen izhod vežemo na enega ali več vhodov drugega gradnika. Logične izhode, razen v posebnih primerih, ne smemo vezati skupaj, ker lahko naredimo kratek stik.

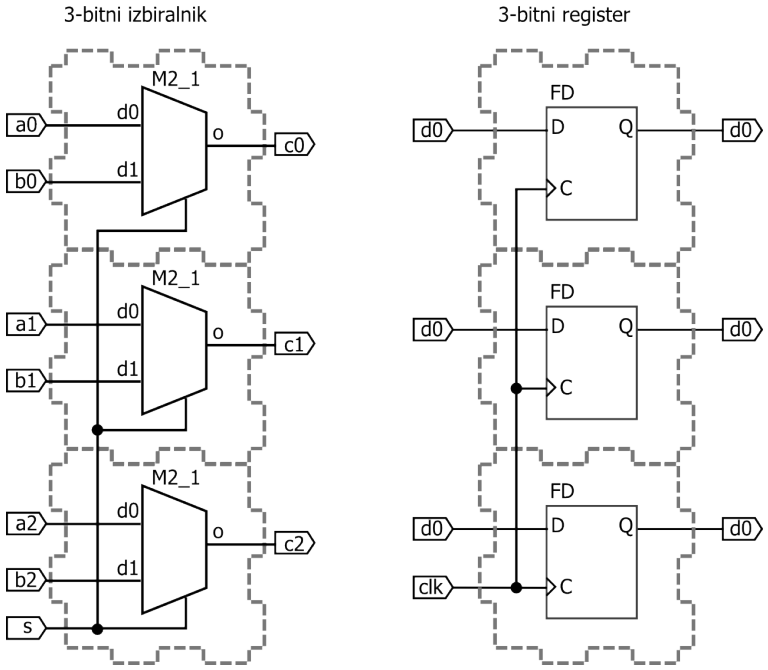
Povezovanje digitalnih gradnikov ni težko in ga včasih primerjamo z zlaganjem kock. Gradniki imajo vedno vhode na levi in izhode na desni strani. Kadar jih vežemo vzporedno, povežemo skupaj nekatere kontrolne signale, podatkovni vhodi in izhodi posameznih gradnikov pa so neodvisni.

Primer *zaporedne vezave* gradnikov je vezava polnih seštevalnikov v večbitnem seštevalniku. Na sliki 4.9 je shema 3-bitnega seštevalnika, kjer so posamezni gradniki povezani drug za drugim preko vhodov in izhodov za prenos. Pri seštevanju v splošnem vsaka števka na vhodu vpliva na celoten rezultat, npr. sprememba na signalu $a0$ lahko vpliva na spremembo izhoda $s2$. Signal mora potovati čez vse gradnike in zakasnitve se seštevajo.



Slika 4.9: Izvedba 3-bitnega seštevalnik z zaporedno vezavo.

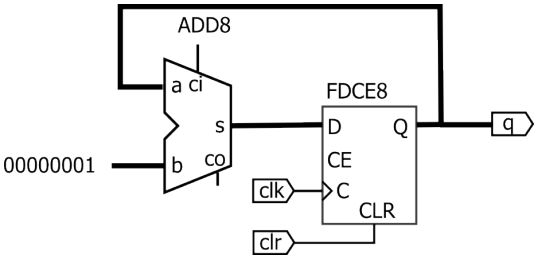
Primer *vzporedne vezave* gradnikov je na sliki 4.10. Značilnost vzporedne vezave je, da ne poveča *zakasnitev* signalov v primerjavi s posameznim gradnikom. V splošnem pričakujemo, da imajo večja vezja večjo zakasnitev, ker morajo signali potovati med zaporednimi gradniki ali logičnimi vrati. Vsak element 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.



Slika 4.10: Izvedba 3-bitnega izbiralnika in registra z vzporedno vezavo.

4.3 Diagram stanj

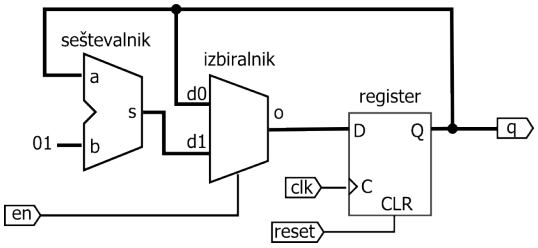
S povezavo kombinacijskih in sekvenčnih elementov dobimo nove uporabne elemente vezja. Vzemimo večbitni seštevalnik in na en vhod pripeljemo vrednost 1 v binarni obliki, npr. kombinacijo ”00000001”na vhod 8-bitnega seštevalnika. Izhod povežemo na vhod registra, izhod registra pa nazaj na drug vhod seštevalnika, kot prikazuje slika 4.11.



Slika 4.11: Shema 8-bitnega števca.

Naredili smo števec, ki ob vsaki novi fronti ure poveča izhod za 1. S signalom clr postavimo števec v začetno stanje ”00000000”, uporabimo pa lahko tudi signal za omogočanje registra s katerim bi omogočili prištevanje vrednosti ali pa ohranjali zadnje stanje.

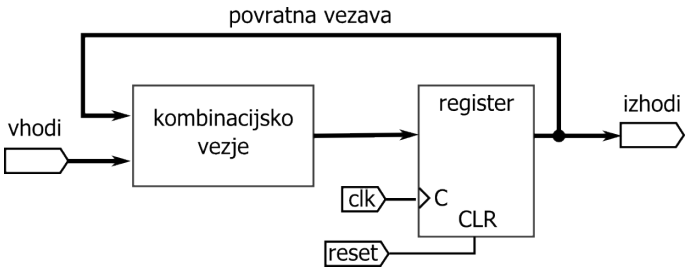
Analizirajmo še enkrat primer števca z vhodom za omogočanje. Stanje števca shranjuje sekvenčni gradnik - register. Register naj bo tokrat sestavljen iz osnovnih pomnilnih gradnikov, ki imajo le dva kontrolna vhoda: uro clk in signal clr za postavljanje izhoda na 0. Logiko za omogočanje naredimo z izbiralnikom, ki ob $en = 0$ pripelje na vhod registra kar njegov izhod, tako da register ohranja vrednost. Kadar je $en = 1$ pa dobi register na vhod vrednost izhoda povečano za 1, ki jo dobimo iz seštevalnika.



Slika 4.12: Shema števca s signalom za omogočanje.

Števec je sestavljen iz kombinacijskega vezja, ki ga predstavljata seštevalnik in izbiralnik ter sekvenčnega registra. Povezavo iz izhoda registra, preko kombinacijskega vezja nazaj na vhod registra imenujemo **povratna vezava**. Povratna vezava je osnovni princip s katero so narejena sekvenčna vezja, kot prikazuje slika 4.13. V povratni vezavi mora biti vključen vsaj en sekvenčni gradnik (register ali flip-flop).

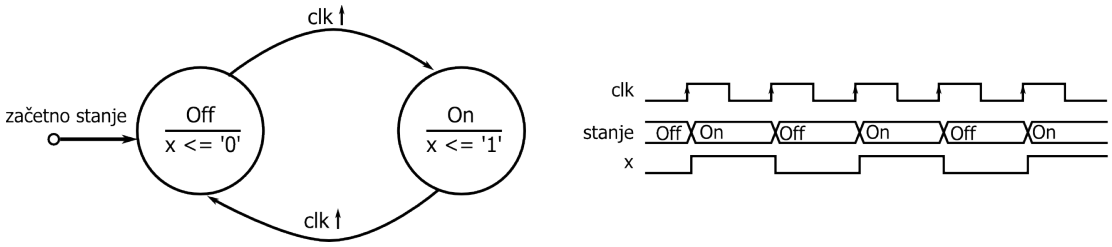
Za opis sekvenčnih vezij s povratno vezavo uporabljamo **diagram stanj**. Diagram stanj je grafičen opis sekvenčnega vezja, v katerem so prikazana stanja vezja in pogoji za prehod med stanji. Stanje sekvenčnega vezja je določeno z logičnim stanjem na izhodu registra. Register



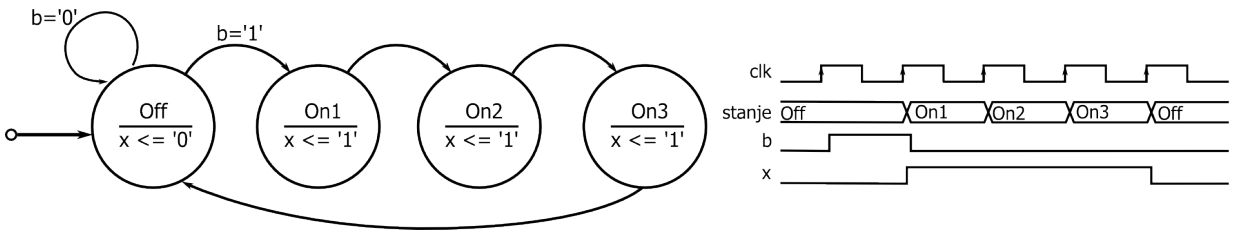
Slika 4.13: Shema sekvenčnega vezja s povratno vezavo.

z $n-$bitnim izhodom ima 2^n možnih binarnih stanj in to je tudi največje število kombinacij, ki predstavljajo stanja. V diagramu stanj uporabljamo namesto binarnih kombinacij kar simbolična imena stanj. Stanja so predstavljena s krožnicami v katerih je ime stanja, prehodi pa s puščicami. Na puščice zapišemo pogoje za prehod. S puščico posebej označimo začetno stanje, v katerega pridemo ob signalu reset.

Slika 4.14 prikazuje primer preprostega diagrama z dvemi stanji: Off in On. Poleg stanja smo v krožnici zapisali vrednost izhodnega signala x. V začetnem stanju Off je izhod x enak '0'. Ob prednji fronti ure se izvrši prehod v stanje On v katerem se postavi x na vrednost '1'. Ob naslednji prednji fronti gremo spet nazaj v Off. Na časovnem izseku iz simulacije vidimo preklapljanje med stanji in potek izhodnega signala. Na izhodu x dobimo periodičen signal, ki je dvakrat počasnejši od vhodne ure.



Slika 4.14: Diagram stanj in časovni potek na simulaciji.



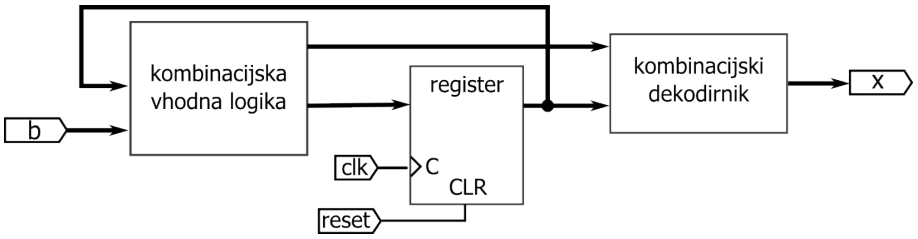
Slika 4.15: Diagram stanj in časovni diagram za 3 cikle dolge impulze na izhodu.

Programska oprema na podlagi diagrama stanj avtomatsko naredi ustrezno sekvenčno vezje. Za preprost diagram z dvemi stanji bo v vezju le en flip-flop in enostavna kombinacijska logika.

Ker bodo naša sekvenčna vezja vedno sestavljena iz flip-flopov ali registrov, ki preklapljajo izhode ob prednji fronti ure, lahko pri risanju diagrama izpustimo vsakokratno pisanje ure pri prehodih stanj. Prehodi med stanji so običajno odvisni tudi od vrednosti vhodnih signalov.

Poglejmo si primer vezja, ki naredi na izhodu impulz dolžine treh urinih ciklov, kadar dobi na vhod b vrednost 1. V diagramu imamo 4 stanja, kor prikazuje slika 4.15.

V začetnem stanju Off ostaja vezje toliko časa, dokler ni izpolnjen pogoj $b =$ '1' in ne pride prednja fronta ure, ki je nismo posebej označili. Nato se ob prednjih frontah zgodijo prehodi v stanja On1, On2 in On3, v katerih dobi izhodni signal x vrednost 1. Ob naslednji fronti pridemo ponovno v začetno stanje, v katerem je izhod x enak 0. Začetno stanje se ohranja dokler je vhodni signal b enak '0', kar smo posebej označili s prehodom stanja samo vase. Načeloma tega ni potrebno označiti, saj je privzeto da se stanje ohranja, dokler niso izpolnjeni pogoji za prehod. Vezje je narejeno s končnim avtomatom, ki je shematsko prikazan na sliki 4.16.

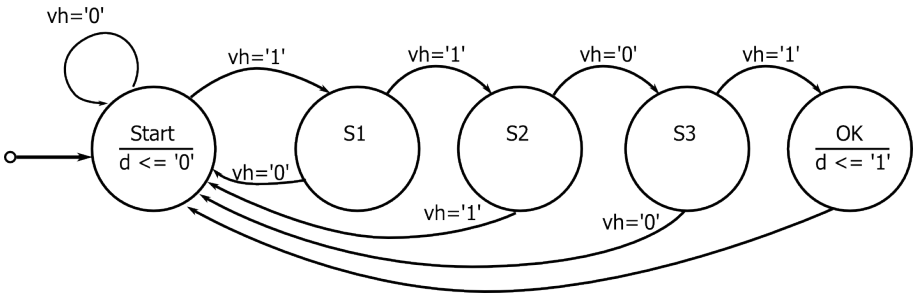


Slika 4.16: Shema končnega avtomata za izvedbo diagrama stanj.

Končni avtomat vsebuje register, kombinacijsko vhodno logiko za prehode med stanji in dekodirnik za izhodne signale. Programska oprema določi velikost registra na podlagi števila stanj - v primeru 4 stanj potrebujemo 2-bitni register. Nato določi optimalno izvedbo kombinacijske vhodne logike in dekodirnika za izhodne signale.

Primer 4.1

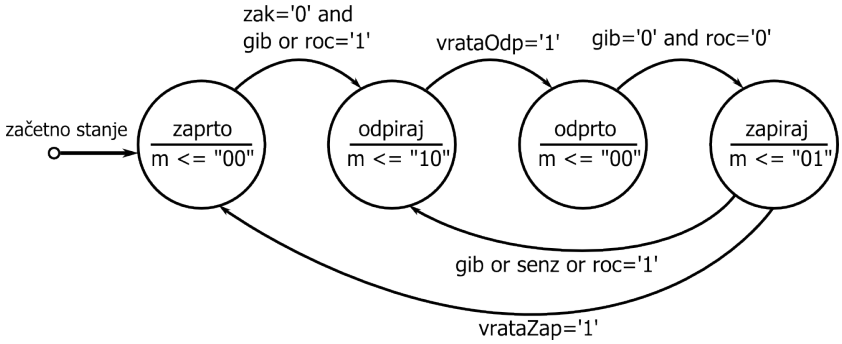
Diagram stanj uporabljamo za opis vezij, ki morajo generirati ali pa se odzivati v določenem zaporedju. Videli smo, da z diagramom lahko opišemo sekvenčno vezje, ki naredi na izhodu neko zaporedje stanj. Diagram na sliki 4.17 pa opravlja obratno nalogo: opazuje vhodni signal vh in ugotavlja ali se spreminja po določenem zaporedju. Če se vhod ob zaporednih frontah ure postavi na vrednosti 1, 1, 0 in 1, gre avtomat skozi vsa stanja do končnega, kjer postavi izhod d na 1. V primeru kakršnekoli druge kombinacije se avtomat vrne v začetno stanje.



Slika 4.17: Diagram stanj za detekcijo zaporedja.

Primer 4.2

Poglejmo primer diagrama stanj za krmiljenje avtomatskih vrat, ki ima več vhodnih signalov in pogojev za prehode, kot prikazuje slika 4.18. Začetno stanje je zaprto in v tem stanju ostajamo, dokler ne zaznamo gibanja (gib) ali ročnega odpiranja (roc) pod pogojem, da vrata niso zaklenjena (zak). Nato gremo v stanje odpiraj, kjer se vključi motor za odpiranje vrat ($m <= 10$). Iz tega stanja gremo naprej, ko senzor na vratih javi, da so do konca odprta ($vrataOdp$). V stanju odprto je motor ponovno neaktiven. Iz tega stanja gremo v stanje zapiranja ko senzor ne zaznava več gibanja in ni aktivirano ročno odpiranje. V stanju zapiraj se aktivira motor za zapiranje vrat ($m <= 01$). V primeru, da zaznamo gibanje ali senzor ovire med vrati ali pa je aktiviramo ročno odpiranje se takoj vrnemo na stanje odpiranja. Če pa se vrata do konca zaprejo, kar javi senzor vrataZap, pridemo ponovno v začetno stanje.



Slika 4.18: Diagram stanj za avtomatska vrata.