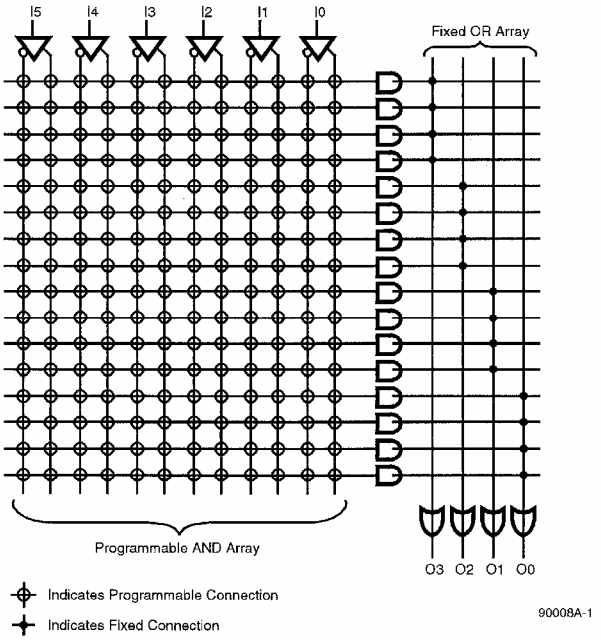
3. Programabilna logička polja (PAL sklopovi)

3.1 Osnove PAL sklopova

3.1.1 Uvod

Digitalni sklopovi poznati pod nazivom PAL vrlo su česti sastavni elementi u ugrađenim računarskim sustavima. Naziv PAL je skraćenica imena ovoga sklopa na engleskom jeziku, a to je **P**rogrammable **A**rray **L**ogic. Prevedeno na hrvatski jezik PAL predstavlja programabilno logičko polje (matricu), odnosno digitalni sklop koji korisnik može konfigurirati ("programirati") za obavljanje određene logičke funkcije. Drugim riječima, krajnji korisnik bez poznavanja i korištenja tehnoloških postupaka izrade digitalnog sklopovlja upisuje svoju funkciju u silicij. Često se u literaturi vezano uz programabilne sklopove ovog tipa pojavljuje pojam PLD (*engl.* ***P****rogrammable* ***L****ogic* ***D****evice*) koji obuhvaća nešto širi spektar sklopova sličnih PAL-u. PAL predstavlja jednu podgrupu PLD sklopova.

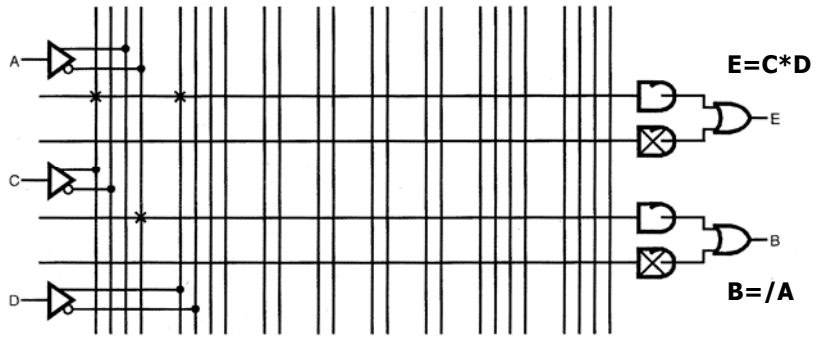
Tehnologija PROM (*engl. Programmable Read Only Memory*) sklopova postavila je temelje za razvoj tehnologije za izradu PAL-ova. Standardni PAL sklopovi sastoje se od programabilne I (*engl. AND*) matrice iza koje se nalazi fiksna ILI (*engl. OR*) matrica. Na slici 3.1 prikazana je arhitektura PAL sklopa.



Slika 3.1 Arhitektura PAL sklopova

Ulazne linije PAL-a se dovode na stupce I matrice, te se u recima ostvaruju željene I logičke funkcije, odnosno generiraju tzv. produktni izrazi. Produktni izrazi (*engl. product terms*) predstavljaju I funkcije različitih ulaza ili njihovih invertiranih varijanti, tj. jedna I vrata. Naziv produktni izrazi dolazi od činjenice da se u Bool-ovoj algebri I funkcija često radi lakšeg zapisa označava kao produkt logičkih varijabli, dok se ILI funkcija označava kao suma logičkih varijabli. Produktni izrazi dovode se na ulaze u ILI matricu gdje se kombiniraju da bi se dobili željeni izlazni signali iz PAL-a.

Za rad s PAL-ovima nije nužno potrebno znati kako se pojedine funkcije implementiraju u PAL-u. Da bi se ipak zaokružila slika o ovom tipu digitalnog sklopovlja, na slici 3.2 dan je prikaz implementacije dvaju funkcija: E = C*D (I vrata) i B=/A (inverter). Svi faktori koji mogu ući u jedan produktni izraz prikazani su u obliku vertikalnih linija. Jedan produktni izraz predstavljen je jednom horizontalnom linijom pri čemu simbol X označava mjesta gdje postoji spoj u matrici. Nekorišteni produktni izrazi su u ovom primjeru prekriženi.



Slika 3.2 Prikaz realizacije jednostavnih funkcija

PAL sklopovi koji se sastoje samo od I i ILI matrica nazivaju se kombinacionim PAL-ovima. Postoji i drugi tip PAL-ova kojima se osim kombinacionih funkcija mogu ostvarivati i sekvencijalne funkcije (npr. brojila) zbog toga što osim logičkih matrica sadrže i određeni broj bistabila. Takvi PAL-ovi se često nazivaju registarskim. Svaki bistabil uz još neke dodatne elemente koji definiraju izlaznu funkciju čini makroćeliju.

Složenost i pogodnost pojedinih tipova PAL-ova za ostvarenje neke logičke funkcije može se analizirati na temelju slijedećih parametara: broj ulaznih linija, broj izlaznih linija, maksimalni broj produktnih izraza po jednom ILI sklopu, broj bistabila, tip bistabila (obični D-bistabil, bistabil sa asinkronom set i reset funkcijom), izvor takta za bistabile (zajednički za sve bistabile u PAL-u ili programabilan za svaki pojedini), način upravljanja izlaza bistabila (zajednički za sve bistabile u PAL-u ili programabilan za svaki pojedini) itd. Izuzetno važni parametri za odabir PAL-a su i njegova brzina te potrošnja. Brzina i potrošnja u velikoj mjeri ovise o tehnologiji izrade PAL-a.

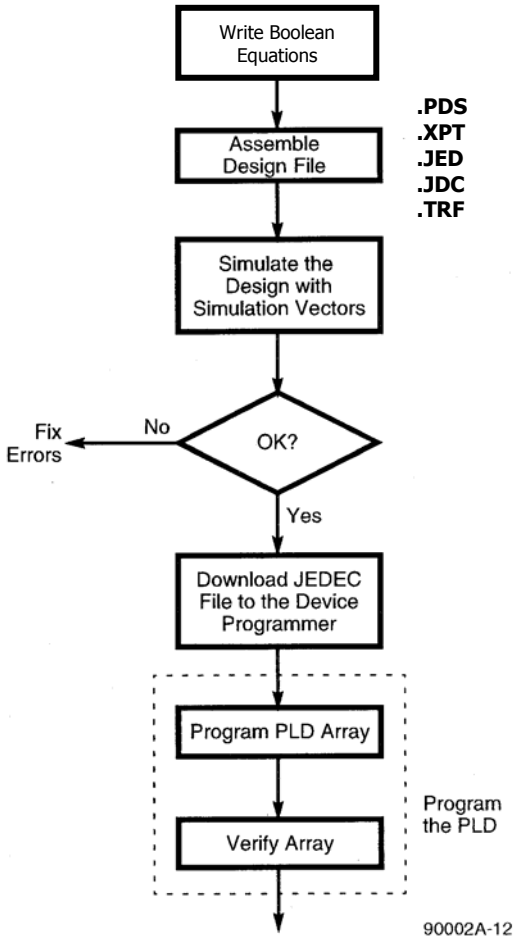
Postoji više tehnologija za izradu PAL-ova odnosno više načina ostvarivanja spojeva u matricama. U ECL i bipolarnoj tehnologiji spojevi se ostvaruju pomoću posebnih elemenata tzv. osigurača (*engl. fuse*), u UV-EPROM (brisanje pomoću ultraljubičaste svjetlosti) i EEPROM CMOS (*engl. Electricaly Eraseble*) tehnologijama je osnovica memorijska ćelija tipa E/EEPROM, te kod CMOS RAM (*engl. Random Access Memory*) tehnologije osnovica je RAM ćelija. U pravilu ECL i bipolarna tehnologija rezultiraju bržim sklopovima koji više troše. Nasuprot tome, CMOS sklopovi odlikuju se malom potrošnjom, ali im je vrijeme propagacije s ulaza na izlaz nešto veće. Redovito se PAL-ovi iste unutrašnje strukture izrađuju u više različitih varijanti brzine i potrošnje.

EEPROM CMOS PAL-ove odlikuje još jedno važno svojstvo, a to je mogućnost električkog brisanja. Ono se izvodi u trajanju od nekoliko sekundi za razliku od UV-EPROM tipa gdje je postupak brisanja dugotrajan i nespretan. Kod bipolarnih PAL-ova brisanje i reprogramiranje uopće nije moguće.

3.1.2 Projektiranje PAL sklopova

Ostvarivanje potrebne logičke funkcije u PAL-u podrazumijeva korištenje odgovarajuće programske podrške i uređaja za programiranje PAL-a. Programska podrška omogućuje unos logičke funkcije, simulaciju i generiranje izlazne datoteke s podacima za programiranje. Na laboratorijskim vježbama koristit će se programska paket PALASM, a njegova upotreba će biti objašnjena u nastavku u poglavljima 3.2 i 3.3.

Postupak projektiranja PAL-a prikazan je na slici 3.3. Unos funkcije rezultira kreiranjem datoteke pod imenom 'ime.PDS'. Pozivom prevodioca logička funkcija napisana u tekstualnom obliku se najprije minimizira po pravilima Boole-ove algebre, zatim prilagođuje unutarnjoj arhitekturi PAL-a te na kraju implementira. Implementacija u stvari predstavlja označavanje mjesta u matrici i na dodatnim elementima PAL makroćelija koja je programiranjem potrebno prospojiti. Vizualni prikaz krajnjeg izgleda unutrašnjosti projektiranog PAL-a dan je u datoteci 'ime.XPT'. Izlazna datoteka koja se koristi za stvarno programiranje je 'ime.JED'. Simulacija predstavlja važni korak u projektiranju, a njen je cilj ispitati da li se izlazne varijable ponašaju kako je predviđeno uz variranje vrijednosti na ulaznim pinovima. Simulacija rezultira 'ime.TRF' datotekom, a moguće je simulaciju pogledati bilo u obliku logičkih stanja, bilo u obliku vremenskih dijagrama. Nakon provedenog prevođenja, primjena simulacije formira još jednu izlaznu datoteku s ekstenzijom '.JDC' koja pored konfiguracije PAL-a sadrži i pobudne i odzivne sekvence za provjeru funkcionalnosti (tzv. test vektori).

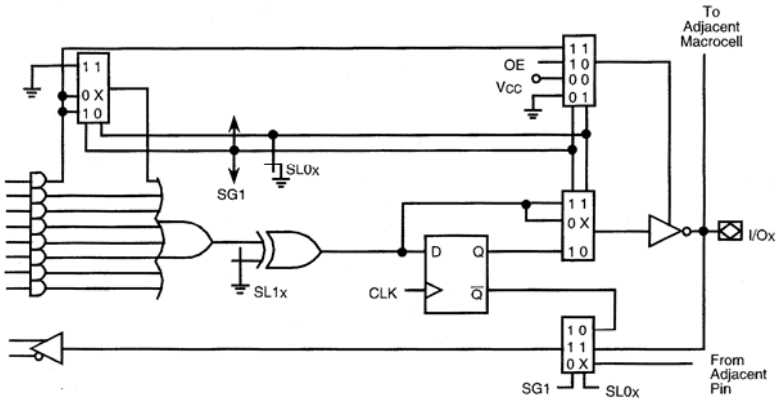


Slika 3.3 Postupak projektiranja PAL-a

Programiranje PAL-ova u okviru ovih laboratorijskih vježbi izvodi se uređajem koji se zove univerzalni programator kojim se osim PAL sklopova može programirati i sklopove tipa PROM, EPROM, EEPROM i druge. Tok programiranja je slijedeći. Najprije treba odabrati tip i proizvođača PAL sklopa. U slučaju da se radi o reprogramabilnom PAL sklopu potrebno je PAL izbrisati i provjeriti da li je to uspješno izvedeno (*engl. blank check*). Nadalje, .JED datoteku treba učitati u privremeni međuspremnik te pokrenuti postupak programiranja. Tokom programiranja automatski se izvodi i provjera da li je sadržaj .JED datoteke ispravno zapisan u PAL. Postupak projektiranja PAL-a time je završen. Korištenjem .JDC datoteke moguće je na nekim PAL programatorima nakon programiranja PAL-a provesti i provjeru funkcionalnosti pobudom sa odabranim test vektorima te usporedbom stvarnog i očekivanog odziva PAL sklopa.

3.1.3 Sklop PALCE16V8

Na laboratorijskim vježbama radit će se sa sklopom PALCE16V8. To je CMOS PAL s električkim brisanjem, a takvi PAL-ovi poznati su i pod nazivom GAL pa je čest naziv ovog PAL-a još i GAL16V8. Ovaj PAL je registarski, s makroćelijom koja je prikazana na slici 3.4, a ima 8 izlaza te maksimalno 7-8 mogućih produktnih izraza po ILI funkciji. PAL je sinkroni, tj. ulaz signala takta zajednički je za sve bistabile.



Slika 3.4 Makroćelija sklopa PALCE16V8

Na ovom PAL-u svakom izlaznom pinu pripada po jedna makroćelija. Ta se makroćelija može konfigurirati na jedan od slijedećih načina: kao registarski izlaz, kombinacioni izlaz, kombinacioni ulazno/izlazni pin ili kao obični ulazni pin. Prikaz mogućih konfiguracija makroćelije dan je na slici 3.5. Konfiguracija makroćelije ovisi o kontrolnim linijama SG0, SG1, te $SL0_0$-$SL0_7$ i $L0_0$-$SL0_7$, koje se tokom programiranja postavljaju u određena stanja ovisno o realizaciji logičke funkcije i tipu dotičnog pina.

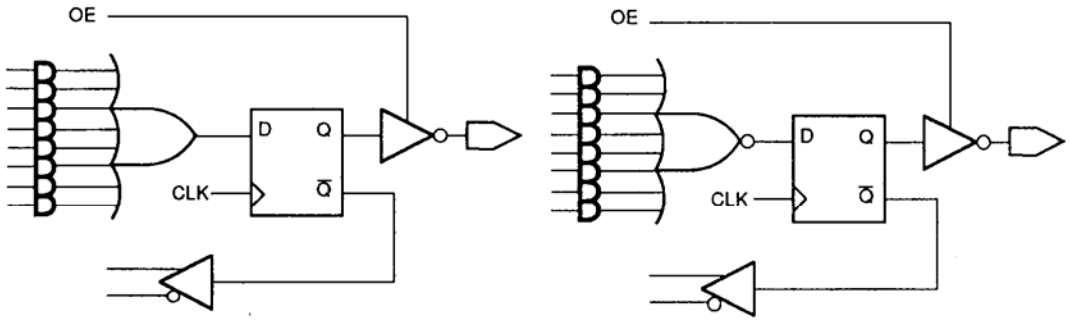
Konfiguracija prikazana na slici 3.5a) je registarska konfiguracija. U ILI funkciji mogu se koristiti svih osam produktnih izraza, a polaritet izlaznog signala se može odabrati po želji. Bistabil poprima novo stanje na rastući brid signala takta CLK, a izlazni pogonski sklop s tri stanja omogućuje se ulaznim pinom $\overline{OE}$. Izlaz bistabila $\overline{Q}$ vraća se natrag u I-ILI matricu za ostvarenje logičkih funkcija kod kojih je potrebna i povratna veza.

Pinove PAL-a moguće je deklarirati kao dvosmjerne (ulazno/izlazne), kao što je prikazano na slici 3.5b). U tom slučaju se u ILI funkciji može koristiti samo 7 produktnih izraza dok je osmi iskorišten za omogućavanje izlaznog pogonskog sklopa s tri stanja.

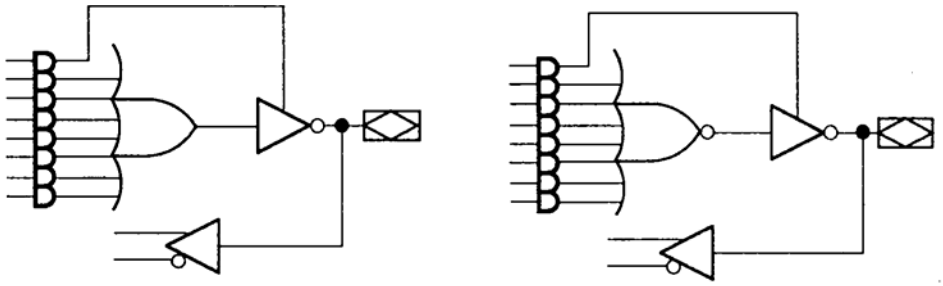
U konfiguraciji kombinacionog izlaza, slika 3.5c), ponovno se koriste svih osam mogućih produktnih izraza. CLK i $\overline{OE}$ se ne koriste pa se također mogu deklarirati kao ulazni kombinacioni pinovi. Povratna veza sa $\overline{Q}$ pina ne postoji, a pogonski sklop s tri stanja je stalno omogućen.

Pinovi mogu biti i čisti ulazni, slika 3.5d), pri čemu je pogonski sklop s tri stanja stalno onemogućen, a povratna veza se koristi kao ulaz za pripadni I/O pin.

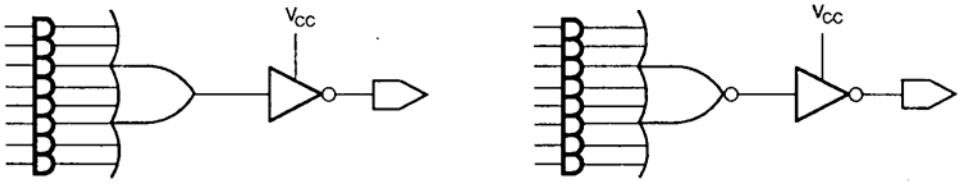
a) registarska konfiguracija, izlaz aktivan nisko ili visoko



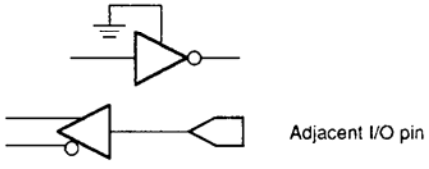
b) kombinaciona ulazno/izlazna konfiguracija, izlaz aktivan nisko ili visoko



c) kombinacioni izlaz, izlaz aktivan nisko ili visoko



d) kombinacioni ulaz



Slika 3.5 Moguće konfiguracije makroćelije

3.2 Osnove projektiranja PAL-ova programom PALASM

U ovom dodatku su opisani osnovni postupci i naredbe koje se koriste kod projektiranja PAL-ova programskim paketom PALASM. Postupci su dani onim redom kojim bi se trebali i koristiti, dakle pokretanje programa, unošenje zaglavlja i logičkog opisa funkcije u novu datoteku ili mijenjanje već postojeće datoteke, prevođenje dizajna, simulacija, te evaluacija dobivenih rezultata i ispravljanje eventualnih pogrešaka.

3.2.1 Pokretanje programa PALASM

Program **PALASM** pokreće se tipkanjem PALASM na komandnoj liniji DOS prozora kojeg prije toga treba pokrenuti. Program **PALASM** po ulasku aktivira meni **FILE** koji služi za manipulacije s datotekama, postavljanje direktorija te podešavanje parametara radne okoline.

3.2.2 Unos zaglavlja dizajna

Novi dizajn PAL-a počinje aktiviranjem naredbe **BEGIN NEW DESIGN** te odabirom **TEXT** formata i upisom imena nove datoteke. Unos se završava pritiskom na funkcijsku tipku F10 nakon čega se automatski otvara forma za unos podataka vezanih uz novi dizajn tzv. dio za deklaraciju (**DECLARATION SEGMENT**). Dio za deklaraciju dobro je u potpunosti popuniti zbog bolje dokumentiranosti dizajna, a sadrži slijedeće podatke (Tabela 3.1):

Tabela 3.1 Ključne riječi u dijelu za deklaraciju i njihov opis

Ključna riječ	Opis i korištenje
TITLE	naziv dizajna (što predstavlja)
PATTERN	kratko ime dizajna
REVISION	verzija dizajna
AUTHOR	autor dizajna
COMPANY	ustanova
DATE	datum izrade
CHIP_NAME	ime sklopa, vezano uz ime .PDS datoteke
DEVICE	tip sklopa koji treba obavezno unijeti; dolaskom na polje automatski se pojavljuje meni sa kojeg se odabere odgovarajući tip PAL-a.

Nakon odabira tipa sklopa dolazi dio deklaracije pinova (**PIN DECLARATIONS**) gdje se definira tip svakog korištenog pina PAL-a. Opis je dan u Tabeli 3.2.

3.2.3 Unos logičkih izraza

Nakon opisa sklopa i pinova automatski se ulazi u tekstualni editor u kojem se dalje radi unos logičkih izraza koji će definirati logičku funkciju PAL-a. To se izvodi u dijelu za Boole-ove izraze (**BOOLEAN EQUATION SEGMENT**) poslije ključne riječi **EQUATIONS** koja mora postojati.

Zadnji segment u datoteci predstavlja simulacijski dio (**SIMULATION SEGMENT**) gdje se iza ključne riječi **SIMULATION** koja mora postojati pišu izrazi vezani uz simulaciju.

Nakon unesenih promjena u tekstualnom editoru se pritiskom na tipke ALT i F otvara meni za manipulaciju s datotekama te se nova datoteka treba pohraniti jer će u protivnom promjene biti izgubljene. Izlazak iz tekstualnog editora te povratak u PALASM program ostvaruje se pritiskom na tipke ALT i X.

Tabela 3.2 Ključne riječi u dijelu za deklaraciju pinova i njihov opis

Ključna riječ	Opis i korištenje
P/N	za PALCE16V8 za sve linije odabrati PIN
NUMBER	fizički broj pina na sklopu; ne moraju se unositi po redu i ne treba unositi one koji se ne koriste; treba unijeti pinove za napajanje i masu pod imenima VCC i GND; moguće unijeti niz pinova u jednoj naredbi poput vektora npr. PIN 6,10..8 DATA[3..0] ili PIN 12..18 A[1..7]
NAME	logičko ime pina može biti alfanumeričko do 14 znakova; koristiti samo '_' i '/' od specijalnih znakova; znak / (negacija) prije imena pina koristi se ako se želi definirati da će pripadna linija biti aktivna nisko odnosno u stanju logičke 0
PAIRED WITH PIN	ostaviti prazno (nije podržano ovim tipom PAL-a)
STORAGE	tip pina koji može biti **COMBINATORIAL**, **REGISTERED**, **LATCHED** ili prazno polje, a odabire se pritiskom na tipku F2; za pinove VCC i GND ostaviti prazno polje
COMMENT	a) znak ; inače označava komentar iza kojeg može slijediti bilo što jer će biti ignorirano od strane prevodioca; u slučaju deklaracije pinova se radi lakšeg snalaženja unosi tip pina INPUT, OUTPUT, IO ili prazno polje; za VCC i GND ovo polje ostaviti prazno

3.2.4 Mijenjanje postojećeg dizajna

Mijenjanje postojećeg dizajna ostvaruje se pokretanjem naredbe **RETRIEVE EXISTING DESIGN** (otvori postojeći dizajn) pod **FILE** meni opcijom te odabirom **TEXT** formata i upisom imena postojeće datoteke. Unos se završava pritiskom na funkcijsku tipku F10. Nakon toga se kursorima treba pomaknuti na **EDIT** meni te odabrati opciju **TEXT FILE**, čime se automatski pokreće tekstualni editor, u ovom slučaju će to biti **EDIT.COM** .

Meniji tekstualnog editora **EDIT** aktiviraju se pritiskom na tipku ALT i slova koje je bijele boje u riječima na vrhu ekrana. Nakon unesenih promjena u tekstualnom editoru se pritiskom na tipke ALT i F otvara meni za manipulaciju s datotekama te se nova datoteka treba pohraniti. Izlazak iz tekstualnog editora te povratak u PALASM program ostvaruje se pritiskom na tipke ALT i X.

3.2.5 Prevođenje dizajna

Nakon završenog unosa potrebnih logičkih izraza treba provesti prevođenje u format pogodan za izlaznu datoteku. Prevođenje se izvodi odabirom menija RUN te opcije **COMPILATION** nakon čega se prihvaćanjem opcija pritiskom na tipku F10 pokreće

prevođenje. Ako se želi izvesti i prevođenje i simulacija s menija RUN odabire se opcija **BOTH**. Nakon toga treba dva puta pritisnuti F10 da bi se proces pokrenuo.

3.2.6 Simulacija

Nakon unošenja simulacijskih test vektora upisom u SIMULATION SEGMENT datoteke s dizajnom, simulacija se pokreće odabirom menija RUN te opcije **SIMULATION** nakon čega se prihvaćanjem opcija pritiskom na tipku F10 pokreće simulacija. Ako se želi izvesti i prevođenje i simulacija s menija RUN odabire se opcija BOTH. Nakon toga treba dva puta pritisnuti F10 da bi se proces pokrenuo.

3.2.7 Ispravljanje pogrešaka u dizajnu i simulaciji

Svi postupci i rezultati dobiveni tokom prevođenja i simulacije spremaju su u datoteku **PALASM.LOG** koja ostaje vidljiva na ekranu nakon završetka bilo prevođenja bilo simulacije, a koju se može i naknadno pogledati biranjem opcije **EXECUTION LOG FILE** pod menijem **VIEW**. Treba obratiti pažnju na retke datoteke u kojima se navodi broj pogrešaka (ERROR COUNT) te upozorenja (WARNING COUNT) koji rezultiraju nakon određenog faze pojedinog postupka. Pogreške kod prevođenja uglavnom ukazuju na pogreške u sintaksi dok pogreške u simulaciji ukazuju na neispravnost ostvarene logičke funkcije. Da bi konačni dizajn PAL-a bio ispravan, sve pogreške i upozorenja bi trebalo otkloniti provjerom napisane logičke funkcije, deklaracije pinova, simulacijskih vektora i izlaznih podataka iz simulacije.

3.2.8 Pregled rezultata dizajna

Izborom menija VIEW može se biranjem odgovarajuće opcije pregledati izgled rezultirajućih datoteka za jedan dizajn. Biranjem opcije **DESIGN FILE** se može pogledati tekuća datoteka sa dizajnom, ali se ne može mijenjati. Opcija **REPORTS, FUSE MAP** daje prikaz datoteke **ime.XPT** sa izgledom ostvarenih spojeva u unutrašnjosti PAL-a označenih simbolom X. Opcija **JEDEC DATA** daje prikaz izlaznih **.JED** datoteka (bez simulacijskih vektora) ili **.JDC** datoteka (simulacijski vektori uključeni). Pregled podataka dobivenih simulacijom ostvaruje se opcijama **SIMULATION DATA** (podaci dani u obliku L i H) ili **WAVEFORM DISPLAY** (valni oblici) pri čemu dodatna opcija **HISTORY** uključuje sve definirane pinove na PAL-u prema fizičkom redoslijedu pinova, a opcija **TRACE** rezultira prikazom samo pinova navedenih u simulaciji prema navedenom redoslijedu. Opcija **PINOUT** daje prikaz sklopa sa fizičkim rasporedom pinova, njihovim nazivima i tipovima.

3.2.9 Pomoć

Pomoć pri radu s pojedinim naredbama i ključnim riječima, te kod pojave grešaka može se dobiti ulaskom u meni **DOCUMENTATION** i odabirom odgovarajuće opcije.

3.3 Sintaksa za pisanje logičkih izraza i simulaciju u PALASM programu

Za ostvarivanje logičkih funkcija PAL-om korištenjem PALASM programa, koriste se određene naredbe i ključne riječi koje za program imaju neko značenje. Te ključne riječi mogu se podijeliti u više skupina: one koje se koriste za pisanje logičke funkcije, prikazane u tabeli 3.3, one koje se koriste kod simulacije (tabela 3.4) i druge.

Tabela 3.3 Ključne riječi za pisanje logičkih funkcija

DIZAJN POMOĆU FUNKCIJA BOOL-OVE ALGEBRE		
sintaksa	**značenje**	**primjer**
=	operator pridruživanja; izrazu navedenom s lijeve strane operatora pridružuje se bilo koja kombinacija ulaza s desne strane; vrijedi za kombinacione, registarske i lached linije	A=B
/	operator negiranja (**NOT**)	A=/B
*	I operator (**AND**)	C=A*B
+	ILI operator (**OR**)	C=A+B
:+:	operator EKSKLUZIVNI ILI (**XOR**)	C=A:+:B
:*:	operator EKSKLUZIVNI NILI (**XNOR**)	C=A:*:B
;	komentar	; ovo ne ulazi u kod
..	raspon (engl. range) u slučaju korištenja vektorskog zapisa	INPUT[0..5]
,	separator kod nabrajanja	IN[1,3,4,..6]
:	vrijednost kod CASE naredbe (vidi u nastavku)	#b1010:
[ ]	vektorski prikaz	IN[2]
( )	izraz	(A*B)(C+D)
{ }	zamjena za izraz	IN=A*B OUT={IN}+C
>	veće od …	IF A>2 THEN …
>=	veće ili jednako	IF A>=2 THEN …
<	manje od …	WHILE B<3 DO …
<=	manje ili jednako	IF A<=2 THEN …
<>	različito od …	IF A<>2 THEN …
‘ ‘	oznake za string	STRING INPUT ‘A+/B’
#d ili #D	decimalni zapis	#d40

DIZAJN POMOĆU FUNKCIJA BOOL-OVE ALGEBRE (nastavak)		
sintaksa	**značenje**	**primjer**
#b ili #B	binarni zapis	#b0101
#o ili #O	oktalni zapis	#O14
#h ili #H	heksadecimalni zapis	#H1f
	prazno polje (separator)	PIN12 UL1 RE
VCC	simbol za logičku jedinicu	OUT=VCC
GND	simbol za logičku nulu	B3=0
GROUP	omogućava grupiranje više pinova pod jednim imenom; sve što se odnosi na grupu, odnosi se na svaki pojedini element grupe; zapisuje se poslije deklaracije pinova, a prije funkcije: GROUP ime_grupe pin1, pin2 …	GROUP SVI X1,B1
STRING	omogućuje objedinjavanje nekog niza znakova jednim imenom	STRING S1 'A=B+C' SEL1 = S1
.TRST	rezervirana riječ u funkcijskom izrazu koja definira aktivno stanje driver-a s tri stanja na pripadnim izlaznim pinovima	Q1.TRST = A*/B

CASE (signali uvjeta) BEGIN vrijednost1: BEGIN izraz1 END vrijednost2: BEGIN izraz2 END OTHERWISE: BEGIN izraz3 END END	b) CASE struktura omogućuje uvjetnu provjeru signala u odnosu na više konstantnih vrijednosti koje signali uvjeta mogu poprimiti. Ovisno o vrijednosti, izvršavaju se pripadni izrazi. One vrijednosti koje nisu navedene u CASE naredbi smatraju se ostalim vrijednostima i spadaju pod dio OTHERWISE.	**CASE** (A,D) BEGIN 0: BEGIN C=A*B END 3: BEGIN C=A+B END OTHERWISE: BEGIN C=0 END END

IF (uvjet) **THEN**
BEGIN
izraz1
END
ELSE
BEGIN
izraz2
END
	IF-THEN-ELSE struktura omogućuje uvjetno izvođenje Boole-ovih izraza u smislu: ako je uvjet istinit, izvodi se izraz1, a ako ne, izvodi se izraz2	**IF** (/I1,/I2=#b11) **THEN**
		BEGIN
		O3=A*B
		END
		ELSE
		BEGIN
		O3=A+B
		END

Tabela 3.4 Sintaksa programa PALASM vezana uz simulaciju

<table>
<tr><th colspan="3">SINTAKSA VEZANA UZ PROCES SIMULACIJE</th></tr>
<tr><th>sintaksa</th><th colspan="2">značenje</th><th>primjer</th></tr>
<tr><td>TRACE_ON</td><td colspan="2">definira koje signale (pinove) treba pratiti tokom simulacije; moguće pregrupiranje i poredak signala po želji; moguće definirati polaritet signala;
TRACE_ON ime_pina1, ime_pina2, …</td><td rowspan="5">SIMULATION
TRACE_ON A /B CLK
SETF /A /CLK
SETF CLK
CHECK B
TRACE_OFF</td></tr>
<tr><td>TRACE_OFF</td><td colspan="2">označava kraj simulacijskog segmenta koji je počeo s TRACE_ON</td></tr>
<tr><td>SETF</td><td colspan="2">dodjeljuje vrijednosti pojedinim ulazima tokom simulacije; ulazi mogu biti sa ili bez prefiksa tj. znaka negacije (/)
SETF (prefix)pin1, (prefix)pin2 ..</td></tr>
<tr><td>CHECK</td><td colspan="2">naredba kojom se verificira da je vrijednost signala na pinu onakva kakva bi trebala biti; prefix odgovara logičkom stanju na pinu, a može biti nul-prefix (log. jedinica), / (log. nula), ^ stanje visoke impedancije, % bilo koje stanje
CHECK (prefix)pin1, (prefix)pin2 ….</td></tr>
<tr><td>CHECKQ</td><td colspan="2">naredba kojom se verificira da je vrijednost signala na izlazu iz registra onakva kakva bi trebala biti; prefix odgovara logičkom stanju, a može biti bez prefixa (log. jedinica) i / (log. nula)
CHECKQ (prefix)pin1, (prefix)pin2 ….</td></tr>
<tr><td colspan="2">FOR var := start TO end DO
BEGIN
izrazi
END</td><td>FOR petlja kojom se definira koliko puta će se izvesti blok naredbi za simulaciju unutar BEGIN i END; start i end su početna i završna vrijednost varijable var</td><td>TRACE_ON CLK
COUNT
SETF /CK
FOR X := 1 TO 20 DO
BEGIN
SETF CLK
SETF /CLK
END</td></tr>
</table>

IF (uvjet) **THEN**
BEGIN
zadatak1
END
ELSE
BEGIN
zadatak2
END
	IF-THEN-ELSE struktura omogućuje uvjetno izvođenje tokom simulacije u smislu: ako je uvjet istinit, napravi ovo, a ako ne, napravi ono	**IF** (J = 5) **THEN**
		BEGIN
		CHECK Q0
		END
		ELSE
		BEGIN
		CHECK /Q0
		END

WHILE (uvjet) **DO**
BEGIN
zadatak
END
	WHILE-DO petlja predstavlja strukturu koja provjerava logičko stanje uvjeta i izvodi zadatak dok je uvjet ispunjen	SETF /CK COUNT
		WHILE(/BIT1* BIT2)
		DO
		BEGIN
		SETF CLK
		SETF /CLK
		END

3.4 Opis makete korištene na laboratorijskim vježbama

Shema digitalnog dijela makete dana je na slici 3.6, a prikaz štampane pločice PAL makete sa strane komponenata dan je na slici 3.7.

Raspored ulaznih i izlaznih pinova PAL-a te mase i napajanja dan je u Tabeli 3.5. Ulazi na PAL-u (sklop označen sa U202) nalaze se na pinovima od 2 do 9. Na ulaze PAL sklopa od pinova 2 do 8 spojeni su kratkospojnici. Stanje tih ulaza kada kratkospojnik nije spojen je logička 1. Spajanjem pripadnog kratkospojnika se dotični ulaz postavlja u stanje logičke 0.

Tipka X210 spojena je na ulazni pin 9. Također je u stanju logičke 1 kada tipka nije pritisnuta. Pritiskom na tipku ulaz se spaja na GND tj. u stanju je logičke 0.

Pin 1 – CLK je također ulaz na koji je spojen izvor takta. Takt se generira bilo sklopom A201, bilo pritiskom i otpuštanjem mikroprekidača X202. Izvor i frekvencija takta ovisi o grupi kratkospojnika, tj. da li su prospojeni ili ne. U Tabeli 3.6 dan je prikaz ovisnosti izvora i frekvencije generatora takta o prisutnosti kratkospojnika.

Ulazni pin 11 za omogućavanje izlaza, $\overline{OE}$ (*engl. output enable*) je fiksno prospojen na masu, tj. izlazni pogonski sklopovi s tri stanja unutar PAL sklopa su stalno aktivirani, te su izlazi bistabila trajno propušteni na izlazne pinove PAL-a.

Izlazi PAL-a, pinovi 12-19 spojeni su preko pogonskih sklopova i serijskih otpornika na 7-segmentni prikazivač. Ovisno o logičkim stanjima na ulazima te ostvarenoj logičkoj funkciji u PAL-u pojedini segmenti prikazivača se pale odnosno gase.

Tabela 3.5 Raspored ulaznih i izlaznih pinova na PAL-u

Ulazni pinovi na PAL-u	**Izlazni pinovi na PAL-u**
1 – CLK	12
2	13
3	14
4	15
5	16
6	17
7	18
8	19
9	
11 - $\overline{OE}$	
10 – GND	
20 – VCC	

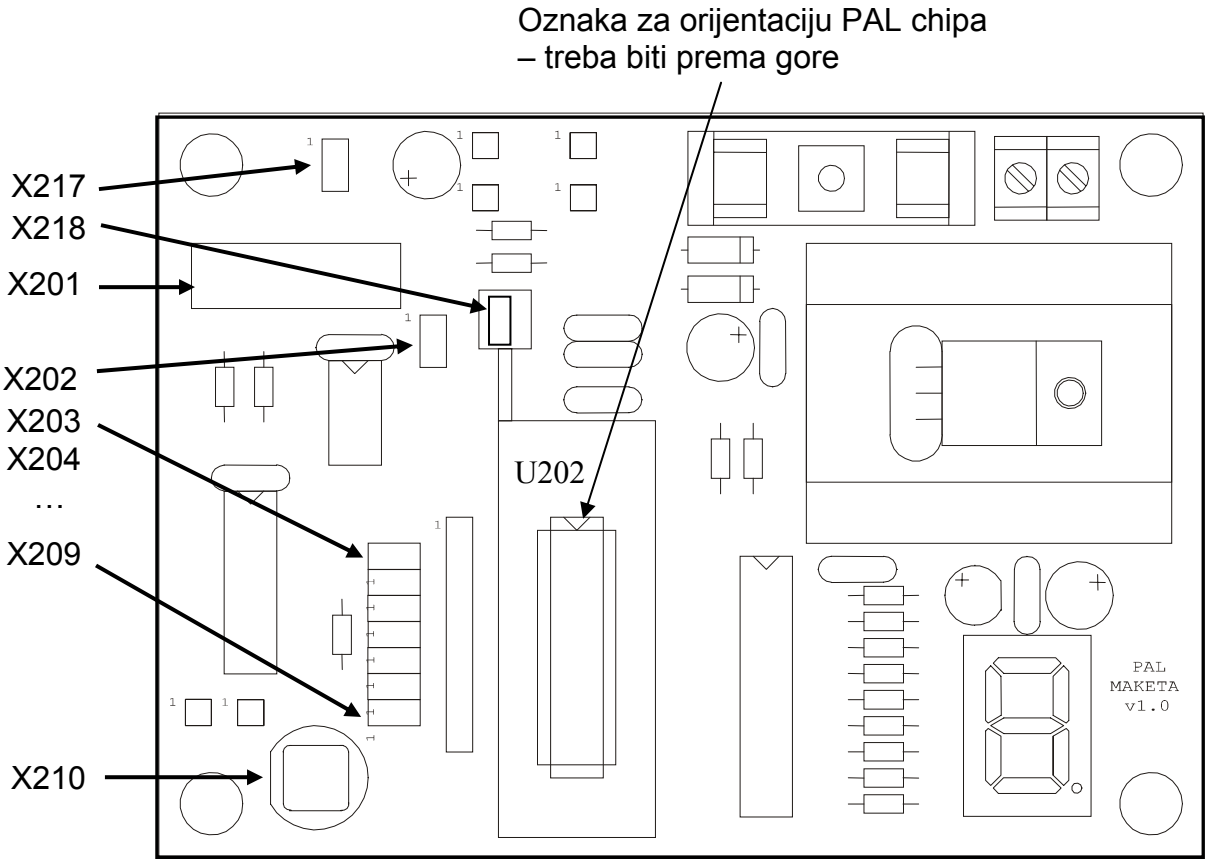
slika 3.6 PCAD shema

Slika 3.6 PCAD shema

Tabela 3.6 Izvor i frekvencija takta ovisno o spoju kratkospojnika

IZVOR TAKTA	KRATKOSPOJNICI		
	X201	X217	X218
Oscilator A201, f = 0,5 Hz	■	□	□
Oscilator A201, f = 5 kHz	□	□	□
Mikroprekidač X202	bilo što	■	■

■ - kratkospojnik spojen
□ - kratkospojnik odspojen



Slika 3.7 Prikaz štampane pločice PAL makete sa strane komponenata

3.5 Vježba 6: Projektiranje kombinacionih i registarskih funkcija PAL sklopom

U slijedećem poglavlju će biti opisano nekoliko primjera realizacije određenih funkcija implementiranih sklopom PALCE16V8. Posebno će biti dani primjeri kombinacionih, a posebno primjeri realizacije registarskih funkcija. Kompletna rješenja primjera (*engl. listing*) dani su u poglavlju 3.6.

3.5.1 Primjeri realizacije kombinacionih funkcija

a) Primjer 1.

Treba realizirati enkoder s 4 ulaza na 2 izlaza. Enkoder je sklop koji na temelju jedne od 2^n ulaznih linija generira podatak na n izlaznih bita (slika 3.8). U Primjeru 1. stanja na 4 ulazne linije bit će enkodirane kao 2-bitni podatak i prikazane na dvije izlazne linije.

Slika 3.8 Enkoder 4/2

b) Rješenje

Postupak dizajna navedene funkcije počinje pisanjem tablice istinitosti (tabela 3.7).

Tabela 3.7 Tablica istinitosti za Primjer 1. (enkoder 4/2)

ULAZI				IZLAZI	
A	B	C	D	C0	C1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

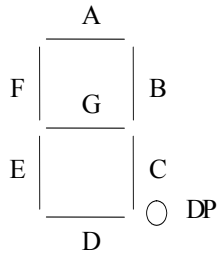
Funkcija koju treba realizirati je čisto kombinaciona. Uvidom u tabelu, identificiraju se stanja logičke 1 za izlaze C0 i C1 te se mogu pisati Boole-ove jednadžbe:

```
C0 = /A * /B *  C * /D
   + /A * /B * /C *  D

C1 = /A *  B * /C * /D
   + /A * /B * /C *  D
```

Ovakva realizacija enkodera pretpostavlja da će u jednom trenutku biti aktivan samo jedan ulaz. U slučaju da su aktivna dva ulaza dogodio bi se konflikt koji ovom realizacijom ne bi bio razriješen.

c) Primjer 2.

Treba realizirati funkciju prikaza 4-bitnih heksadecimalnih brojeva (0-9,A,B,C,D,E,F) na sedamsegmentnom prikazivaču (*engl. display*) koji ima i decimalnu točku kao osmi segment. Raspored segmenata na prikazivaču dan je na slici 3.9. Treba realizirati i funkciju testiranja svih segmenata, tj. da se na zahtjev upale svi segmenti. Također provesti i simulaciju funkcije test vektorima.



Slika 3.9 Sedamsegmentni prikazivač s decimalnom točkom

d) Rješenje

Da bi se ostvarila funkcija prikaza broja potrebne su 4 ulazne linije, nazovimo ih D0, D1, D2, D3, gdje je D[0] najmanje značajan, a D[3] najznačajniji bit, te osam izlaznih linija: SA, SB, …, SG, SDP. Svaki izlaz spojen je na jedan segment prikazivača. Potrebna je također i jedna dodatna ulazna linija, /TST, aktivna u logičkoj nuli, koja će uzrokovati istovremeno paljenje svih segmenata. Funkcije je moguće realizirati čisto kombinacionim elementima jer nije potrebno pamtiti vrijednosti stanja. Segment se pali dovođenjem logičke 1 na pripadni izlaz PAL-a. Tablica istinitosti za navedenu funkciju je slijedeća (tabela 3.8):

Tabela 3.8 Tablica istinitosti za Primjer 2. (dekoder za 7-segmentni prikazivač)

ULAZI					IZLAZI							
/TST	**D[3]**	**D[2]**	**D[1]**	**D[0]**	**SA**	**SB**	**SC**	**SD**	**SE**	**SF**	**SG**	**SDP**
0	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	0
1	0	0	1	0	1	1	0	1	1	0	1	0
1	0	0	1	1	1	1	1	1	0	0	1	0
1	0	1	0	0	0	1	1	0	0	1	1	0
1	0	1	0	1	1	0	1	1	0	1	1	0
1	0	1	1	0	1	0	1	1	1	1	1	0
1	0	1	1	1	1	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1	1	1	1	0
1	1	0	0	1	1	1	1	1	0	1	1	0
1	1	0	1	0	1	1	1	0	1	1	1	0
1	1	0	1	1	0	0	1	1	1	1	1	0
1	1	1	0	0	1	0	0	1	1	1	0	0
1	1	1	0	1	0	1	1	1	1	0	1	0
1	1	1	1	0	1	0	0	1	1	1	1	0
1	1	1	1	1	1	0	0	0	1	1	1	0

Funkcija ovisnosti pojedinog izlaza o ulazima može se napisati na više načina. Jedan od njih je da se kao u Primjeru 1. identificiraju i raspišu stanja kada će pojedini segment biti aktivan tj. u logičkoj 1. Segment će biti aktivan ili kada prikazuje neki od brojeva ili kada je aktivan signal /TST koji uzrokuje paljenje svih segmenata. Signal za aktivaciju svih segmenata aktivan je u nuli pa se stoga i deklarira kao /TST u bloku za deklaraciju pinova. **Dobra je praksa prilikom projektiranja PAL-a pinove deklarirati prema njihovom stvarnom polaritetu, tj. ako je signal SIG aktivan u 0, u bloku za deklaraciju pinova ga treba deklarirati kao /SIG, odnosno ako je aktivan u 1, treba ga deklarirati kao SIG. Nakon takve deklaracije ulaznih signala svi signali se u logičkim izrazima koriste u pozitivnoj logici (aktivni u 1), neovisno o njihovom stvarnom polaritetu na ulazu.** U nastavku su radi dužine prikazani samo primjeri za segmente A i B. Izrazi za ostale segmente pišu se analogno tome.

```
SA = /TST * (  /D[3] * /D[2] * /D[1] * /D[0] +
               /D[3] * /D[2] *  D[1] * /D[0] +
               /D[3] * /D[2] *  D[1] *  D[0] +
               /D[3] *  D[2] * /D[1] *  D[0] +
               /D[3] *  D[2] *  D[1] * /D[0] +
               /D[3] *  D[2] *  D[1] *  D[0] +
                D[3] * /D[2] * /D[1] * /D[0] +
                D[3] * /D[2] * /D[1] *  D[0] +
                D[3] * /D[2] *  D[1] * /D[0] +
                D[3] *  D[2] * /D[1] * /D[0] +
                D[3] *  D[2] *  D[1] * /D[0] +
                D[3] *  D[2] *  D[1] *  D[0] ) +
      TST * (VCC)

SB = /TST * (  /D[3] * /D[2] * /D[1] * /D[0] +
               /D[3] * /D[2] * /D[1] *  D[0] +
               /D[3] * /D[2] *  D[1] * /D[0] +
               /D[3] * /D[2] *  D[1] *  D[0] +
               /D[3] *  D[2] * /D[1] * /D[0] +
               /D[3] *  D[2] *  D[1] *  D[0] +
                D[3] * /D[2] * /D[1] * /D[0] +
                D[3] * /D[2] * /D[1] *  D[0] +
                D[3] * /D[2] *  D[1] * /D[0] +
                D[3] *  D[2] * /D[1] *  D[0] ) +
      TST * (VCC)
```

Drugi način za prikaz ovisnosti izlaza o ulazima za navedeni primjer realiziran je pomoću struktura IF-THEN-ELSE i CASE uz korištenje vektorskog zapisa. Signali SA, SB, ..., SG, SDP mogu se prikazati kao vektor S[7..0] pri čemu je S[7] = SA, S[6] = SB itd. U funkciji CASE se na temelju mogućih vrijednosti ulaznih linija D[3..0] izlazne linije S[7..0] postavljaju u stanje logičke 1 ili 0 prema tablici istinitosti. Npr. ulaznom vektoru D[3..0] = #b0000 (binarna 4-bitna 0) odgovara izlazni vektor S[7..0] = #b11111100; ulaznom vektoru D[3..0] = #b0001 (binarna 4-bitna jedinica) odgovara izlazni vektor S[7..0] = #b01100000, i analogno tome slijedi sve do ulaznog vektora D[3..0] = #b1111 (binarni broj F) kojem odgovara izlazni vektor S[7..0] = #b10001110. Funkcija se dakle može napisati i ovako:

```
IF ( TST ) THEN
    BEGIN
      S[7..0] = #B11111111
    END
ELSE


    BEGIN
      CASE ( D[3..0] )
```

```
        BEGIN
          #b0000: BEGIN
                    S[7..0] = #b11111100
                  END
          #b0001: BEGIN
                    S[7..0] = #b01100000
                  END
          #b0010: BEGIN
                    S[7..0] = #b11011010
                  END
                 ....

          #b1111: BEGIN
                    S[7..0] = #b10001110
                  END
        END
END
```

Simulacija će biti objašnjena za gornji oblik realizacije. U postupku simulacije najprije treba definirati koje će se ulazne i izlazne linije pratiti (naredba TRACE_ON). Polaritet linija je najzgodnije postaviti kao što je to učinjeno u definiciji pinova, a u simulaciji vrijedi isto pravilo kao i kod pisanja izraza, dakle signal je aktivan bez znaka negacije, odnosno neaktivan ili u logičkoj 0 ako je napisan sa znakom negacije. Slijedeća naredba je SETF kojom se ulazne linije postavljaju u neko stanje, a nakon nje slijedi naredba CHECK u kojoj se navode vrijednosti koje izlazi moraju poprimiti za navedene vrijednosti ulaza. Par naredbi SETF-CHECK se ponavlja dok nisu definirani svi test vektori od interesa. Završna naredba kojom prestaje simulacija je TRACE_OFF.

U gornjem primjeru su radi jednostavnijeg zapisa odmah nakon definicije pinova navedeni string-ovi koji će se koristiti u naredbi CHECK za provjeru stanja. Pa su tako neki od string izraza slijedeći:

```
STRING   SVI    '  S[7]  S[6]  S[5]  S[4]  S[3]  S[2]  S[1]  S[0] '
STRING   NULA   '  S[7]  S[6]  S[5]  S[4]  S[3]  S[2] /S[1] /S[0] '
STRING   JEDAN  ' /S[7]  S[6]  S[5] /S[4] /S[3] /S[2] /S[1] /S[0] '
...       ...                                  .....
STRING   HEXAF  '  S[7] /S[6] /S[5] /S[4]  S[3]  S[2]  S[1] /S[0] '
```

, a simulacija izgleda ovako:

```
SIMULATION
TRACE_ON  /TST  D[3..0] S[7..0]
SETF  /D[3] /D[2]  D[1]  D[0]  TST
CHECK  SVI
SETF  /D[3] /D[2] /D[1] /D[0] /TST
CHECK  NULA
SETF  /D[3] /D[2] /D[1]  D[0] /TST
CHECK  JEDAN
.....
SETF   D[3]  D[2]  D[1]  D[0] /TST
CHECK  HEXAF
TRACE_OFF
```

Ukupna rješenja primjera dana su u poglavlju 3.6.

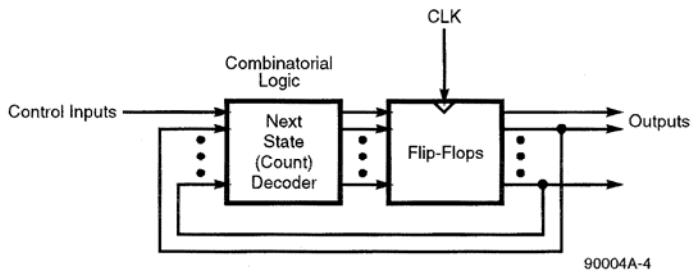
3.5.2 Primjer realizacije registarske funkcije

a) Primjer 3.

Treba realizirati 3-bitno sinkrono binarno brojilo na način da se brojanje brojila vidi na sedamsegmentnom prikazivaču kao paljenje segmenata: A (najznačajniji bit), G i D (najmanje značajan bit) ovisno o stanju brojila. Brojilo se resetira (svi izlazi u 0) dovođenjem aktivnog signala (logičke 0) na ulaznu liniju /RES. Brojilo broji kada je aktivna ulazna linija CE (aktivna u 1), u protivnom ostaje u prethodnom stanju. Treća ulazna linija UD definira da li brojilo broji prema naprijed (ako je u logičkoj 1) ili unazad (stanje logičke 0). Izlazi iz bistabila trajno su omogućeni linijom /OE. Napisati postupak simulacije.

b) Rješenje

Sinkrono binarno brojilo predstavlja sekvencijalni sklop koji na rastući brid signala takta, koji je zajednički za sve bistabile brojila, mijenja stanja na izlazu prema nekom predefiniranom pravilu. Novi ulaz u bistabile brojila dobiva se iz kombinacione logičke funkcije svih izlaza dovedenih u povratnoj petlji, Slika 3.10.



Slika 3.10 Sinkrono binarno brojilo

U ovom primjeru je za razliku od prethodno objašnjenih kombinacionih funkcija potrebno poznavati prethodno stanje sklopa da bi se prešlo u slijedeće. U tu svrhu koristit će se tri tzv. registarska izlaza na PAL sklopu koji su nazvani B2 (najznačajniji bit), zatim B1 te B0 (najmanje značajan bit), a svaki od njih predstavlja izlaz iz jednog bistabila. Stanja na ulazima u bistabile označavaju se istom oznakom kao i izlazi jer u biti predstavljaju izlaz u slijedećem koraku. Vremenski takt dovodi se na CLK ulaze bistabila, a izlazi se mijenjaju na rastući brid takta.

Tablica istinitosti se za sekvencijalne sklopove prikazuje u obliku tabele prelaska iz koje mora biti vidljivo iz kojeg stanja se prelazi u koje novo stanje. Za naš primjer dana je Tabelom 3.9, uz napomenu da stanje $B_{20}B_{10}B_{00}$ označava neku bilo koju konkretnu kombinaciju, koja je za navedeni slučaj ista i u ovom trenutku i u slijedećem koraku.

Tabela 3.9 Tablica istinitosti za Primjer 3. (binarno brojilo)

ULAZI			TRENUTNO STANJE IZLAZA			SLIJEDEĆE STANJE IZLAZA		
/RES	**CE**	**UD**	**B2**	**B1**	**B0**	**B2**	**B1**	**B0**
0	X	X	X	X	X	0	0	0
1	0	X	B_{20}	B_{10}	B_{00}	B_{20}	B_{10}	B_{00}
1	1	1	0	0	0	0	0	1
1	1	1	0	0	1	0	1	0
1	1	1	0	1	0	0	1	1
1	1	1	0	1	1	1	0	0
1	1	1	1	0	0	1	0	1
1	1	1	1	0	1	1	1	0
1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	0	0
1	1	0	0	0	0	1	1	1
1	1	0	0	0	1	0	0	0
1	1	0	0	1	0	0	0	1
1	1	0	0	1	1	0	1	0
1	1	0	1	0	0	0	1	1
1	1	0	1	0	1	1	0	0
1	1	0	1	1	0	1	0	1
1	1	0	1	1	1	1	1	0

Mogu se pisati izrazi:

```
B0 = /RES * ( /CE *  B0   +
              CE * /B0 ) +
       RES * GND

B1 = /RES * (  UD *  CE * ( /B2 * /B1 *  B0 +
                           /B2 *  B1 * /B0 +
                            B2 * /B1 *  B0 +
                            B2 *  B1 * /B0 ) +
               /UD *  CE * ( /B2 * /B1 * /B0 +
                           /B2 *  B1 *  B0 +
                            B2 * /B1 * /B0 +
                            B2 *  B1 *  B0 ) +
               /CE *  B1  )  +
       RES * GND


B2 = /RES * (  UD *  CE * ( /B2 *  B1 *  B0 +
                            B2 * /B1 * /B0 +
                            B2 * /B1 *  B0 +
                            B2 *  B1 * /B0 ) +
               /UD *  CE * ( /B2 * /B1 * /B0 +
                            B2 * /B1 *  B0 +
                            B2 *  B1 * /B0 +
                            B2 *  B1 *  B0 ) +
               /CE *  B2 ) +
       RES * GND
```

Simulacija registarskog dizajna u odnosu na kombinacioni razlikuje se po činjenici da se prelazak iz stanja u stanje događa na rastući brid vremenskog takta. U slučaju

simulacije treba dakle ulaznu liniju CLK postaviti prvo u stanje 0, zatim postaviti neku kombinaciju na ulaznim linijama /RES, CE i UD, promijeniti CLK u 1 čime se ostvaruje rastući brid, te onda opet vratiti CLK u stanje logičke 0. Za simulaciju treba osim toga prije svega aktivirati ulaz /OE. Stanje na izlazima može se provjeriti naredbom CHECK i odgovarajućom kombinacijom na linijama B2, B1 i B0 ili nakon svakog rastućeg brida takta ili nakon nekoliko njih. Jedan primjer simulacije slijedi.

```
SIMULATION
TRACE_ON  CLK  /RES  CE  UD  B2  B1  B0
SETF  /CLK  OE                ; aktiviraj OE
SETF   RES                    ; postavi reset
SETF   CLK                    , rastu•i brid na CLK
CHECK  /B2 /B1 /B0            ; provjeri da su svi izlazi u 0
SETF  /CLK
SETF  /RES  CE UD             ; omogu•i brojanje prema gore
FOR I := 1 TO 10 DO           ; broji od 1 do 10
    BEGIN
      SETF   CLK
      SETF  /CLK
    END
CHECK  /B2 B1 /B0             ; provjeri izlaze
SETF  /CE                     ; onemogu•i brojanje
FOR I := 1 TO 3 DO            ; broji od 1 do 3
    BEGIN
      SETF   CLK
      SETF  /CLK
    END
CHECK  /B2 B1 /B0             ; provjeri izlaze (… nepromijenjeni)
SETF  CE                      ; ponovno omogu•i brojanje prema gore
FOR I := 1 TO 3 DO            ; broji od 1 do 3
    BEGIN
      SETF   CLK
      SETF  /CLK
    END
CHECK  B2 /B1 B0              ; provjeri izlaze
SETF  /UD                     ; brojanje unazad
FOR I := 1 TO 6 DO            ; broji od 1 do 6
    BEGIN
      SETF   CLK
      SETF  /CLK
    END
CHECK  B2 B1 B0               ; provjeri izlaze
TRACE_OFF
```

Treba naglasiti da se u ovom primjeru linijom /RES ostvaruje sinkroni reset, tj. ova linija mora biti aktivna na rastući brid takta da bi se sklop resetirao. Isto tako mora biti neaktivna kod pojave brida takta da bi sklop prešao u normalni način rada. Ukupna rješenja primjera dana su u poglavlju 3.6.

3.5.3 ZADATAK 1.

Potrebno je realizirati slijedeću kombinacionu funkciju. Ulaz u PAL predstavljaju tri ulazne linije nazovimo ih U, V i W. Iz PAL-a izlazi 8 linija koje su spojene na sedamsegmentni prikazivač s decimalnom točkom (opisan u Primjeru 2. za kombinacione funkcije). Na prikazivaču treba prikazati decimalni broj koji odgovara sumi vrijednosti postavljenih na ulaznim linijama. Kada je suma jednaka 3 aktivirati i decimalnu točku. Npr. ako je stanje na ulazima (binarno) U = 1, V = 1 i W = 0 , izlaz treba pokazivati broj 2 (za prikaz broja na prikazivaču vidi Primjer 2.). Raspored pinova i signala dan je u Tabeli 3.10. Kod realizacije imena signala možete uzeti po želji. Preporuka: koristiti vektorski zapis.

Tabela 3.10 Raspored ulaznih i izlaznih pinova na PAL-u za Zadatak 1.

ULAZI		**IZLAZI**	
broj pina PAL-a	**signal**	**broj pina PAL-a**	**signal**
2	U	17	SA
3	V	16	SB
4	W	14	SC
		13	SD
		12	SE
		18	SF
20	VCC	19	SG
10	GND	15	SDP

a) <u>Popuniti tablicu istinitosti koja slijedi.</u>

U	V	W	SUMA	SA	SB	SC	SD	SE	SF	SG	SDP
0	0	0									
0	0	1									
0	1	0									
0	1	1									
1	0	0									
1	0	1									
1	1	0									
1	1	1									

b) <u>Realizirati funkciju sklopom PALCE16V8.</u>

- U direktoriju **C:\PALASM\PURS** napraviti radni direktorij koji treba nazvati svojim prezimenom (do max. 8 slova, ne koristiti naša slova). Raditi u tom direktoriju.
- Pomoć pri obavljanju pojedinih zadataka kao i sintaksu programskog alata PALASM možete naći u poglavljima 3.2 i 3.3..
- Pozvati PALASM i podesiti radni direktorij.

- Nakon otvaranja nove datoteke od općih podataka o dizajnu popuniti samo CHIP NAME polje (nazvati onako kako ćete nazvati cijelu datoteku) i najvažniji podatak u ovom dijelu, a to je tip sklopa. **Uzeti PAL16V8.**
- Deklarirati pinove (prema tabeli 3.10) i voditi računa o njihovom tipu (kombinacioni ili registarski).
- Napisati Boole-ove izraze koristeći napisanu tablicu istinitosti.
- Prevesti dizajn i ispraviti eventualne pogreške.
- Realizirati simulaciju uz korištenje poglavlja 3.3. Navesti sve moguće vrijednosti ulaza (test vektore) i provjeriti izlaze (vidi simulaciju u Primjeru 2. teoretskog dijela ove vježbe).

c) <u>Provjeriti ispravnost rada ostvarene funkcije u PAL-u na maketi</u>

- obavijestiti voditelja laboratorija ili demonstratore da želite isprogramirati PAL;
- preuzeti isprogramirani PAL;
- prije početka rada **napajanje** modula mora biti **isključeno** !
- staviti PAL u podnožje na mjestu U202, tako da je oznaka na rubu chip-a okrenuta prema gore (dobro gledati orijentaciju chip-a; vidi Sliku 3.7);
- uključiti napajanje;
- Pročitati poglavlje 3.4. Proučiti shemu makete dane na slici 3.6 i izgled gornje strane štampane pločice dan na slici 3.7 da biste ustanovili koji kratkospojnik utječe na koji ulaz PAL-a. Promjenama na dotičnim kratkospojnicima provjeriti rad PAL-a. Kada kapica nije stavljena na pinove kratkospojnika ulazna linija je jednaka 1, kada je stavljena, linija je u stanju logičke 0.
- Ispitati rad sklopa za sve vrijednosti ulaznih linija PAL-a.
- Pažljivo stavljati i vaditi PAL iz podnožja da se ne oštete nožice.
- U slučaju da ostvarena funkcija nije ispravna, provjeriti i ispraviti dizajn te ponoviti postupak.

3.5.4 ZADATAK 2.

Potrebno je realizirati slijedeću registarsku funkciju. Na sedamsegmentnom prikazivaču samo jedan svijetleći segment treba kružiti po gornjem dijelu koji čine segmenti A, B, G i F. prikazivača. Drugim riječima prvo je aktivan samo jedan segment, pa se on zagasi, a postaje aktivan prvi do njega u krugu itd. Kruženje počinje paljenjem gornjeg segmenta i nastavlja se u smjeru kazaljke na satu. Prije prvog paljenja gornjeg segmenta, svi segmenti se prvo trebaju zagasiti. Segmente koji se ne koriste deklarirati kao kombinacione ulaze i postaviti ih u GND (da ne svijetle). Ostvariti i funkciju da kada se aktivira dodatna ulazna linija SMJER, kruženje segmenta promijeni smjer u suprotan, a uz to se upali decimalna točka. Imena pinova (signala) odabrati po želji. Raspored pinova i signala dan je u Tabeli 3.11.

Tabela 3.11 Raspored ulaznih i izlaznih pinova na PAL-u za Zadatak 2.

ULAZI		IZLAZI	
broj pina PAL-a	signal	broj pina PAL-a	signal
2	SMJER	17	SA
11	/OE	16	SB
1	CLK	14	SC
		13	SD
		12	SE
		18	SF
20	VCC	19	SG
10	GND	15	SDP

a) <u>Popuniti tablicu istinitosti koja slijedi.</u>

SMJER	STARO STANJE					NOVO STANJE				
	SA	SB	SG	SF	SDP	SA	SB	SG	SF	SDP

b) <u>Realizirati funkciju sklopom PALCE16V8</u>

- Raditi u već postojećem direktoriju **C:\PALASM\PURS*prezime***.
- Novi zadatak raditi u datoteci istog imena kao u Zadatku 1. ali umjesto "_1" dodati "_2" npr. PETRIN_2.
- Da bi se ubrzao rad, dozvoljeno je kopiranje i mijenjanje datoteke koja je dobivena u Zadatku 1., ali na kraju moraju postojati obje datoteke. Ako se datoteka kopira, paziti da se u njoj promijeni sve što je potrebno (da ne bi ostali neki podaci od prošlog zadatka).
- Deklarirati pinove (prema tabeli 3.11). **PAZI ! Ovaj dizajn je registarski.**
- Napisati Bool-ove izraze koristeći napisanu tablicu istinitosti.
- Ako se koristi CASE naredba, svakako napisati stanje sklopa uz bilo koju vrijednost koja nije definirana (OTHERWISE – vidi sintaksu naredbe CASE).
- Realizirati simulaciju; navesti sve moguće vrijednosti ulaza (test vektore) i provjeriti izlaze (vidi simulaciju u Primjeru 3.).

c) <u>Provjeriti ispravnost rada ostvarene funkcije u PAL-u na maketi</u>

- Provjeriti rad sklopa kao u prethodnom zadatku.

3.6 Vježba 6. - dodatak: Ukupna rješenja primjera opisanih u vježbi

a) Primjer 1.

```
;PALASM Design Description
;---------------------------------- Declaration Segment ------------
TITLE    Sedamsegmentni dekoder kombinaciona varijanta I
PATTERN  SEDSEG_1
REVISION 1.0
AUTHOR   Davorka Petrinovic
COMPANY  FER/ZESOI
DATE     10/17/98

CHIP  SEDSEG_1  PALCE16V8
;---------------------------------- PIN Declarations ---------------
PIN  2          D[3]                          COMBINATORIAL ; INPUT
PIN  3          D[2]                          COMBINATORIAL ; INPUT
PIN  4          D[1]                          COMBINATORIAL ; INPUT
PIN  5          D[0]                          COMBINATORIAL ; INPUT
PIN  17         SA                            COMBINATORIAL ; OUTPUT
PIN  16         SB                            COMBINATORIAL ; OUTPUT
PIN  14         SC                            COMBINATORIAL ; OUTPUT
PIN  13         SD                            COMBINATORIAL ; OUTPUT
PIN  12         SE                            COMBINATORIAL ; OUTPUT
PIN  18         SF                            COMBINATORIAL ; OUTPUT
PIN  19         SG                            COMBINATORIAL ; OUTPUT
PIN  15         SDP                           COMBINATORIAL ; OUTPUT
PIN  9          /TST                          COMBINATORIAL ; INPUT
PIN  10         GND                                         ;
PIN  20         VCC                                         ;

;----------------------------------- Boolean Equation Segment ------
EQUATIONS

SA =  /TST * ( /D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 0
               /D[3] * /D[2] *  D[1] * /D[0] +     ; HEX 2
               /D[3] * /D[2] *  D[1] *  D[0] +     ; HEX 3
               /D[3] *  D[2] * /D[1] *  D[0] +     ; HEX 5
               /D[3] *  D[2] *  D[1] * /D[0] +     ; HEX 6
               /D[3] *  D[2] *  D[1] *  D[0] +     ; HEX 7
                D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 8
                D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 9
                D[3] * /D[2] *  D[1] * /D[0] +     ; HEX A
                D[3] *  D[2] * /D[1] * /D[0] +     ; HEX C
                D[3] *  D[2] *  D[1] * /D[0] +     ; HEX E
                D[3] *  D[2] *  D[1] *  D[0] ) +   ; HEX F
       TST * ( VCC )

SB =  /TST * ( /D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 0
               /D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 1
               /D[3] * /D[2] *  D[1] * /D[0] +     ; HEX 2
               /D[3] * /D[2] *  D[1] *  D[0] +     ; HEX 3
               /D[3] *  D[2] * /D[1] * /D[0] +     ; HEX 4
               /D[3] *  D[2] *  D[1] *  D[0] +     ; HEX 7
                D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 8
                D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 9
                D[3] * /D[2] *  D[1] * /D[0] +     ; HEX A
                D[3] *  D[2] * /D[1] *  D[0] ) +   ; HEX D
       TST * ( VCC )
```

```
SC =  /TST * ( /D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 0
               /D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 1
               /D[3] * /D[2] *  D[1] *  D[0] +     ; HEX 3
               /D[3] *  D[2] * /D[1] * /D[0] +     ; HEX 4
               /D[3] *  D[2] * /D[1] *  D[0] +     ; HEX 5
               /D[3] *  D[2] *  D[1] * /D[0] +     ; HEX 6
               /D[3] *  D[2] *  D[1] *  D[0] +     ; HEX 7
                D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 8
                D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 9
                D[3] * /D[2] *  D[1] * /D[0] +     ; HEX A
                D[3] * /D[2] *  D[1] *  D[0] +     ; HEX B
                D[3] *  D[2] * /D[1] *  D[0] ) +   ; HEX D
       TST * ( VCC )

SD =  /TST * ( /D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 0
               /D[3] * /D[2] *  D[1] * /D[0] +     ; HEX 2
               /D[3] * /D[2] *  D[1] *  D[0] +     ; HEX 3
               /D[3] *  D[2] * /D[1] *  D[0] +     ; HEX 5
               /D[3] *  D[2] *  D[1] * /D[0] +     ; HEX 6
                D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 8
                D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 9
                D[3] * /D[2] *  D[1] *  D[0] +     ; HEX B
                D[3] *  D[2] * /D[1] * /D[0] +     ; HEX C
                D[3] *  D[2] * /D[1] *  D[0] +     ; HEX D
                D[3] *  D[2] *  D[1] * /D[0] ) +   ; HEX E
       TST * ( VCC )

SE =  /TST * ( /D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 0
               /D[3] * /D[2] *  D[1] * /D[0] +     ; HEX 2
               /D[3] *  D[2] *  D[1] * /D[0] +     ; HEX 6
                D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 8
                D[3] * /D[2] *  D[1] * /D[0] +     ; HEX A
                D[3] * /D[2] *  D[1] *  D[0] +     ; HEX B
                D[3] *  D[2] * /D[1] * /D[0] +     ; HEX C
                D[3] *  D[2] * /D[1] *  D[0] +     ; HEX D
                D[3] *  D[2] *  D[1] * /D[0] +     ; HEX E
                D[3] *  D[2] *  D[1] *  D[0] ) +   ; HEX F
       TST * ( VCC )

SF =  /TST * ( /D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 0
               /D[3] *  D[2] * /D[1] * /D[0] +     ; HEX 4
               /D[3] *  D[2] * /D[1] *  D[0] +     ; HEX 5
               /D[3] *  D[2] *  D[1] * /D[0] +     ; HEX 6
                D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 8
                D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 9
                D[3] * /D[2] *  D[1] * /D[0] +     ; HEX A
                D[3] * /D[2] *  D[1] *  D[0] +     ; HEX B
                D[3] *  D[2] * /D[1] * /D[0] +     ; HEX C
                D[3] *  D[2] *  D[1] * /D[0] +     ; HEX E
                D[3] *  D[2] *  D[1] *  D[0] ) +   ; HEX F
       TST * ( VCC )
```

```
SG =  /TST * ( /D[3] * /D[2] *  D[1] * /D[0] +     ; HEX 2
               /D[3] * /D[2] *  D[1] *  D[0] +     ; HEX 3
               /D[3] *  D[2] * /D[1] * /D[0] +     ; HEX 4
               /D[3] *  D[2] * /D[1] *  D[0] +     ; HEX 5
               /D[3] *  D[2] *  D[1] * /D[0] +     ; HEX 6
                D[3] * /D[2] * /D[1] * /D[0] +     ; HEX 8
                D[3] * /D[2] * /D[1] *  D[0] +     ; HEX 9
                D[3] * /D[2] *  D[1] * /D[0] +     ; HEX A
                D[3] * /D[2] *  D[1] *  D[0] +     ; HEX B
                D[3] *  D[2] * /D[1] *  D[0] +     ; HEX D
                D[3] *  D[2] *  D[1] * /D[0] +     ; HEX E
                D[3] *  D[2] *  D[1] *  D[0] ) +   ; HEX F
       TST * ( VCC )

SDP =  TST * ( VCC )

;---------------------------------- Simulation Segment ------------
SIMULATION

;------------------------------------------------------------------
```

b) Primjer 2.

```
;PALASM Design Description

;---------------------------------- Declaration Segment ------------
TITLE    Sedamsegmentni dekoder kombinaciona varijanta II (vektorski)
PATTERN  DEC7SEG
REVISION 1.0
AUTHOR   Davorka
COMPANY  FER/ZESOI
DATE     10/17/98

CHIP  DEC7SEG  PALCE16V8

;---------------------------------- PIN Declarations ---------------
PIN  2,3,4,5    D[3..0]                          COMBINATORIAL ; INPUT
PIN  17,16,14..12,18,19,15      S[7..0]          COMBINATORIAL ; OUTPUT
PIN   9         /TST                             COMBINATORIAL ; INPUT
PIN  10         GND                                            ;
PIN  20         VCC                                            ;

STRING        SVI       ' S[7]  S[6]  S[5]  S[4]  S[3]  S[2]  S[1]  S[0]'
STRING        NULA      ' S[7]  S[6]  S[5]  S[4]  S[3]  S[2] /S[1] /S[0]'
STRING        JEDAN     '/S[7]  S[6]  S[5] /S[4] /S[3] /S[2] /S[1] /S[0]'
STRING        DVA       ' S[7]  S[6] /S[5]  S[4]  S[3] /S[2]  S[1] /S[0]'
STRING        TRI       ' S[7]  S[6]  S[5]  S[4] /S[3] /S[2]  S[1] /S[0]'
STRING        CETIRI    '/S[7]  S[6]  S[5] /S[4] /S[3]  S[2]  S[1] /S[0]'
STRING        PET       ' S[7] /S[6]  S[5]  S[4] /S[3]  S[2]  S[1] /S[0]'
STRING        SEST      ' S[7] /S[6]  S[5]  S[4]  S[3]  S[2]  S[1] /S[0]'
STRING        SEDAM     ' S[7]  S[6]  S[5] /S[4] /S[3] /S[2] /S[1] /S[0]'
STRING        OSAM      ' S[7]  S[6]  S[5]  S[4]  S[3]  S[2]  S[1] /S[0]'
STRING        DEVET     ' S[7]  S[6]  S[5]  S[4] /S[3]  S[2]  S[1] /S[0]'
STRING        HEXAA     ' S[7]  S[6]  S[5] /S[4]  S[3]  S[2]  S[1] /S[0]'
STRING        HEXAB     '/S[7] /S[6]  S[5]  S[4]  S[3]  S[2]  S[1] /S[0]'
STRING        HEXAC     ' S[7] /S[6] /S[5]  S[4]  S[3]  S[2] /S[1] /S[0]'
STRING        HEXAD     '/S[7]  S[6]  S[5]  S[4]  S[3] /S[2]  S[1] /S[0]'
STRING        HEXAE     ' S[7] /S[6] /S[5]  S[4]  S[3]  S[2]  S[1] /S[0]'
STRING        HEXAF     ' S[7] /S[6] /S[5] /S[4]  S[3]  S[2]  S[1] /S[0]'

;---------------------------------- Boolean Equation Segment ------
EQUATIONS

IF (TST) THEN
   BEGIN
        S[7..0] = #B11111111
   END
ELSE
   BEGIN
        CASE (D[3..0])
        BEGIN
             #b0000: BEGIN
                     S[7..0]=#B11111100
                     END
             #b0001: BEGIN
                     S[7..0]=#B01100000
                     END
             #b0010: BEGIN
                     S[7..0]=#B11011010
                     END
```

```
            #b0011: BEGIN
                    S[7..0]=#B11110010
                    END
            #b0100: BEGIN
                    S[7..0]=#B01100110
                    END
            #b0101: BEGIN
                    S[7..0]=#B10110110
                    END
            #b0110: BEGIN
                    S[7..0]=#B10111110
                    END
            #b0111: BEGIN
                    S[7..0]=#B11100000
                    END
            #b1000: BEGIN
                    S[7..0]=#B11111110
                    END
            #b1001: BEGIN
                    S[7..0]=#B11110110
                    END
            #b1010: BEGIN
                    S[7..0]=#B11101110
                    END
            #b1011: BEGIN
                    S[7..0]=#B00111110
                    END
            #b1100: BEGIN
                    S[7..0]=#B10011100
                    END
            #b1101: BEGIN
                    S[7..0]=#B01111010
                    END
            #b1110: BEGIN
                    S[7..0]=#B10011110
                    END
            #b1111: BEGIN
                    S[7..0]=#B10001110
                    END
        END
END
;---------------------------------- Simulation Segment ------------
SIMULATION
TRACE_ON /TST D[3..0] S[7..0]
SETF /D[3] /D[2] D[1] D[0] TST
CHECK SVI
SETF /D[3] /D[2] /D[1] /D[0] /TST
CHECK NULA
SETF /D[3] /D[2] /D[1] D[0] /TST
CHECK JEDAN
SETF /D[3] /D[2]  D[1] /D[0] /TST
CHECK DVA
SETF /D[3] /D[2]  D[1] D[0] /TST
CHECK TRI
SETF /D[3]  D[2] /D[1] /D[0] /TST
CHECK CETIRI
SETF /D[3]  D[2] /D[1] D[0] /TST
CHECK PET
SETF /D[3]  D[2]  D[1] /D[0] /TST
CHECK SEST
```

```
SETF /D[3]  D[2]  D[1] D[0] /TST
CHECK SEDAM
SETF  D[3] /D[2] /D[1] /D[0] /TST
CHECK OSAM
SETF  D[3] /D[2] /D[1] D[0] /TST
CHECK DEVET
SETF  D[3] /D[2]  D[1] /D[0] /TST
CHECK HEXAA
SETF  D[3] /D[2]  D[1] D[0] /TST
CHECK HEXAB
SETF  D[3]  D[2] /D[1] /D[0] /TST
CHECK HEXAC
SETF  D[3] D[2] / D[1] D[0] /TST
CHECK HEXAD
SETF  D[3]  D[2]  D[1] /D[0] /TST
CHECK HEXAE
SETF  D[3]  D[2]  D[1] D[0] /TST
CHECK HEXAF
TRACE_OFF

;--------------------------------------------------------------------
```

c) Primjer 3.

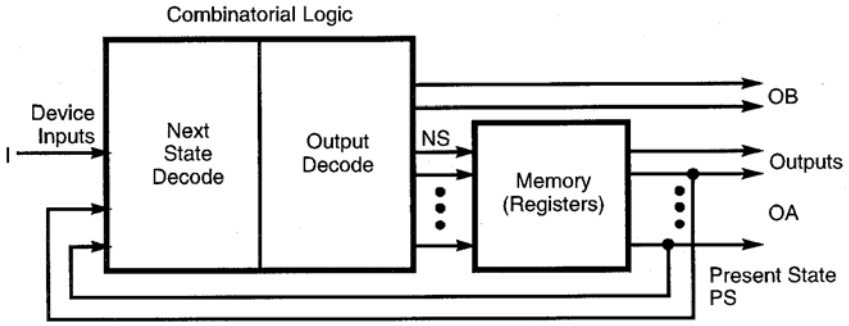
```
;PALASM Design Description

;---------------------------------- Declaration Segment ------------
TITLE    3-bitno binarno brojilo sa UP/DOWN linijom
PATTERN  BIN3CNT2
REVISION 1.0
AUTHOR   Davorka Petrinovic
COMPANY  FER/ZESOI
DATE     10/26/98

CHIP  BIN3CNT2  PALCE16V8

;---------------------------------- PIN Declarations ---------------
PIN  1          CLK                          COMBINATORIAL ; INPUT
PIN  2          CE                           COMBINATORIAL ; INPUT
PIN  3          UD                           COMBINATORIAL ; INPUT
PIN  9          /RES                         COMBINATORIAL ; INPUT
PIN  10         GND                                        ;
PIN  20         VCC                                        ;
PIN  17         B2                           REGISTERED ; OUTPUT
PIN  19         B1                           REGISTERED ; OUTPUT
PIN  13         B0                           REGISTERED ; OUTPUT
PIN  12         E                            COMBINATORIAL ; OUTPUT
PIN  14         C                            COMBINATORIAL ; OUTPUT
PIN  15         DP                           COMBINATORIAL ; OUTPUT
PIN  16         B                            COMBINATORIAL ; OUTPUT
PIN  18         F                            COMBINATORIAL ; OUTPUT
PIN  11         /OE                          COMBINATORIAL ; ENABLE

;---------------------------------- Boolean Equation Segment ------
EQUATIONS

; slijedeci izlazi postavljaju se u nulu da ne bi nakon dovođenja
; napajanja bili upaljeni
E  = GND
C  = GND
DP = GND
B  = GND
F  = GND

; aktivni izlazi brojila

B0 = /RES * (  CE * /B0 +
              /CE *  B0 ) +
      RES * GND


B1 = /RES * (  UD * CE * ( /B2 * /B1 *  B0 +
                           /B2 *  B1 * /B0 +
                            B2 * /B1 *  B0 +
                            B2 *  B1 * /B0 ) +
              /UD * CE * ( /B2 * /B1 * /B0 +
                           /B2 *  B1 *  B0 +
                            B2 * /B1 * /B0 +
                            B2 *  B1 *  B0 ) +
              /CE * B1 ) +
      RES * GND
```

```
B2 = /RES * (  UD * CE * ( /B2 *  B1 *  B0 +
                           B2 * /B1 * /B0 +
                           B2 * /B1 *  B0 +
                           B2 *  B1 * /B0 ) +
               /UD * CE * ( /B2 * /B1 * /B0 +
                           B2 * /B1 *  B0 +
                           B2 *  B1 * /B0 +
                           B2 *  B1 *  B0 ) +
               /CE * B2 ) +
      RES * GND

;---------------------------------- Simulation Segment ------------
SIMULATION

TRACE_ON /OE CLK /RES CE UD B2 B1 B0
SETF /CLK OE
SETF RES
SETF CLK
SETF /CLK
SETF /RES CE UD
FOR I := 1 TO 10 DO
  BEGIN
        SETF CLK
        SETF /CLK
  END
CHECK  /B2 B1 /B0
SETF /CE
FOR I := 1 TO 3 DO
  BEGIN
        SETF CLK
        SETF /CLK
  END
CHECK  /B2 B1 /B0
SETF CE
FOR I := 1 TO 3 DO
  BEGIN
        SETF CLK
        SETF /CLK
  END
CHECK   B2 /B1 B0
SETF /UD
FOR I := 1 TO 6 DO
  BEGIN
        SETF CLK
        SETF /CLK
  END
CHECK   B2 B1 B0
TRACE_OFF

;--------------------------------------------------------------------
```

3.7 Logički automati s konačnim brojem stanja

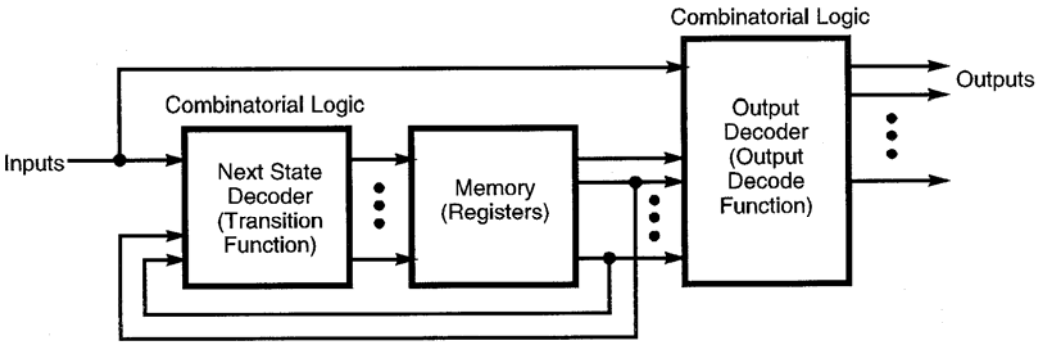
3.7.1 Uvod

Logički automat s konačnim brojem stanja (*engl. Finite State Machine*) (u nastavku korišten izraz konačni automat) je digitalni uređaj koji prolaskom kroz niz unaprijed određenih stanja izvršava određene zadatke. Stanje predstavlja niz vrijednosti mjerenih na različitim dijelovima sklopa. Općeniti oblik konačnog automata prikazan je na slici 3.11.



Slika 3.11 Općeniti blokovski prikaz konačnog automata

Osim ulaza i izlaza konačni automat sastoji se još i od dva ključna elementa: kombinaciona logika i memorija (registri, tj. nizovi bistabila). Memorija se koristi za pohranu stanja konačnog automata u čijoj vrijednosti je zbrojena "prošlost" automata. Kombinaciona logika može se raščlaniti na dva funkcionalna bloka kako je prikazano na slici 3.12, a to su dekoder slijedećeg stanja i izlazni dekoder. Dekoder slijedećeg stanja generira slijedeće stanje konačnog automata dok izlazni dekoder generira stvarne izlaze. Iako obavljaju dvije različite funkcije, funkciju prelaska iz stanja u stanje te izlaznu funkciju, oba funkcionalna bloka se često realiziraju jednom kombinacionom matricom (gornja slika).



Slika 3.12 Prikaz konačnog automata s razdvojenom kombinacionom logikom

Konačni automat obavlja dvije osnovne operacije:

- prolazi kroz niz stanja pri čemu dekoder slijedećeg stanja na temelju trenutnog stanja i ulaznih uvjeta određuje slijedeće stanje;
- također na temelju trenutnog stanja i ulaznih uvjeta generira nizove izlaznih signala koji nastaju kao posljedica prelaska iz stanja u stanje.

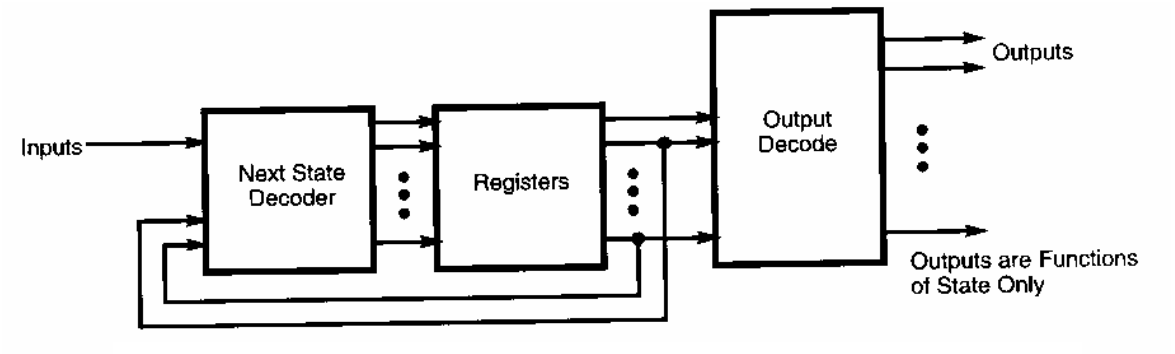
Konačni automati koriste se za modeliranje memorijskih digitalnih sustava kod kojih izlazi ovise o trenutnim ulazima, ali i o prethodnim stanjima na ulazima. Koriste se u nizu aplikacija za upravljanje sustavom. Funkcije koje se mogu ostvariti konačnim automatom su slijedeće: arbitracija, praćenje događaja, ispitivanje višestrukih uvjeta, usklađivanje kašnjenja, generiranje upravljačkih signala itd. Većina konačnih automata su sinkroni sekvencijalni sklopovi kod kojih se prelazak iz stanja u stanje događa istovremeno, na rastući brid signala takta prospojenog na sve izlazne bistabile i bistabile stanja. Primjer jednog takvog jednostavnog automata su i prethodno spominjane registarske realizacije (razna brojila, djelila i sl.). Izvedivi su i asinkroni konačni automati kod kojih nisu svi bistabili spojeni na isti signal takta. Ovakvi su sklopovi, međutim vrlo osjetljivi, složeni za realizaciju i rijetko korišteni. U nastavku će biti korišteni i opisivani samo sinkroni konačni automati.

3.7.2 Tipovi konačnih automata

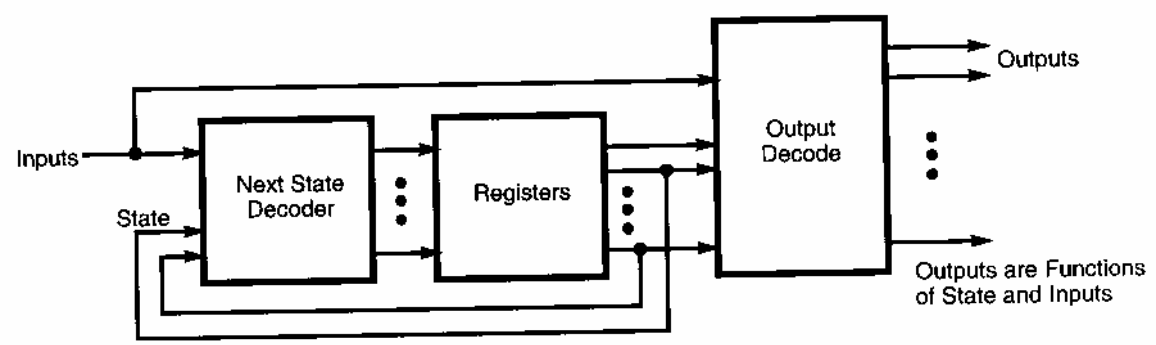
Postoje dva opće poznata tipa konačnih automata: Moore-ov (prikazan na slici3.13) i Mealy-jev (slika 3.14). Ono što određuje tip konačnog automata je korištenje ulaznih signala u p rocesu generiranja slijedećeg izlaza. Kao što se vidi na slici, izlazi Moore-ovog automata funkcija su samo trenutnog stanja. Za razliku od Moore-ovog, Mealy-ev tip automata predstavlja općenitiji model jer i trenutna stanja i signali ulaza sudjeluju u generiranju izlaza.

Kod implementacije Moore-ovog i Mealy-jevog automata na PAL sklopovima sva kombinaciona logika, i za dekodiranje slijedećeg stanja i za dekodiranje izlaza ostvarena je pomoću iste AND-OR matrice. Iako je općenito moguće ostvariti i sinkrone i asinkrone automate, ovdje ćemo razmatrati samo one sinkrone. Tako je na slici 3.15 prikazan sinkroni Moore-ovog automat. Iz slike je vidljivo da izlazi kod ovog konačnog automata mogu biti generirani upravo kada i trenutna stanja.

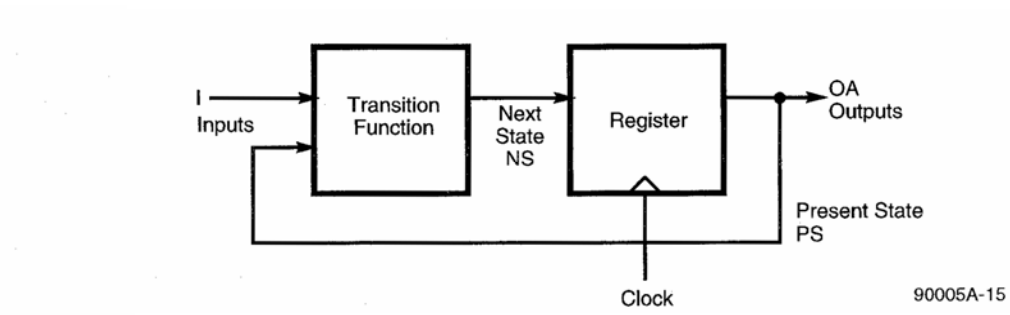
Za razliku od toga, izlazi kod Mealy-jevog sinkronog automata, prikazanog na slici 3.16, ovise o ulazima te su propušteni kroz dodatni izlazni registar. Drugim riječima, izlazi se zbog tog registra ne pojavljuju u stanju za koje su predviđeni već jedno stanje iza. Dakle izlazi za stanjima kasne za jedan signal takta. To se može vidjeti i na slici 3.17 koja prikazuje vremenske dijagrame različitih konačnih automata, za dijagram pod nazivom Registered Mealy Output.



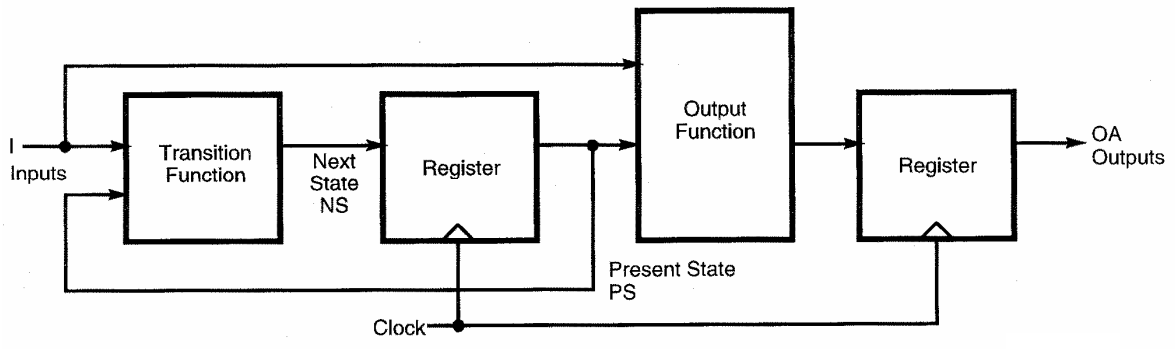
Slika 3.13 Blokovski dijagram Moore-ovog konačnog automata



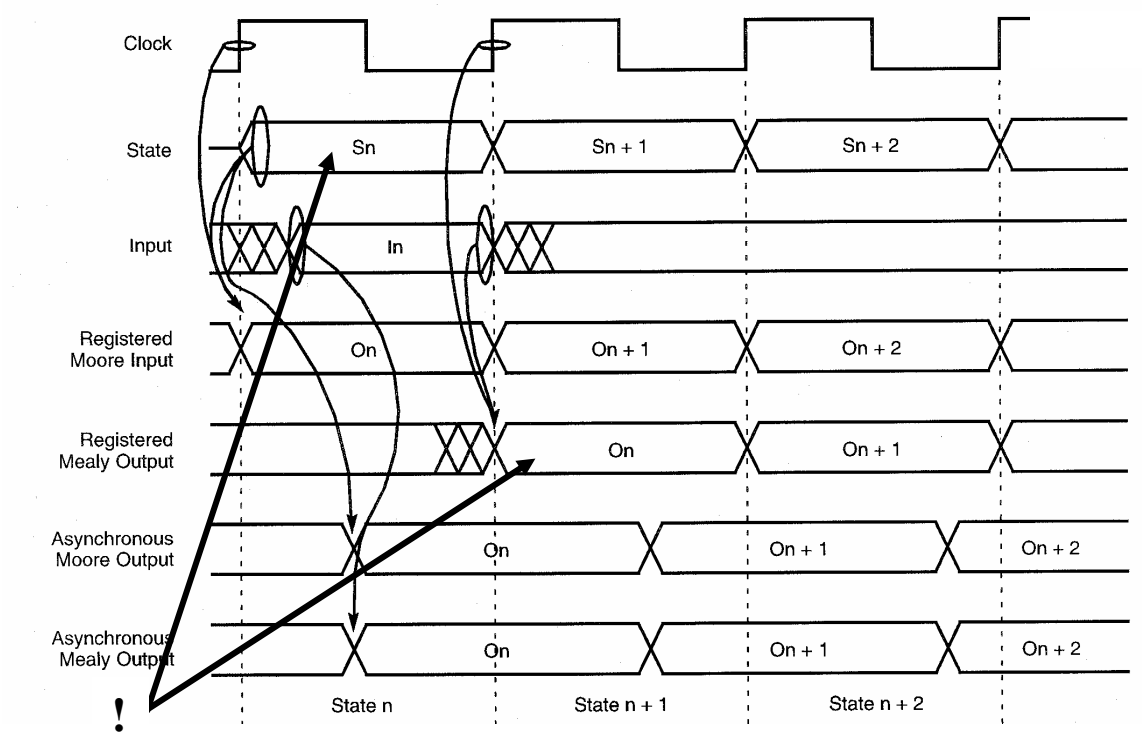
Slika 3.14 Blokovski dijagram Mealy-jevog konačnog automata



Slika 3.15 Izvedba sinkronog Moore-ovog konačnog automata pomoću PAL sklopova



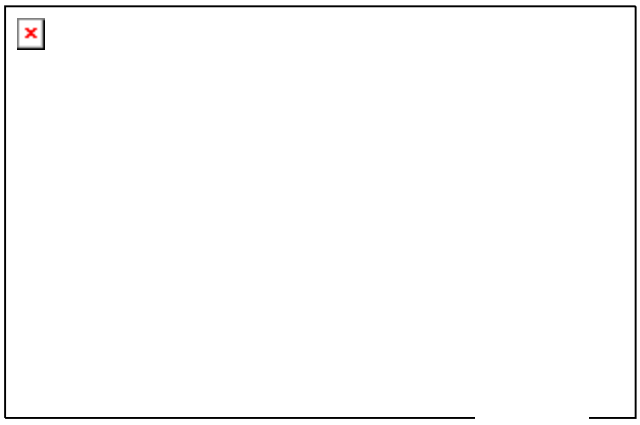
Slika 3.16 Izvedba sinkronog Mealy-jevog konačnog automata pomoću PAL sklopova



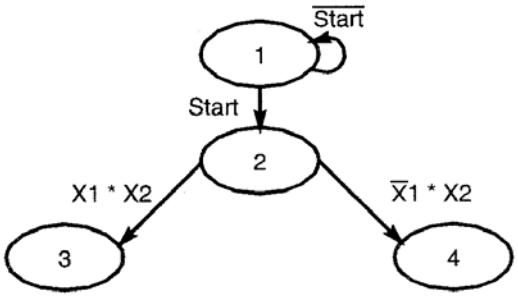
Slika 3.17 Vremenski dijagrami Moore-ovog i Mealy-jevog konačnog automata (Registered označava sinkroni dizajn)

3.7.3 Prikaz konačnih automata

Konačni automati mogu se prikazati na više načina. Jedan od često korištenih je prikaz pomoću dijagrama stanja (*engl. State Transition Diagram*) dan na slici 3.18. Svaki krug predstavlja jedno stanje, a strelice označavaju iz kojeg stanja se prelazi u koje slijedeće stanje. Ulazi koji uzrokuju prelazak iz stanja u stanje te izlazi koji su posljedica pripadnog stanja označeni su pored strelica, odijeljeni horizontalnom crtom. Jedan kompaktniji prikaz pomoću dijagrama stanja dan je na slici 3.19, gdje su ulazni signali zamijenjeni Boole-ovim izrazom definirajući ulaznu kombinaciju koja uzrokuje prelazak iz stanja.



Slika 3.18 Prikaz konačnog automata dijagramom stanja



Slika 3.19 Prikaz konačnog automata dijagramom stanja uz korištenje Boole-ovih izraza

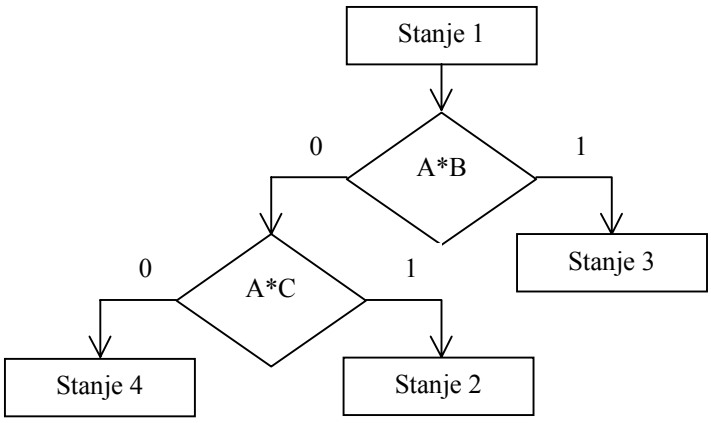
Konačni automati mogu se opisati i u tabelarnoj formi pomoću tabela prijelaza stanja koje imaju format kako je prikazano u tabeli 3.12. U tabeli trebaju biti popisana sva stanja i sve kombinacije ulaznih bitova te pripadni izlazi.

Tabela 3.12 Prikaz automata pomoću tabele prijelaza stanja

Trenutno stanje	Ulazi	Slijedeće stanje	Generirani izlazi
S0-Sn	I0-Im	S0-Sn	O0-Op

Još jedan prikaz konačnih automata je pomoću dijagrama toka dan na slici 3.20 gdje su stanja prikazana pomoću pravokutnika, a uvjeti za prelazak iz jednog stanja u drugo crtaju se pomoću simbola za testiranje koji može imati i više grananja.

Svi prikazi su jednako dobri i ispravni, a njihovo korištenje treba prilagoditi složenosti i broju ulaza i izlaza konačnog automata.



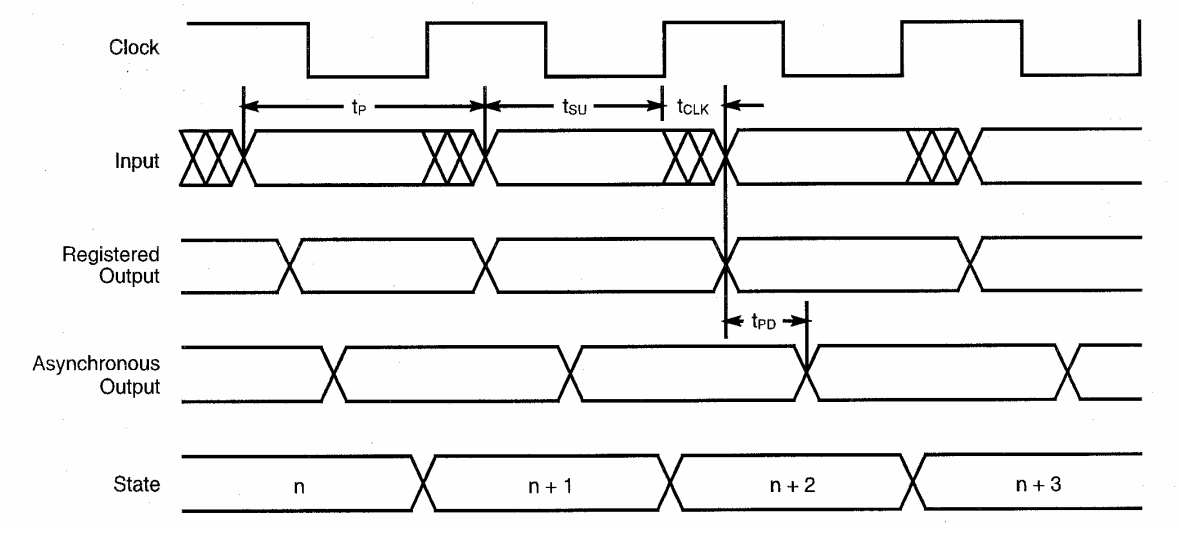
Slika 3.20 Prikaz automata pomoću dijagrama toka

3.7.4 Vremenski odnosi u konačnim automatima

Kao što je već rečeno, signal takta u sinkronom sustavu predstavlja osnovu za evaluaciju ispravnog rada konačnog automata jer se sve promjene događaju na njegov rastući brid. Sve ulazne i izlazne funkcije specificiraju se u odnosu na rastući brid signala takta. Na slici 3.21 dan je prikaz vremenskih odnosa signala takta, ulaza, stanja i izlaza.

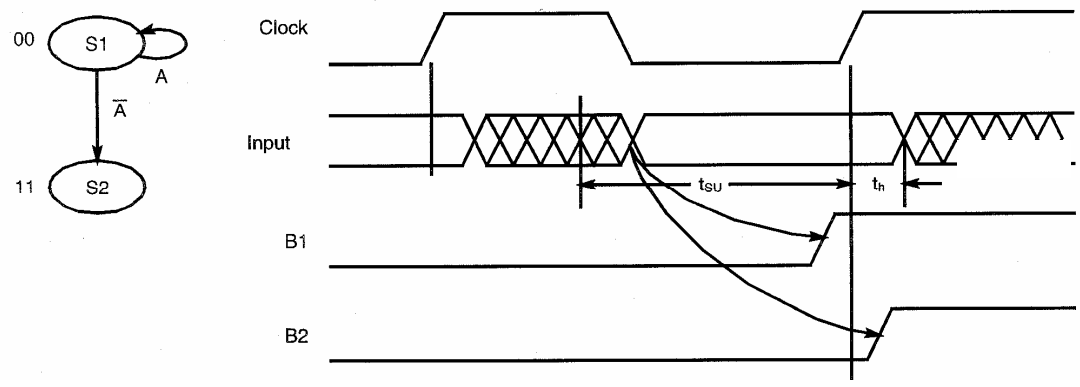
Registarski izlazi bit će dostupni nakon vremena t_{CLK} koje predstavlja kašnjenje od pojave signala takta do pojave izlaza (engl. clock-to-output propagation delay). Asinkroni izlazi traže još i dodatno vrijeme propagacije t_{PD} prije nego što postanu važeći.

Da bi sklop dobro radio, ulazi u bistabile trebaju biti stabilni prije slijedećeg rastućeg brida takta najmanje za vrijeme postavljanja t_S (set-up time), u protivnom bi u bistabile moglo biti spremljeno neispravno stanje. Da bi se to izbjeglo, trajanje signala takta t_p mora biti veće od sume vremena postavljanja i vremena potrebnog da se nakon rastućeg brida pojavi novi važeći izlaz, dakle $t_P = t_{CLK} + t_S$. Time je određena najveća dozvoljena frekvencija signala takta.



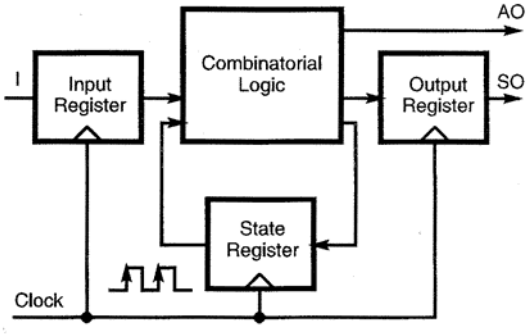
Slika 3.21 Prikaz vremenskih odnosa uz najveću dozvoljenu frekvenciju rada

Ulazi u konačni automat često mogu biti i asinkroni i slučajni, npr. ulaz s tikovnice ili senzora. Može se dakle dogoditi da nisu zadovoljeni zahtjevi za vrijeme postavljanja prije rastućeg brida takta i vrijeme držanja (hold-time) dok se izlazi mijenjaju. Prikaz ove situacije dan je na slici 3.22. Dijagram stanja pokazuje dva moguća prijelaza iz stanja S1 (kod 00) nazad u to isto stanje ili u stanje S2 (kod 11). O tome koji prijelaz će stvarno biti ovisi o stanju ulazne liniji A, asinkrone u odnosu na signal takta. Ovaj ulaz može se promijeniti u bilo kojem trenutku unutar ciklusa takta. Ako se ona promijeni tako da nisu zadovoljena vremena t_s i t_h, može se dogoditi da jedan bistabil za stanja na tu promjenu ulaza prijeđe u stanje 1, a drugi ne. Nastala stanja 01 i 10 u ovom slučaju nisu definirana pa je ponašanje konačnog automata nepredviđeno i potpuno krivo.



Slika 3.22 Prikaz neispravnog rada sklopovlja uzrokovanog asinkronim ulazima

Najsigurnije rješenje ovog problema predstavlja pretvaranje asinkronih ulaza u sinkrone. To se ostvaruje dovođenjem takvog ulaza na bistabil koji će zapamtiti stanje ulaza također na rastući brid signala takta automata, kako je prikazano na slici 3.23. Nedostatak ovog rješenja je u tome što će trenutak reagiranja automata na ulaz biti pomaknut za trajanje jedne periode signala takta od stvarne pojave ulaza jer je ulazni signal koji se koristi u automatu u stvari izlaz sa ulaznog registra.



Slika 3.23 Pretvaranje asinkronog ulaza u sinkroni

3.8 Sintaksa za realizaciju konačnih automata pomoću PAL sklopova

PAL sklopovi su veoma pogodni za realizaciju konačnih automata. Nekoliko primjera konačnih automata realiziranih PAL-om su slijedeći: posebna brojila, prekidni kontroleri, neki tipovi sklopovlja za prikaz video signala i sl. Namijenjeni su za jednostavne upravljačke aplikacije kada se zahtijeva velika brzina, ali relativno mali broj ulaznih i izlaznih pinova.

U ranijim varijantama programske podrške za dizajn PAL sklopova konačni automat opisivan je Boole-ovim izrazima, a novije varijante sadržavaju posebno osmišljenu sintaksu za konačne automate. Ključne riječi i njihovo korištenje sadržani su u tabeli 3.13.

Tabela 3.13 Sintaksa programa PALASM vezana uz konačne automate

sintaksa	značenje
STATE	označava početak segmenta sa stanjima u realizaciji konačnog automata
MEALY_MACHINE	označava Mealy tip konačnog automata
MOORE_MACHINE	označava Moore tip konačnog automata
DEFAULT_BRANCH stanje	za definiciju stanja u koje će se prijeći u slučaju da niti jedan uvjet iz segmenta CONDITIONS (globalna default vrijednost) nije ispunjen
stanje2 := uvjet1 -> stanje1 + uvjet2 -> stanje2 …. +-> local_default_stanje	STATE TRANSITION EQUATIONS izrazi prelaska stanja u stanje; local default predstavlja stanje u koje će se preći ako nije zadovoljen niti jedan uvjet prelaska
za Mealy tip: ime_stanja.OUTF = uvjet1 -> izlazna _komb1 + uvjet2 -> izlazna _komb2 …. +-> local_default_izlazi za Moore tip: ime_stanja.OUTF = izlazna _kombinacija	STATE OUTPUT EQUATIONS izlazne jednadžbe; kod Mealy tipa na izlaz mogu utjecati i neke ulazne linije preko uvjeta; kod Moore tipa izlazi idu direktno
stanjeX = bitstanja1 * bitstanja2 *…* bitstanjan	STATE ASSIGNMENT EQUATIONS izrazi za pridjeljivanje bitova stanja i definiranja kodova za stanja
CONDITIONS	ključna riječ kojom počinje segment za definiciju izraza (uvjeta) za grananja

3.9 Vježba 7: Realizacija logičkih automata s konačnim brojem stanja korištenjem PAL sklopova

3.9.1 Primjer realizacije konačnog automata

U ovom poglavlju bit će opisan primjer realizacije konačnog automata PAL sklopom PALCE16V8.

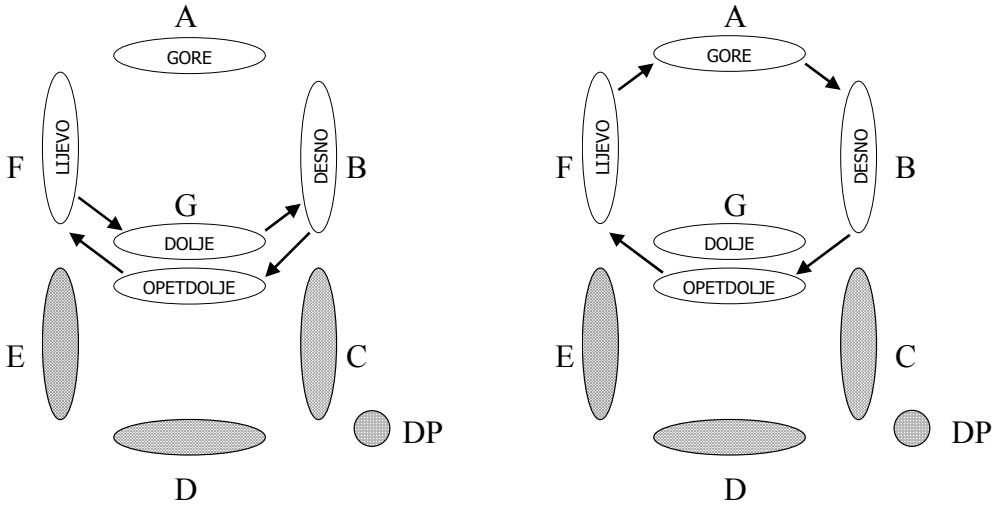
a) <u>Primjer 1.</u>

Korištenjem konačnog automata tipa Mealy treba realizirati slijedeću funkciju. Inicijalno stanje kod uspostavljanja napajanja je stanje INIT uz koje ne svijetli niti jedan segment. Nakon inicijalnog stanja treba ući u stanje koje uzrokuje paljenje lijevog segmenta. Nadalje, ovisno o stanju na ulaznoj liniji VRTI treba svijetliti jedan od segmenata A, B, F ili G prikazivača. Ako je VRTI u neaktivnom stanju (u logičkoj 0) segmenti trebaju svijetliti slijedećim redoslijedom: F→G→ B→G→F i tako stalno (slika 3.24 a)). Na prikazivaču to izgleda kao da se svijetleći segment "ljulja". Kada je VRTI u aktivnom stanju (logička 1), svijetleći segment treba kružiti u smjeru kazaljke na satu tj. redoslijed je slijedeći: F→A→ B→G→F itd. (slika 3.24 b)).

Ulaz u sklop je i linija /RES (sinkroni reset) sa tipke koja u aktivnom stanju (logička nula) uzrokuje da neovisno o tome koji je segment do tada svijetlio, počinje svijetliti segment F i to tako dugo dok je /RES aktivan.

Još jedna ulazna linija INV uvjetuje da li će aktivni izlazi svijetliti ili će biti ugašeni. Drugim riječima kada je ta linija u 0, svijetlit će samo oni segmenti koji bi po gornjoj funkciji trebali svijetliti, a ostali će biti ugašeni. Ako je na ulazu INV logička jedinica, logički nivoi na izlazima bit će takvi da se onaj segment koji bi po predviđenoj funkciji trebao svijetliti zagasi, a oni koji su neaktivni se sada upale (inverzno stanje svijetli-ne svijetli od originalnog).

Nacrtati i dijagram prelaska stanja u stanje.



a) VRTI neaktivan b) VRTI aktivan

Slika 3.24 Prikaz aktivnosti segmenata prikazivača ovisno o liniji VRTI

b) Rješenje

Dobro je najprije ustanoviti koje su linije ulazne, a koje izlazne.

ulazne: VRTI	izlazne (prikažimo ih vektorski):	D[3] … seg. A
INV		D[2] … seg. B
/RES		D[1] … seg. F
		D[0] …. seg. G

Tip dizajna kojeg treba ostvariti PAL-om je dizajn pomoću konačnog automata i to Mealy tipa (slika 3.16). To su podaci koje najprije treba navesti kod pisanja funkcija za PAL. To se izvodi tako da se u dijelu datoteke pod nazivom State Equation Segment, iza kojeg će slijediti dizajn konačnog automata navedu slijedeće ključne riječi:

; ---------- State Equation segment ----------

```
STATE
MEALY_MACHINE
```

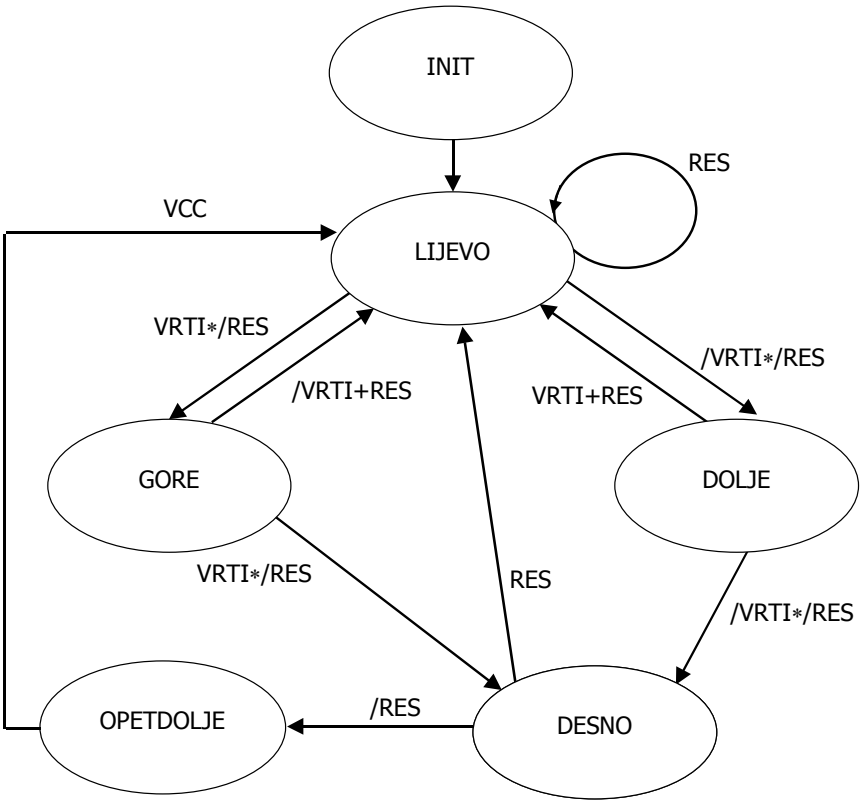
Ako se pogleda slika 3.16 implementacije Mealy sinkronog automata, mogu se uočiti nekoliko cjelina koje treba realizirati. Jednu cjelinu čini funkcija kojom se opisuje prelazak konačnog automata iz jednog stanja u drugo ovisno o ulazima (*engl. Transition Function*). Stanja čine izlazi iz prvog registra. Ova stanja će na slijedeći brid signala takta sudjelovati zajedno s ulaznim signalima u generiranju izlaza u skladu s definiranom izlaznom funkcijom (*engl. Output Function*). Izlazna funkcija je druga cjelina koju treba realizirati. Izlazi prema segmentima nalaze se na izlazu drugog registra u modelu Mealy-evog automata. Treba naglasiti da će zbog modela s dva seta registara u kaskadi, izlazne linije pokazivati vrijednosti koje odgovaraju prethodnom, a ne trenutnom stanju na registrima stanja. Drugim riječima, promjene na izlazima kasne za promjenama stanja za jedan period signala takta.

S obzirom da se automat treba realizirati sklopom PALCE16V8 koji ima 8 bistabila, razmotrimo koliko bistabila pripada stanjima, a koliko izlazima. Treba upravljati s četiri izlazna segmenta pa je stoga četiri bistabila rezervirano za izlaze. Preostali bistabili se mogu koristiti za pamćenje stanja. Izlazi tih bistabila PAL-a spojeni su na one segmente prikazivača koji su prikazani šrafirano na slici 3.24. S obzirom na tu činjenicu, biti će moguće osim izlaza pratiti i kako se mijenjaju stanja konačnog automata koja bi inače bila skrivena da dotični bistabili PAL-a nisu prospojeni na pripadne segmente prikazivača.

Najprije ćemo definirati funkciju prelaska iz stanja u stanje. Prvo stanje je inicijalno stanje INIT, iz kojeg se prelazi u drugo stanje, kada treba svijetliti lijevi segment, te ga stoga nazovimo stanje LIJEVO. Preostala stanja su prikazana na slici 3.24, a njihova imena su u skladu sa segmentom koji treba svijetliti uz pripadno stanje, pa su to dakle još DOLJE, DESNO, OPETDOLJE i GORE. Treba uočiti razliku između stanja DOLJE i OPETDOLJE. Iako u oba stanja treba biti pogonjen segment G, nije svejedno na koji način je to postignuto (da li prelaskom iz stanja LIJEVO ili iz stanja DESNO) pa takvu situaciju treba raščlaniti na dva stanja. Prelazi iz stanja u stanje ovise o aktivnosti linije VRTI (a) i b) dio slike 3.24). U oba nacrtana slučaja ulazna linija /RES nije bila aktivna iako i ona utječe na to kako se prolazi kroz određena stanja.

Funkciju prelaska stanja najjednostavnije je u našem slučaju prikazati dijagramom stanja kako je to učinjeno na slici 3.25. Prelazak iz stanja u stanje označen je strelicom koja kreće od prethodnog, a završava na slijedećem stanju. Mnemonici označeni uz

pojedine linije koje povezuju dva stanja označavaju uvjete koji moraju biti zadovoljeni za prelazak iz jednog stanja u drugo. Moguć je i slučaj da se ostaje u jednom stanju dok je određeni uvjet ispunjen (primjer koji veže stanje LIJEVO i liniju /RES).



Slika 3.25 Dijagram prelaska stanja

Analizirajmo sliku 3.25 za slučaj stanja LIJEVO. U stanje LIJEVO dolazi se iz stanja INIT. Isto tako u ovo stanje se uvijek dolazi i iz stanja OPETDOLJE, neovisno o stanju linije VRTI (vidi sliku 3.24), a i neovisno o liniji /RES zato jer je ovo stanje upravo stanje u koje se dolazi kada je linija /RES aktivna. Zato je uvjet kraj ove veze označen sa VCC (oznaka fiksne aktivnosti, napajanje) što označava da je to bezuvjetni prijelaz. Ako je linija /RES aktivna, sklop stalno ostaje u stanju LIJEVO. Iz svih ostalih stanja se u stanje LIJEVO dolazi uz aktivnu liniju /RES. Obratiti pažnju da su u dijagramu stanja linije prikazane kao neinvertirane kada su aktivne, a invertirane kada su neaktivne, neovisno o tome kako je dotična linija deklarirana u PAL sklopu. Zbog toga npr. uz vezu stanja LIJEVO i linije /RES piše RES, a ne /RES. Iz stanja LIJEVO se u stanje GORE ide kada nije aktivna linija /RES, ali kada je aktivan VRTI. Suprotno tome, u stanje DOLJE se ide kada sklop nije pod resetom, a linija VRTI je neaktivna. Iz stanja LIJEVO se ne može direktno doći u stanja DESNO i OPETDOLJE što se vidi i na dijagramu kao nepostojanje pripadnih linija sa strelicama.

Prema sintaksi PALASM-a, prelazak iz stanja u stanje (STATE TRANSITION), može se napisati na slijedeći način:

```
stanjeX    :=  uvjet1 -> stanje1
           +   uvjet2 -> stanje2
           +-> stanje_inače
```

Uvjeti mogu biti složeni izrazi kombinirani od više varijabli, pa se radi jednostavnosti mogu zapisati na jednom mjestu u obliku jedne varijable koja se zatim koristi u samim izrazima za prijelaz stanja. Uvjeti se pišu nakon ključne riječi CONDITIONS, a to je napravljeno i za ovaj primjer:

```
; uvjeti grananja
CONDITIONS
POCNIVRTI  =  /RES *  VRTI
NAZAD      =  /RES * /VRTI
```

Uvjeti prelaska pišu se za svako stanje tako da se definira pod kojim uvjetima i u koje slijedeće stanje se treba prijeći. Uvjeti prelaska za dijagram na slici 3.25 dani su u nastavku.

```
; jednadžbe prelaska stanja
INIT       :=  VCC -> LIJEVO        ; iz inicijalnog stanja odi
                                ; bezuvjetno u stanje LIJEVO

LIJEVO     :=  NAZAD -> DOLJE       ; ako je zadovoljen uvjet NAZAD
                                    ; odi u stanje DOLJE
            +   POCNIVRTI -> GORE   ;  ako je zadovoljen POCNIVRTI,
                                    ; odi u stanje GORE
            +-> LIJEVO              ; inače ostani lijevo

DOLJE      :=  NAZAD -> DESNO
            +-> LIJEVO

DESNO      :=  /RES -> OPETDOLJE
            +-> LIJEVO

OPETDOLJE  :=  VCC -> LIJEVO

GORE       :=  POCNIVRTI -> DESNO
            +-> LIJEVO
```

Pojedinim stanjima odgovaraju i pripadne vrijednosti izlaza. Tako na primjer, uz stanje LIJEVO treba svijetliti lijevi gornji segment prikazivača. Osim stanja kod Mealy-jevog tipa automata na izlaznu funkciju mogu utjecati i ulazi što je upravo i slučaj u ovom primjeru jer ulaz /INV djeluje na invertiranje prikaza na izlazima. Izlaznu funkciju također treba opisati kao i funkciju za prijelaz stanja, za svako stanje, jedina razlika je da iza svakog imena stanja dolazi izraz "**.OUTF**" što razlikuje jednadžbe stanja od jednadžbi izlaza. Za ovaj primjer treba napisati izraze za oba stanja linije /INV i to je dano u nastavku. U stanju INIT neovisno o stanju linije /INV ne smije svijetliti niti jedan segment.

```
; izlazne jednadžbe
; segmenti                  A       B       F       G

INIT.OUTF      =   INV -> /D[3] * /D[2] * /D[1] * /D[0] ; ne svijetle
               +->        /D[3] * /D[2] * /D[1] * /D[0] ; ne svijetle

LIJEVO.OUTF    =   INV ->  D[3] *  D[2] * /D[1] *  D[0] ; sv. svi osim F
               +->        /D[3] * /D[2] *  D[1] * /D[0] ; svijetli F

DOLJE.OUTF     =   INV ->  D[3] *  D[2] *  D[1] * /D[0]
               +->        /D[3] * /D[2] * /D[1] *  D[0] ; svijetli G
```

```
DESNO.OUTF      =   INV ->  D[3] * /D[2] *  D[1] *  D[0]
                +->        /D[3] *  D[2] * /D[1] * /D[0] ; svijetli B

OPETDOLJE.OUTF =   INV ->  D[3] *  D[2] *  D[1] * /D[0]
                +->        /D[3] * /D[2] * /D[1] *  D[0] ; svijetli G

GORE.OUTF       =   INV -> /D[3] *  D[2] *  D[1] *  D[0]
                +->         D[3] * /D[2] * /D[1] * /D[0] ; svijetli A
```

Na početku je rečeno da je od bistabila PAL sklopa 4 potrošeno na izlaze, a četiri su slobodna. Kao što se vidi iz dijagrama stanja i jednadžbi stanja, za realizaciju ovog automata potrebno je 6 stanja. Svakom od tih stanja može se pridijeliti kod (kombinacija) koja jednoznačno opisuje (definira) to stanje. Za 6 stanja, potrebna su 3 bita za kod, s tim da su dvije kombinacije od njih 8 nekorištene (ima više mogućnosti nego što nam treba). Npr. stanje LIJEVO može biti opisano pomoću tri bita kao npr. 0 1 0, stanje DESNO npr. 1 0 0 i tako redom mogu biti opisana sva stanja. Na temelju ove diskusije može se zaključiti da su za pamćenje stanja potrebna tri bistabila PAL sklopa, za svaki bit koda po jedan. S obzirom da su na laboratorijskoj maketi za rad s PAL-ovima svi izlazi PAL sklopa spojeni na segmente pokazivača, to znači da će i sadržaji bistabila koji pamte stanja biti vidljivi na pripadnim segmentima. Za stanja se dakle koriste bistabili spojeni na segmente C, D i E, a pripadne izlaze označimo sa SB[2], SB[1] i SB[0].

Programski paket PALASM tokom prevođenja automatski pridjeljuje kodove za pojedina stanja, pri čemu svaki kod odgovara nekoj od kombinacija na izlazima iz bistabila za stanja. Ovo se međutim može napraviti i ručno, prema željama i potrebama korisnika postupkom tzv. dodjeljivanja bitova za stanja (STATE ASSIGNMENT) pri čemu korisnik definira koji kod na izlazima bistabila odgovara kojem stanju. Kod postupka dodjeljivanja bitova za zapis stanja treba imati na umu slijedeće. Kod tipa PAL sklopa korištenog na vježbama se nakon uspostavljanja napajanja, svi izlazi bistabila automatski resetiraju, tj. postavljaju u stanje logičke nule. Između izlaza iz pojedinog bistabila i pripadnog pina PAL-a nalazi se inverter (vidi na Slici 3.4). S obzirom na ove invertere su kod uspostavljanja napajanja, a prije dolaska prvog impulsa na signalu takta, na izlaznim pinovima PAL-a logičke jedinice te tako u prvom trenu gore svi segmenti. Ovo stanje na izlazima SB[2..0] = [1 1 1] mora kod uspostavljanja napajanja odgovarati prvom stanju automata, koje je u ovom primjeru nazvano INIT. Ako su linije SB[2..0] deklarirane u pozitivnoj logici, to bi značilo da su kod inicijalizacije navedeni izlazi aktivni jer su u logičkoj jedinici te stanje INIT ima kod SB[2] SB[1] SB[0]. Nasuprot tome, izlaze se može deklarirati i kao invertirajuće, /SB[2..0], pri čemu se onda stanje INIT opisuje kao /SB[2] /SB[1] /SB[0], tj. uz ovakvu varijantu su u inicijalnom stanju izlazi neaktivni (iako jednaki logičkoj jedinici). To je u skladu sa često korištenom situacijom da se u nekom digitalnom dizajnu signali definiraju kao aktivni u logičkoj nuli. Obje varijante se ravnopravno mogu koristiti ovisno o željama i potrebama.

U našem dizajnu su varijable deklarirane u pozitivnoj logici. S obzirom da su vrijednosti varijabli stanja vidljive na nekim segmentima, kodovi stanja su pridijeljeni tako da se lakše mogu povezati sa vrijednostima izlaza u nekom stanju. Pa tako u stanju DOLJE od segmenata spojenih na bistabile stanja svijetli donji (tj. D), za stanje DESNO desni segment (tj. C), a ostali su odabrani na najzgodniji način da asociraju na pripadno stanje. Naravno, dodjela kodova za stanja je mogla biti izvedena na bilo koji drugi način. Pregledni zapis dodjele kodova slijedi:

```
                                                    segmenti (1=svijetli, 0=ne svijetli)
                                                          C       D       E
; dodjela kodova stanjima
INIT       =    SB[2] *  SB[1] *  SB[0]       ;  1       1       1
LIJEVO     =   /SB[2] * /SB[1] * /SB[0]       ;  0       0       0
DOLJE      =   /SB[2] *  SB[1] * /SB[0]       ;  0       1       0
DESNO      =    SB[2] * /SB[1] * /SB[0]       ;  1       0       0
OPETDOLJE  =    SB[2] *  SB[1] * /SB[0]       ;  1       1       0
GORE       =   /SB[2] * /SB[1] *  SB[0]       ;  0       0       1
```

Treba se podsjetiti da kod Mealy konačnog automata do promjene na izlaznim bistabilima dolazi nakon memoriranja novog stanja. Drugim riječima, na jedan rastući brid signala takta postavit će se novo stanje, a tek na slijedeći brid će se ovisno o prethodno postavljenom stanju postaviti izlazi, dakle izlaz kasni za jedan korak od pripadnog stanja. Tako će nakon prvog rastućeg brida na signalu takta, na D[3] do D[0] pojaviti vrijednosti izlaza koje odgovaraju stanju INIT, a to su prema zadatku sve nule neovisno o INV. Da bi se u tom slučaju postiglo da također niti jedan od donjih segmenata ne svijetli, stanju LIJEVO koje slijedi iza INIT pridijeljen je kod SB[2..0]=0 0 0. U tabeli 3.14 su radi boljeg pregleda objedinjene izlazne jednadžbe i dodjela bitova za prikaz stanja u tabelarnom obliku uz naznačene pripadajuće segmente. Nekorištena stanja označena su sa xxx.

Tabela 3.14 Popis stanja, kodova za zapis stanja te vrijednosti izlaza za stanja

INV	ime stanja	SB[2] (C)	SB[1] (D)	SB[0] (E)	D[3] (A)	D[2] (B)	D[1] (F)	D[0] (G)
0	INIT	1	1	1	0	0	0	0
0	OPETDOLJE	1	1	0	0	0	0	1
0	xxxxxx	1	0	1	xxx	xxx	xxx	xxx
0	DESNO	1	0	0	0	1	0	0
0	xxxxxx	0	1	1	xxx	xxx	xxx	xxx
0	DOLJE	0	1	0	0	0	0	1
0	GORE	0	0	1	1	0	0	0
0	LIJEVO	0	0	0	0	0	1	0
1	INIT	1	1	1	1	1	1	1
1	OPETDOLJE	1	1	0	1	1	1	0
1	xxxxxx	1	0	1	xxx	xxx	xxx	xxx
1	DESNO	1	0	0	1	0	1	1
1	xxxxxx	0	1	1	xxx	xxx	xxx	xxx
1	DOLJE	0	1	0	1	1	1	0
1	GORE	0	0	1	0	1	1	1
1	LIJEVO	0	0	0	1	1	0	1

Promjene na kombinacionim ulazima /RES, INV i VRTI se u principu događaju asinkrono od rada automata. Da bi dizajn bio korektno napravljen, sva tri ulaza bi trebalo sinkronizirati, tj. dovesti preko bistabila kako je to opisano u uvodnom dijelu (vidi sliku 3.23). S obzirom da za to za sva tri ulazna signala više u ovom tipu PAL-a nema dovoljno resursa (ostao je samo jedan neiskorišteni bistabil), preko registra je doveden samo signal VRTI.

To je realizirano na slijedeći način. Nazovimo stvarni kombinacioni asinkroni ulaz u PAL sa VRTI_IN. Ovaj signal dovodi se na ulaz preostalog bistabila, a izlaz tog bistabila VRTI želimo dalje koristiti u izrazima za prelazak kroz stanja. Ovaj izlaz spojen je na segment za prikaz decimalne točke pa promjene ovog signala možemo pratiti na tom segmentu. Kada je VRTI neaktivan (odnosno VRTI_IN neaktivan), segment ne svijetli, a svijetli kada je aktivan. Istovremeno se koristeći postojeću povratnu petlju s izlaza registra na ulaz u PAL, signal VRTI koristi kao ulazni signal u uvjetima za grananje iz stanja u stanje. S obzirom da je pravi ulaz VRTI_IN proveden preko bistabila, svaka njegova promjena će se detektirati na promjeni stanja (da li vrti ili ljulja) tek na slijedećem rastućem bridu signala takta. Na prvom se tek vrijednost ulaza zapisuje u bistabil (generira odgovarajući VRTI), a tek na slijedećem taj ulaz sudjeluje u kombinacionoj funkciji za generiranje novog stanja. Navedeno rješenje opisano je slijedećim izrazima.

```
;----------------------- Pin Declarations -----------------
PIN  2        VRTI_IN                  COMBINATORIAL ; INPUT
PIN  15       VRTI                     REGISTERED ; OUTPUT

;----------------------  Boolean Equation Segment ---------
EQUATIONS
VRTI = VRTI_IN
```

Pripadni pinovi se deklariraju kako je navedeno u dijelu za deklaracije pinova. Ostvarenje povratne petlje sa VRTI i VRTI_IN se zapisuje u dijelu za Bool-ove izraze. I Bool-ovi i izrazi vezani uz konačni automat mogu se pojavljivati u istom dizajnu bez obzira na redoslijed, ali moraju biti napisani u dijelu iza ključne riječi EQUATIONS za Boole-ove, odnosno iza STATE za izraze za konačni automat.

3.9.2 ZADATAK 1.

Pomoću sinkronog Moore-ovog automata (vidi sliku 3.15) realizirati slijedeću funkciju. Na sedamsegmentnom prikazivaču samo jedan svijetleći segment treba kružiti po prikazivaču tako da ispisuje broj 8. Prvo stanje je stanje INIT u koje sklop ide nakon uspostavljanja napajanja. Nakon toga ide u stanje kada je upaljen gornji segment. Jednom dodatnom ulaznom linijom SMJER treba omogućiti promjenu smjera kruženja tj. kada je ta linija neaktivna (u nuli) kruženje od gornjeg segmenta kreće u smjeru kazaljke na satu, odnosno aktivna (u jedinici) uvjetuje kruženje u suprotnom smjeru. Razlikovati stanja za prolazak kroz srednji segment. Kod drugog prolaska kroz srednji segment (na povratku) treba osim pripadnog segmenta upaliti i decimalnu točku. Imena pinova (signala) odabrati po želji. Raspored pinova i signala dan je u tabeli 3.15.

PREPORUKE: Za Moore-ov tip automata dovoljno je osim izraza za prijelaz stanja napisati još samo izraze za dodjelu kodova stanjima, a ne treba napisati izlazne jednadžbe jer postojeći bistabili služe i za pamćenje stanja i za izlaze.

Tabela 3.15 Raspored ulaznih i izlaznih pinova na PAL-u za Zadatak 1.

ULAZI		IZLAZI	
broj pina PAL-a	signal	broj pina PAL-a	signal
11	/OE	17	SA
1	CLK	16	SB
2	SMJER	14	SC
		13	SD
		12	SE
		18	SF
20	VCC	19	SG
10	GND	15	SDP

a) <u>Nacrtati dijagram prelaska iz stanja u stanje (koristiti prazni prostor ispod):</u>

b) Realizirati konačni automat sklopom PALCE16V8

- Pomoć pri obavljanju pojedinih zadataka kao i sintaksu programskog alata PALASM možete naći u poglavljima 3.2 i 3.8.
- Raditi u već postojećem direktoriju **C:\PALASM\PURS*prezime***.
- Pozvati PALASM i ako je potrebno podesiti radni direktorij.
- Da bi se ubrzao rad, dozvoljeno je kopiranje i mijenjanje već ostvarenih datoteka, ali na kraju moraju postojati datoteke sa sva tri zadatka vezana uz dizajn pomoću PAL-ova. Ako se datoteka kopira, **paziti** da se u njoj promijeni sve što je potrebno (da ne bi ostali neki podaci od prije).
- Nakon otvaranja datoteke deklarirati pinove prema tabeli 3.15.
- Nakon unošenja ključnih riječi za dizajn konačnim automatom i tip automata, najprije napisati jednadžbu za prelazak iz stanja INIT nakon uspostavljanja napajanja u prvo "normalno" stanje.
- Napisati ostale jednadžbe za prelazak stanja.
- Napisati izraze za dodjelu bitova stanja (STATE ASSIGNMENT). Izlazne jednadžbe ne pisati.
- U zadatku ne treba pisati CONDITION izraze jer su uvjeti grananja jednostavni.
- Ne treba realizirati simulaciju.
- Prevesti dizajn i ispraviti eventualne pogreške.

c) Provjeriti ispravnost rada PAL-a na maketi

- Provjeriti rad sklopa kao što je već prije opisano.

3.10 Vježba 7 - dodatak: Ukupno rješenje primjera opisanog u vježbi

a) Primjer 1.

```
;PALASM Design Description
;---------------------------------- Declaration Segment ------------
TITLE    SEGMENT IDE LIJEVO-DESNO ILI KRUZI, MEALY STATE MACHINE
PATTERN  STATEME1
REVISION 1.0
AUTHOR   Davorka Petrinovic
COMPANY  FER/ZESOI
DATE     11/07/98

CHIP  STATEME1  PALCE16V8

;---------------------------------- PIN Declarations ---------------
PIN  1                  CLK                   COMBINATORIAL ; CLOCK
PIN  2                  VRTI_IN               COMBINATORIAL ; INPUT
PIN  3                  INV                   COMBINATORIAL ; INPUT
PIN  9                  /RES                  COMBINATORIAL ; INPUT
PIN  10                 GND                                 ;
PIN  20                 VCC                                 ;
PIN  17,16,18,19        D[3..0]               REGISTERED ; OUTPUT
PIN  14..12             SB[2..0]              REGISTERED ; OUTPUT
PIN  15                 VRTI                  REGISTERED ; OUTPUT
PIN  11                 /OE                   COMBINATORIAL ; ENABLE

;---------------------------------- Boolean Equation Segment ------
EQUATIONS
VRTI = VRTI_IN    ; asinkroni ulaz VRTI_IN pamti se u bistabilu te se
                  ; dalje koristi izlaz iz bistabila signal VRTI kao
                  ; ulaz za odabir načina rada: da li jedan segment
                  ; kruži ili se ljulja lijevo - desno.
                  ;   Kada   je   izlaz   VRTI   neaktivan   (ljuljanje),
                  ;  decimalna  točka  ne  svijetli,  odnosno  kad  je
                  ; aktivan svijetli.

;---------------------------------- State Segment ------
STATE
MEALY_MACHINE


; počinju TRANSITION izrazi
;--------------------------
INIT              :=  VCC -> LIJEVO

LIJEVO            :=  NAZAD -> DOLJE
                  +   POCNIVRTI -> GORE
                  +-> LIJEVO

DOLJE             :=  NAZAD -> DESNO
                  +-> LIJEVO

DESNO             :=  /RES -> OPETDOLJE
                  +-> LIJEVO

OPETDOLJE         :=  VCC -> LIJEVO

GORE              :=  POCNIVRTI -> DESNO
                  +-> LIJEVO
```

```
; počinju OUTPUT izrazi
;---------------------------
; ako ulaz INV nije aktivan, svijetli samo jedan segment, u protivnom
; prikaz je inverzan, tj. samo aktivni ne svijetli


INIT.OUTF         =   VCC -> /D[3] * /D[2] * /D[1] * /D[0]

LIJEVO.OUTF       =   INV ->  D[3] *  D[2] * /D[1] *  D[0]
                  +->        /D[3] * /D[2] *  D[1] * /D[0]

DOLJE.OUTF        =   INV ->  D[3] *  D[2] *  D[1] * /D[0]
                  +->        /D[3] * /D[2] * /D[1] *  D[0]

DESNO.OUTF        =   INV ->  D[3] * /D[2] *  D[1] *  D[0]
                  +->        /D[3] *  D[2] * /D[1] * /D[0]

OPETDOLJE.OUTF    =   INV ->  D[3] *  D[2] *  D[1] * /D[0]
                  +->        /D[3] * /D[2] * /D[1] *  D[0]

GORE.OUTF         =   INV -> /D[3] *  D[2] *  D[1] *  D[0]
                  +->         D[3] * /D[2] * /D[1] * /D[0]


; počinju STATE ASSIGNMENT izrazi
;--------------------------------

                                                    SB[2]  SB[1]  SB[0]
INIT              =   SB[2] *  SB[1] *  SB[0]    ;  1      1     1
LIJEVO            =  /SB[2] * /SB[1] * /SB[0]    ;  0      0     0
GORE              =  /SB[2] * /SB[1] *  SB[0]    ;  0      0     1
DOLJE             =  /SB[2] *  SB[1] * /SB[0]    ;  0      1     0
DESNO             =   SB[2] * /SB[1] * /SB[0]    ;  1      0     0
OPETDOLJE         =   SB[2] *  SB[1] * /SB[0]    ;  1      1     0


; počinju CONDITION izrazi
;--------------------------------

CONDITIONS
POCNIVRTI         =  /RES *  VRTI
NAZAD             =  /RES * /VRTI
```