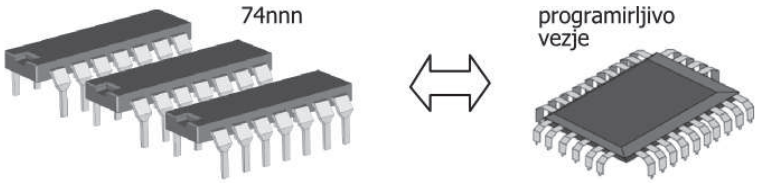
7 Programirljiva logika

Programirljiva logična vezja so elektronske komponente, v katerih s postopkom programiranja oz. konfiguracije oblikujemo digitalno vezje, da opravlja želeno funkcijo. Razlikujejo se po zgradbi, načinu programiranja in zmogljivosti. Spoznali bomo osnovne programirljive matrike PAL in PLA, programirljive naprave CPLD ter zelo zmogljive programirljive matrike FPGA.

7.1 Programirljiva integrirana vezja

Razvoj novega integriranega vezja je dolgotrajen in drag postopek, zato je veliko digitalnih naprav in sistemov narejenih z obstoječimi komponentami (angl. commercial off-the-shelf, COTS). Prve digitalne komponente v obliki integriranih vezij so bili gradniki, ki jih dobimo v družini integriranih vezij *7400*. Tiskana vezja so vsebovala veliko osnovnih integriranih vezij in inženirji so iskali možnost zamenjave množice komponent z enim integriranim vezjem.



Slika 7.1: Zamenjava osnovnih logičnih komponent z enim *programirljivim* vezjem.

Tehnologija programirljivih integriranih vezij omogoča spreminjanje delovanja že izdelanih vezij s spremembami v programski in strojni opremi. Programsko opremo lahko spreminjamo v mikroprocesorskih sistemih, strojno opremo pa v programirljivih vezjih. Novejša vezja

združujejo obe možnosti za izdelavo zelo prilagodljivih digitalnih naprav in sistemov. Prednosti programirljivih integriranih vezij:

- programirljiva vezja omogočajo hiter razvoj prototipa vezja za nov izdelek;
- stroški razvoja so nižji, ker odpade zelo draga priprava proizvodnje polprevodnikov;
- na tiskanem vezju je manj elektronskih komponent, ki so lažje nadomestljive, saj lahko načrt vezja hitro prenesemo v drugo programirljivo vezje.

Logični gradniki, ki omogočajo programiranje integriranih vezij, zasedejo velik del vezja in vezje v praksi ni nikoli 100-odstotno zasedeno. Zaradi tega imajo programirljiva vezja v primerjavi z namenskimi integriranimi vezji nekaj slabosti:

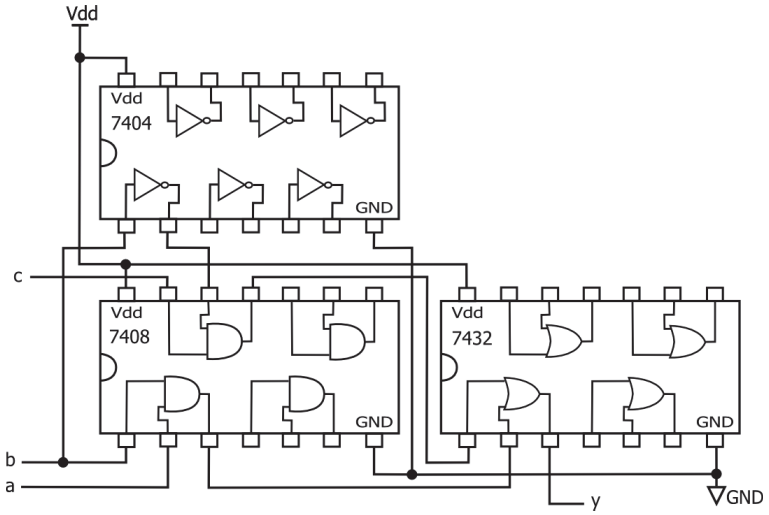
- zaradi večje površine je cena posameznega vezja višja kot cena namenskega vezja;
- programirljivi elementi vnašajo zakasnitve, zato so ta vezja nekoliko počasnejša;
- imajo večjo porabo energije.

7.2 Osnovne programirljive matrike

Poglejmo primer kombinacijskega vezja, ki izvaja logično funkcijo:

$$y = (a \text{ AND } b) \text{ OR } (c \text{ AND NOT } b)$$

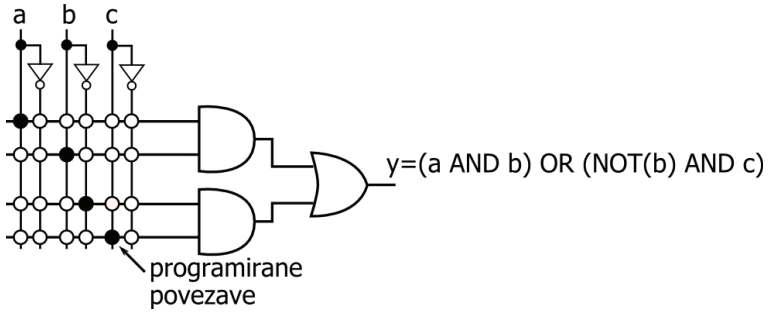
Logično vezje lahko naredimo s tremi integriranimi vezji: 7408, ki vsebuje logična vrata AND, 7432 z vrati OR in 7404, ki vsebuje negatorje.



Slika 7.2: Izvedba logičnega vezja s komponentami iz družine 7400.

Logično funkcijo lahko opišemo s tabelo, ki jo s programiranjem prenesemo v pomnilnik ROM. Vezje naredimo tako, da so vhodni signali vezani na pomnilniške naslove, izhodni pa na podatkovne izhode. Pri večjem številu vhodnih in izhodnih signalov je takšna izvedba zelo potratna glede površine integriranega vezja, sekvenčnih vezij pa s pomnilnikom ROM ne moremo narediti. Princip izdelave kombinacijskih gradnikov vezja z manjšim pomnilnikom oziroma vpogledno tabelo bomo zasledili v strukturah zmogljivih programirljivih vezij.

Kombinacijsko vezje lahko naredimo tudi s *programirljivo matriko*, v kateri programiramo povezave vhodnih signalov na logična vrat AND.



Slika 7.3: Izvedba logičnega vezja s programirljivo matriko.

Programirljiva matrika vsebuje vnaprej povezana logična vrata AND in OR, vsak vhodni signal pa je vezan direktno ali prek negatorja na vertikalne povezave. Na križiščih teh povezav s horizontalnimi povezavami proti vhodom logičnih vrat so programirljivi elementi, ki omogočajo, da povezave sklenemo ali razklenemo. Na shemi matrike so sklenjene oz. programirane povezave, narisane s polnim, razklenjene pa s praznim krogcem.

Izvedba vezja z matriko PAL

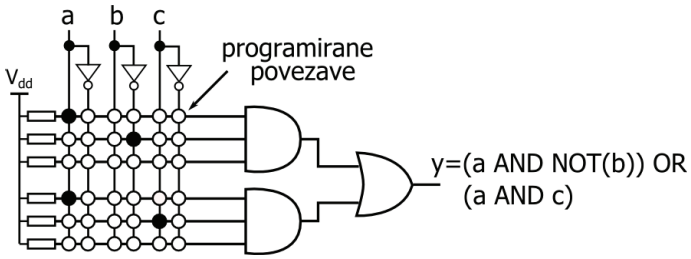
Boolova algebra pravi, da je kombinacijske funkcije vedno mogoče pretvoriti v obliko, kjer je vsak izhodni signal zapisan kot vsota (vrata OR) produktov (vrata AND) vhodov oz. negiranih vhodov. Programirljiva matrika PAL (angl. Programmable Array Logic) je sestavljena iz večvhodnih vrat AND, ki jim v postopku programiranja določimo povezave na vhodne oz. negirane vhodne signale. Pri izvedbi vezja z matriko PAL moramo kombinacijsko logiko v prvem koraku pretvoriti v primerno obliko. Tako na primer logično funkcijo za avtomobilski alarm:

$$alarm = vklop \text{ AND } (NOT(vrata) \text{ OR } gib)$$

pretvorimo v obliko:

$$alarm = (vklop \text{ AND } NOT(vrata))\text{OR}(vklop \text{ AND } gib)$$

Za izvedbo te funkcije potrebujemo dvoje dvovhodnih vrat AND, ki so vezana na vrata OR. Programirljiva matrika PAL ima v praksi logična vrata AND z nekaj 10-vhodnimi signali in večvhodna vrata OR, s katerimi naredimo uporabna kombinacijska vezja. V shemah bomo uporabljali manjše matrike zaradi nazornosti prikaza.



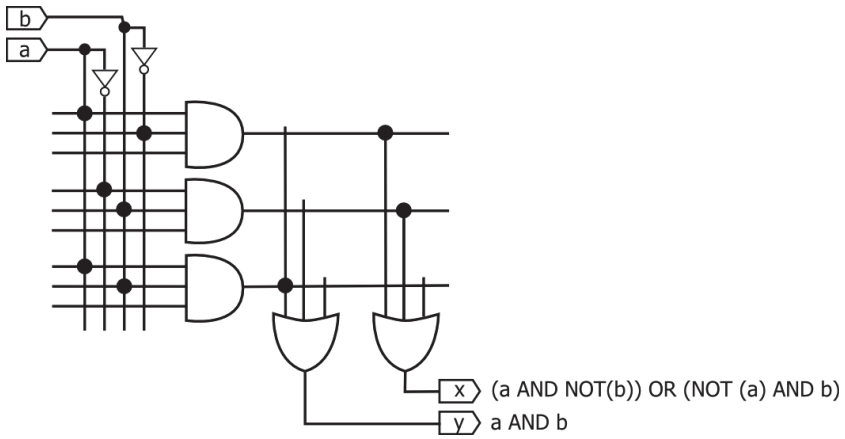
Slika 7.4: Avtomobilski alarm v programirljivi matriki PAL.

V matriki PAL, s katero bi naredili avtomobilski alarm, so prikazane programirane povezave s polnim krogcem. Na horizontalnih povezavah so tudi upori proti napajalni napetosti Vdd, ki zagotavljajo logično 1, kadar so vse povezave prekinjene. Logična 1 na posameznem vhodu vrat AND ne vpliva na funkcijo ostalih vhodov, tako npr. 3-vhodna vrata AND uporabljamo kot 2-vhodna vrata.

Programirljiva matrika PAL je najpreprostejše programirljivo vezje, ki omogoča izvedbo kombinacijskih logičnih vezij. Za izvedbo kompleksnejših funkcij potrebujemo zelo velike matrike, ki so v splošnem slabo izkoriščene. Upori proti napajalni napetosti prestavljajo stalen porabnik toka, zato so ta vezja tudi energijsko potratna. Integrirana vezja PAL se danes ne uporabljajo več, predstavili pa smo jih zato, ker na najbolj nazoren način razložijo zgradbo kompleksnih programirljivih naprav.

Izvedba vezja z matriko PLA

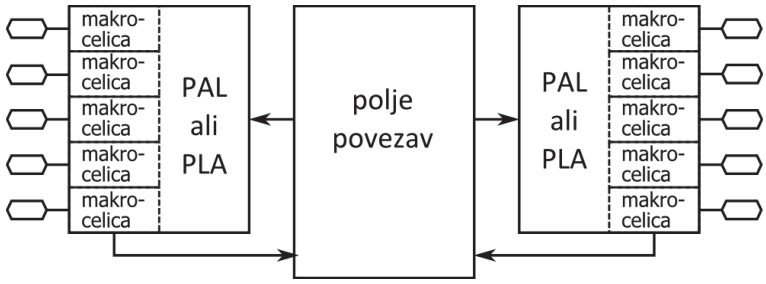
Programirljive matrike z oznako PLA (angl. Programmable Logic Array) imajo možnost programiranja povezav tako na vhodih AND kot na vhodih vrat OR. Takšna matrika doseže boljšo izkoriščenost programirljivega vezja, saj lahko nekatere produktne člene uporabimo večkrat. Primer matrike PLA, ki ima tri produktne člene ter po dva vhodna in izhodna signala, kaže slika 7.5.



Slika 7.5: Programirljiva matrika PLA.

7.3 Programirljive naprave – CPLD

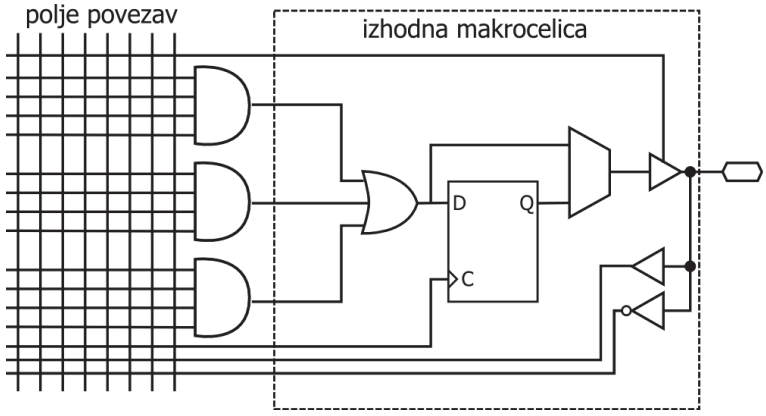
Programirljive naprave z oznako CPLD (angl. Complex Programmable Logic Device) vsebujejo več matrik PLA, ki so med seboj povezane s programirljivim povezovalnim poljem.



Slika 7.6: Blokovna shema vezja CPLD (Xilinx Coolrunner-II).

Vezja Coolrunner-II proizvajalca Xilinx vsebujejo matrike PLA, ki imajo 56 produktnih členov in 16 izhodnih signalov. Kombinacijski izhodi matrike PLA so vezani na makrocelice, ki vsebujejo pomnilne gradnike za izvedbo sekvenčnih vezij. Signali so povezani prek vhodno/izhodnih (I/O) celic na zunanje priključke ali povezovalno polje.

Kompleksne gradnike, ki jih ne moremo narediti v eni matriki PLA, razdelimo in naredimo z več matrikami. Preslikavo logične sheme v strukturo PLA opravlja programska oprema, tako da za njihovo uporabo ni potrebno podrobno poznavanje zgradbe vezij PLD.

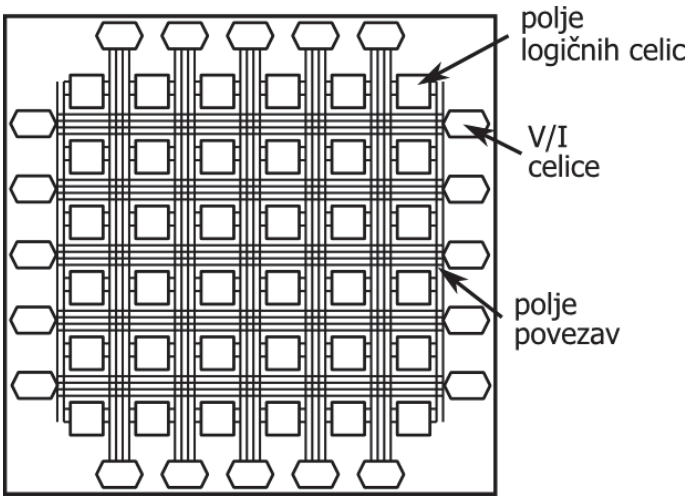


Slika 7.7: Izhodna makrocelica vezja CPLD.

Pri programiranju vezja se določijo povezave v matriki PLA ter povezave v makrocelicah in I/O-blokih ter povezovalni matriki. S takšno strukturo lahko naredimo poljubna digitalna vezja, omejeni smo le z velikostjo gradnika CPLD. Vezja CPLD omogočajo izdelavo logičnih vezij z nekaj 1000 logičnih vrat in nekaj 100 flip-flopov, ki delujejo pri frekvencah ure do okoli $200\,\mathrm{MHz}$. Imajo programski pomnilnik vrste FLASH, ki ga lahko večkrat zapišemo in ohrani vsebino ob izklopu napajanja.

7.4 Programirljiva polja vrat – FPGA

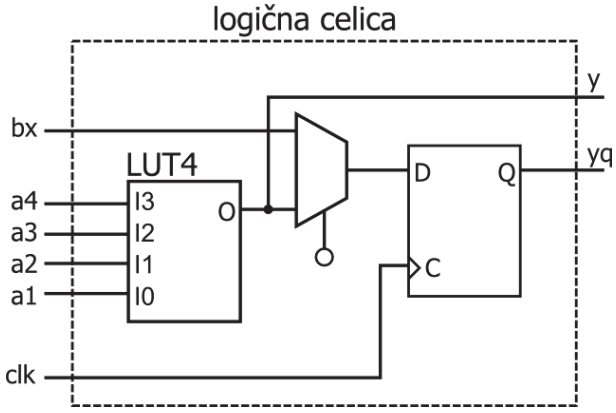
Vezja z oznako FPGA (angl. Field Programmable Gate Array) so narejena iz množice programirljivih logičnih celic in polja povezav, v katerem lahko med seboj povežemo poljubne celice. Okoli programirljive matrike so vhodno/izhodne celice, ki povezujejo signale na zunanje priključke.



Slika 7.8: Blokovna shema vezja FPGA.

Vezja FPGA vsebujejo veliko število celic in največja med njimi omogočajo izdelavo vezij z več kot 10 milijoni logičnih vrat. Sodobna vezja FPGA omogočajo izdelavo celotnih sistemov na integriranem vezju in lahko vsebujejo tudi mikroprocesorje, pomnilnike in namenske vmesnike.

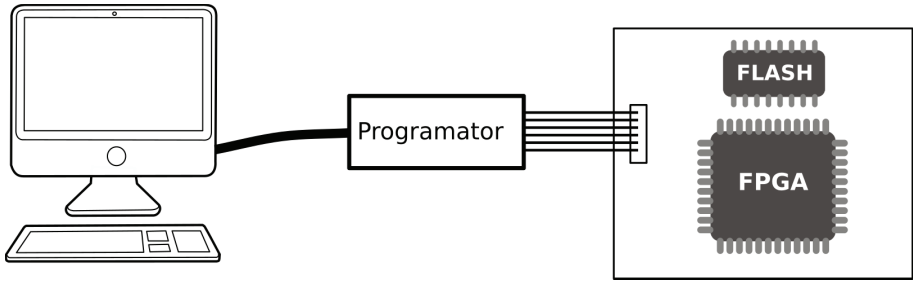
Z logičnimi celicami naredimo gradnike kombinacijskih in sekvenčnih vezij, ki jih s povezavami med celicami združujemo v digitalna vezja. Zgradba celic je odvisna od proizvajalca in družine vezij FPGA, v splošnem pa vsebujejo vpogledne tabele, programirljive izbiralnike in flip-flope.



Slika 7.9: Blokovna shema logične celice.

Struktura vezja FPGA je zelo prilagodljiva in omogoča programiranje povezav med logičnimi celicami, signalov znotraj logičnih celic in vhodno/izhodnih celic ter vsebine vpoglednih tabel. Izvedba povezav je odvisna od tehnologije programirljivega vezja.

V tehnologiji *antifuse* imajo tovarniško izdelana vezja povsod šibke povezave. Nekatere imed povezav v postopku programiranja prekinemo in tako ustvarimo želeno funkcijo. Takšna vezja lahko programiramo samo enkrat. V tehnologijah *EPROM* ali *Flash* je na mestu povezave posebno elektronsko stikalo, ki mu pri programiranju določamo stanje. Programiranje teh vezij lahko ponovimo večkrat. Največ integriranih vezij FPGA je narejenih v tehnologiji CMOS, kjer se programski podatki zapišejo v zapahe. Zapah ob izklopu napajanja izgubi shranjeno stanje, zato imamo poleg vezja FPGA na tiskanem vezju še pomnilnik Flash, iz katerega se ob zagonu naloži vsebina.

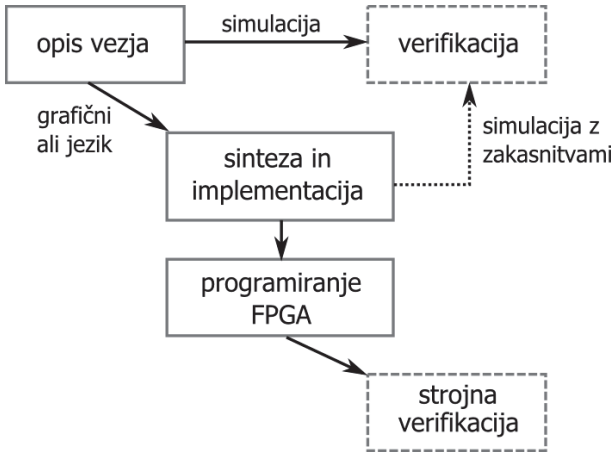


Slika 7.10: Programiranje vezja FPGA.

7.5 Računalniška orodja za programirljiva vezja

Postopek načrtovanja vezja začnemo z vnašanjem opisa vezja v računalnik. Programska oprema za računalniško načrtovanje vezij pozna različne načine vnosa vezja, ki jih v grobem razdelimo v grafični in jezikovni opis. Primer grafičnega opisa je shema vezja. Shemo narišemo v grafičnem urejevalniku s postavljanjem elementov iz knjižnice digitalnih gradnikov in risanjem povezav. Drug primer grafičnega opisa vezja je opis sekvenčnega stroja v obliki diagrama stanj. Jezikovni opis logičnega vezja predstavljajo npr. Boolove enačbe. Jezik za opis strojne opreme HDL (angl. Hardware Description Language) določa pravila takšnega opisa. Danes se uporabljata predvsem jezika VHDL in Verilog, ki omogočata opis vezja z logičnimi izrazi ali pa v obliki algoritma.

Programska oprema datoteke z opisom vezja prevede v računalniku razumljivo jezikovno obliko (HDL), ki je osnova za izvedbo simulacije in ostalih korakov prevajanja. Jezikovni opis vezja se v koraku *sinteze* pretvori v obliko, ki vsebuje vse elemente končnega vezja in povezave med njimi (angl. netlist). Ta datoteka je osnova za naslednje korake prevajanja, ki jih imenujemo tehnološka izvedba oz. *implementacija* vezja.



Slika 7.11: Osnovni koraki načrtovanja s programirljivimi vezji.

Delo razvojnega inženirja je predvsem opis vezja in preverjanje delovanja oz. verifikacija vezja. Verifikacijo najprej opravimo z računalniško simulacijo. Zapleten postopek prevajanja vezja je na srečo avtomatiziran. Programska oprema za računalniško načrtovanje vezij zahteva le nekaj nastavitev, da se implementacija izvede pod želenimi pogoji. Rezultat prevajanja lahko ponovno verificiramo s simulacijo, ki tokrat vsebuje tudi ocenjene zakasnitve gradnikov vezja. Postopek programiranja vezja je odvisen od strojne opreme in izvedbe komunikacije z računalnikom. Običajno ta korak ni zahteven, saj moramo le izbrati pravilne nastavitve in datoteko, ki naj jo programska oprema naloži v vezje.

Po programiranju preverimo delovanje vezja na strojni opremi (npr. na razvojnem sistemu). Ta postopek imenujemo strojna verifikacija. Postopek strojne verifikacije je odvisen predvsem od vrste vezja in nalog, ki jih vezje izvaja. V najpreprostejši obliki zadoščata ročno nastavljanje signalov in vizualni pregled delovanja, v zahtevnejših primerih pa potrebujemo posebno merilno opremo.

Tehnološka izvedba v programirljivem vezju

Korake tehnološke izvedbe vezja bomo predstavili na primeru majhnega digitalnega vezja, ki vsebuje kombinacijske in sekvenčne gradnike. Vezje izvaja logične operacije nad 3-bitnima vhodoma $r1$ in $r2$ glede na stanje krmilnega vhoda p. Kadar je p enak 1, se izvede operacija $r1$ AND $r2$, sicer pa $r1$ OR $r2$. Rezultat operacije se prenese na izhod vezja ($r3$) ob fronti ure.

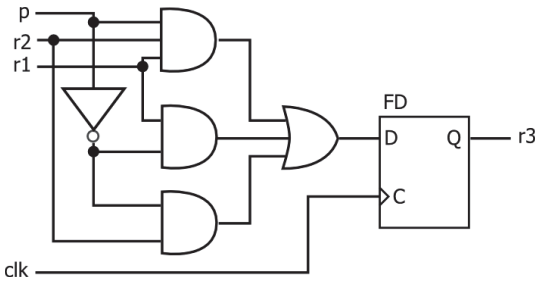
Vezje bo sestavljeno iz logičnih vrat in 3-bitnega registra. Najprej naredimo enostavnejše vezje, ki opravlja operacijo nad enobitnimi signali in shranjuje rezultat v flip-flop D. To vezje bo predstavljalo eno izmed treh celic končnega vezja in ga imenujemo *registrska celica*. Delovanje logike opišemo z enačbo:

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

Kombinacijski izraz je sestavljen iz dveh delov. Kadar je krmilni signal p enak 0, bo vrednost prvega oklepaja 0, drugega pa $r1$ OR $r2$. Podoben razmislek naredimo pri vrednosti krmilnega

vhoda 1. Enačbo še preuredimo, da bo primerna za izvedbo s produktnimi členi programirljive matrike:

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



Slika 7.12: Shema registrske celice logične računske enote.

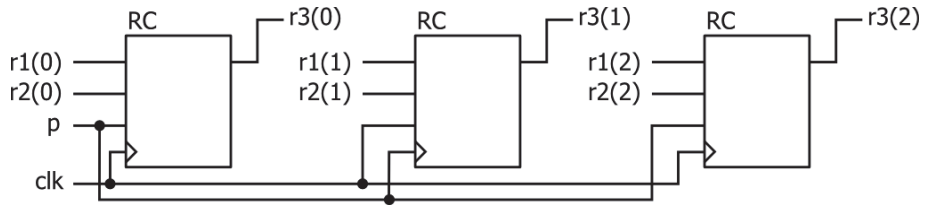
Z *vzporedno* vezavo treh registrskih celic dobimo končno blokovno shemo vezja, kot prikazuje slika 7.13. Delovanje vezja lahko opišemo tudi v jeziku VHDL. Sestavljen je iz deklaracije priključkov (*port*) in arhitekturnega dela, v katerem je obnašanje vezja opisano s pogojnimi stavki (*if*). Programska oprema iz tega opisa sintetizira logično shemo vezja.

Listing 7.1: Opis logične računske enote v jeziku VHDL.

```
entity vezje is
    Port ( clk, p : in  std_logic;
           r1, r2 : in  std_logic_vector(2 downto 0);
           r3     : out  std_logic_vector(2 downto 0));
end vezje;
architecture opis of vezje is
begin
 p: process(clk)
 begin
   if rising_edge(clk) then
     if p='1' then
       r3 <= r1 and r2;
     else
       r3 <= r1 or r2;
     end if;
   end if;
 end process;
end opis;
```

Preslikava v CPLD

Kombinacijski del vezja naredimo s programirljivo matriko. Če primerjamo shemo registrske celice s strukturo programirljive naprave CPLD, ugotovimo, da opisano celico izvedemo z eno



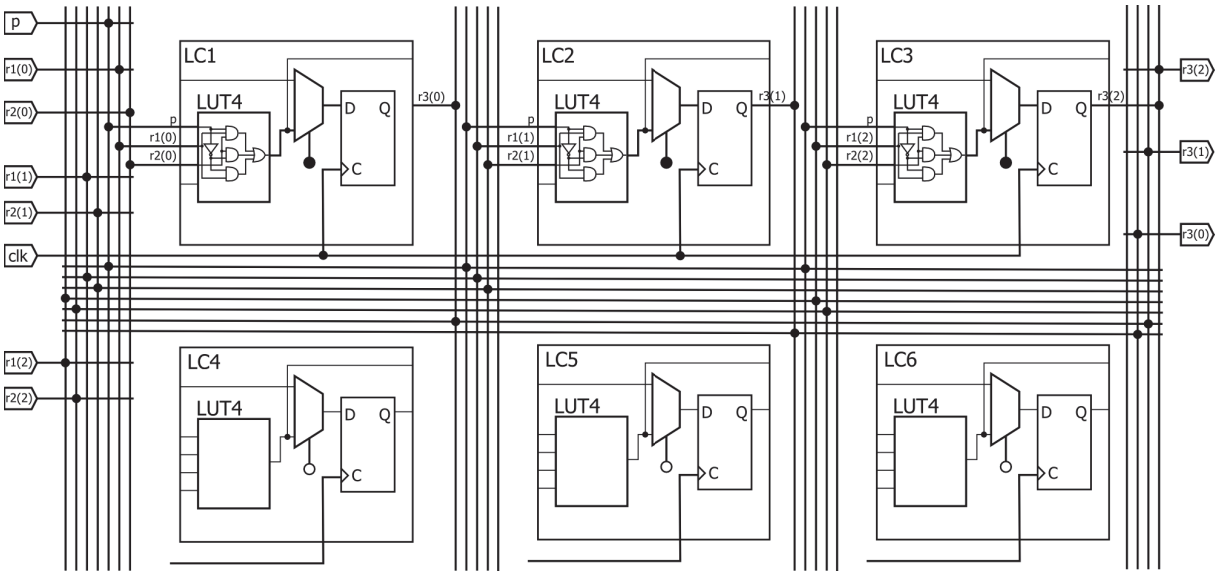
Slika 7.13: Shematski opis 3-bitne logične računske enote.

makrocelico. Celotno vezje naredimo s tremi makrocelicami vezja CPLD, v katerih morajo biti ustrezno nastavljene matrike PLA in podatkovne poti prek flip-flopov. Vzporedno vezavo vhodnih in krmilnih signalov izvedemo v polju povezav.

Naloga programske opreme za tehnološko preslikavo vezja v strukturo CPLD je pripraviti ustrezne oblike (AND–OR) zapisa logičnih izrazov ter dodeliti makrocelice in povezave v polju. Če bi imeli kompleksnejše kombinacijske izraze, bi jih morala programska oprema razdeliti na manjše dele, ki jih lahko preslika v posamezno makrocelico. Sekvenčnim gradnikom dodeli toliko makrocelic, kolikor je vseh flip-flopov v vezju.

Preslikava v FPGA

Za izvedbo s programirljivim vezjem FPGA je treba shemo vezja preslikati v logične celice. Kombinacijsko logiko posamezne registrske celice preslikamo v vpogledno tabelo (LUT), pri kateri uporabimo tri vhode, izhod pa vežemo na flip-flop v logični celici. Vidimo, da bo tudi izvedba celotnega vezja v FPGA zasedla tri logične celice v matriki. Povezave vhodnih in izhodnih signalov pa naredimo s poljem povezav, kot prikazuje slika 7.14.

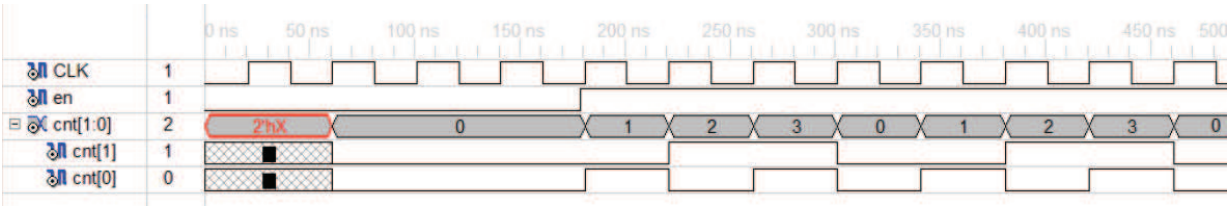


Slika 7.14: Izvedba vezja v FPGA.

Verifikacija

Računalniška simulacija modela vezja je osnovni postopek preverjanja oz. verifikacije vezja. Simulacijo pripravimo tako, da določimo časovni potek spreminjanja signalov na vhodu vezja. Časovni potek vhodnih signalov določimo v testni strukturi (angl. test bench) in je zapisan v grafični obliki, v jeziku HDL ali pa v obliki simulacijskih makrojev. Ko je testna struktura pripravljena, poženemo simulator in pregledamo rezultate na časovnem diagramu signalov (angl. waveform).

Slika 7.15 prikazuje časovni diagram simulacije 2-bitnega števca. V testni strukturi smo nastavili uro in signal en, opazujemo pa vrednost izhoda cnt. Večbitne signale, kot je $cnt(1:0)$, lahko opazujemo v obliki dvojiških, desetiških ali šestnajstiških številskih vrednosti, možen pa je tudi prikaz posameznih bitov $cnt(1)$ in $cnt(0)$.

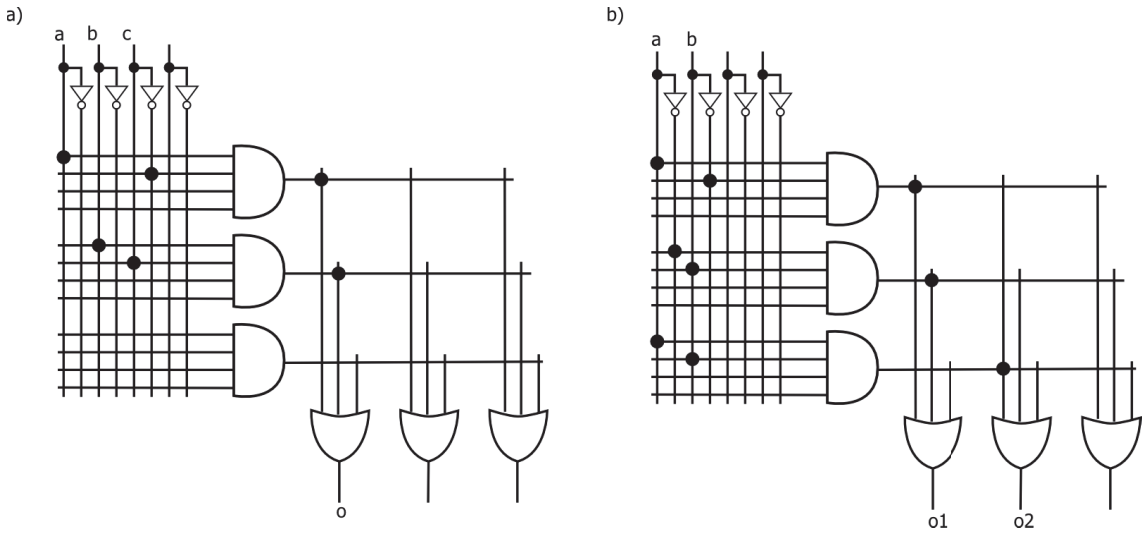


Slika 7.15: Rezultat simulacije 2-bitnega števca.

S pregledom časovnega diagrama ugotovimo, ali se vezje obnaša tako, kot smo pričakovali. V testni strukturi lahko tudi vnaprej predpišemo pričakovane vrednosti izhodov ali pa določimo pravila spreminjanja signalov in tako avtomatiziramo postopek verifikacije. Če želimo z verifikacijo res preveriti delovanje vezja, moramo dobro poznati signale, ki jih lahko pričakujemo na vhodu realnega vezja, in pripraviti testne strukture za veliko različnih primerov.

Naloga

1. Zapiši funkcijo logičnih vezij, ki sta narejeni s programirljivo matriko PLA.



a) o = _____

b) o1 = _____

o2 = _____