2.6 DIGITALNA VEZJA NA OSNOVI PROGRAMIRLJIVE LOGIKE

2.6.1 ENOSTAVNA PROGRAMIRLJIVA LOGIČNA POLJA-SPLOŠNO

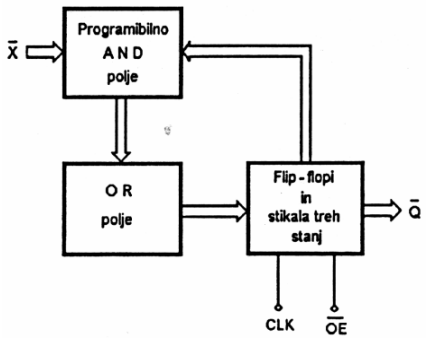
Za enostavna programirljiva logična polja je značilno, da so sestavljena iz matrične strukture, ki vsebuje komponente-vrata logičnega polja IN in iz matrike komponent logičnega polja ALI. Vse komponente v obeh matričnih strukturah so s številnimi programirljivimi povezavami med sabo tudi univerzalno povezane (vse možne kombinacije). S pomočjo programiranja ima uporabnik možnost, da v programirljivem delu logičnega polja s pomočjo programatorja in ustrezne programske opreme vzpostavi oz. prekine potrebne povezave. Na ta način se nepotrebne oz. škodljive povezave prekinejo, ostanejo pa tiste, ki omogočajo realizacijo željenih logičnih funkcij. S takšnimi vezji je mogoče načrtovati zaključene sklope digitalnih vezij v enem samem integriranem vezju.

Glede na to, katera od obeh matrik je fiksna (uporabnik je ne more spreminjati) in katera je programirljiva ločimo: ***PAL*** (programirljivo polje IN, fiksno polje ALI), ***PROM*** (fiksno polje IN, programirljivo polje ALI) in ***PLA*** (programirljivo polje IN, programirljivo polje ALI). Vse tri vrste PROM, PAL in PLA so izvedene v bipolarni tehniki in za katere je značilno, da se pri postopku programiranja neželjene povezave prekinjajo. Vsaka povezava v matriki vsebuje tudi oslabljeno točko, ki jo je mogoče pri postopko programiranja trajno prekiniti s povečanim tokom. Zaradi fizičnih prekinitev postopek seveda ni reverzibilen in je zato možno le enkratno programiranje pri katerem se sprošča tudi precej toplote. V elektronskih vezjih jih najdemo v obliki karakter generatorjev, kodnih pretvornikov, kodnih ključev ipd., vendar niso več primerna za novo vgradnjo . V novejšem času so jih izpodrinila sodobnejša in zmogljivejša programirljiva vezja tipa ***GAL***.

GAL (*Generic Array Logic*) so programirljiva logična polja v MOS tehnologiji, za katere je značilno, da so za križne povezave v matričnem polju uporabljeni MOSFET transistorji, ki so normalno vsi v neprevodnem stanju . Glede na to, da lahko s pomočjo programiranja definiramo stanje prevodnosti MOSFET transistorja, je mogoče večkratno spreminjanje zakonitosti logičnih funkcij – reverzibilen proces. Poleg tega imajo GAL vezja vgrajeno na izhodu še makro celico, kateri je prav tako možno s programiranjem določiti način delovanja. Ker makro celice vsebujejo tudi flip-flope je možno z GAL vezji izvesti tudi različna sekvenčna vezja, poleg tega pa omogočajo, da lahko izhode uporabimo tudi kot vhode.

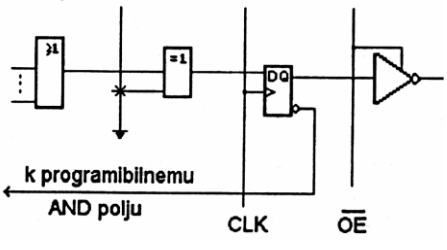
Arhitektura programirljivih vezij

Glede na to, da programirljiva vezja vsebujejo veliko število medsebojnih povezav, bi postalo vezje popolnoma nepregledno, če bi vrisali vse črte. Zato je v uporabi simboličen način ponazarjanja večjega števila povezav s skupnimi lastnostmi, vendar pa je možno preko zunanjih priključkov vsako od povezav enoumno nasloviti in jo programirati. Obstaja veliko različnih izvedb teh vezij, vse z namenom, da bi bila čim bolj univerzalno uporabljiva. Nekatera vezja omogočajo tudi notranje generiranje signalov kot npr. *output enable* (OE) ali urinih impulzov (CLK). Zelo uporabljiva lastnost je tudo programirljiva izhodna polariteta.



Slika 2.20: Blokovna struktura programirljivega digitalnega vezja

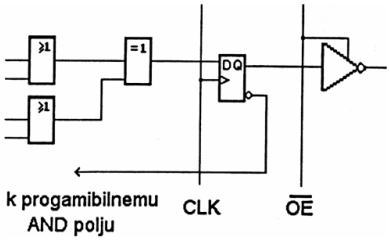
Slika 2.21 prikazuje del programirljivega vezja, pri katerem je poleg ostalega programirljiv tudi en vhod XOR člena, kar omogoča negacijo. Ta vhod je normalno priključen na potencial masein v tem primeru deluje XOR le kot vmesnik-*buffer*. V primeru programiranega stanja na drugem vhodu XOR v logično 1 (prekinjena povezava) pa postane XOR v funkciji negatorja vhodnega signala.



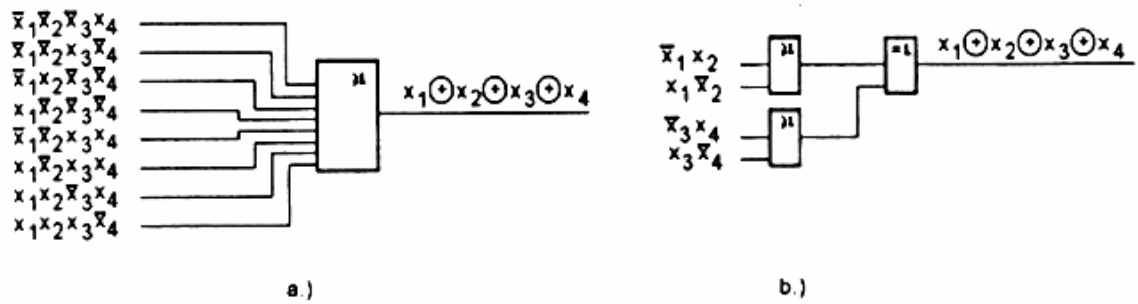
Slika 2.21: XOR v funkciji programabilne izhodne polaritete

Programirjiva izhodna polariteta pa ni ugodna samo navzven, temveč je lahko dobrodošla tudi v smislu poenostavitev logičnih izrazov. Pogosto se namreč zgodi, da zaradi omejenega števila vhodov v polju OR določene funkcije ne moremo realizirati, ker ima le ta v vsoti preveč členov konjunkcij. V takem primeru je smiselno preveriti, če nima morda negirana funkcija manjše število členov, kar pa ne predstavlja več ovire za realizacijo.
Izhodno funkcijo je seveda še potrebno negirati z XOR .

Obstajajo pa tudi izvedbe, ki imajo na istem mestu XOR s fiksno povezanimi vhodi na izhode OR vezij in predstavljajo zaporedno strukturo AND-OR-XOR (slika 2.22). Prednost take arhitekture si lahko najbolje ogledamo na primeru realizacije XOR funkcije med štirimi spremenljivkami: $X_1 \oplus X_2 \oplus X_3 \oplus X_4$.

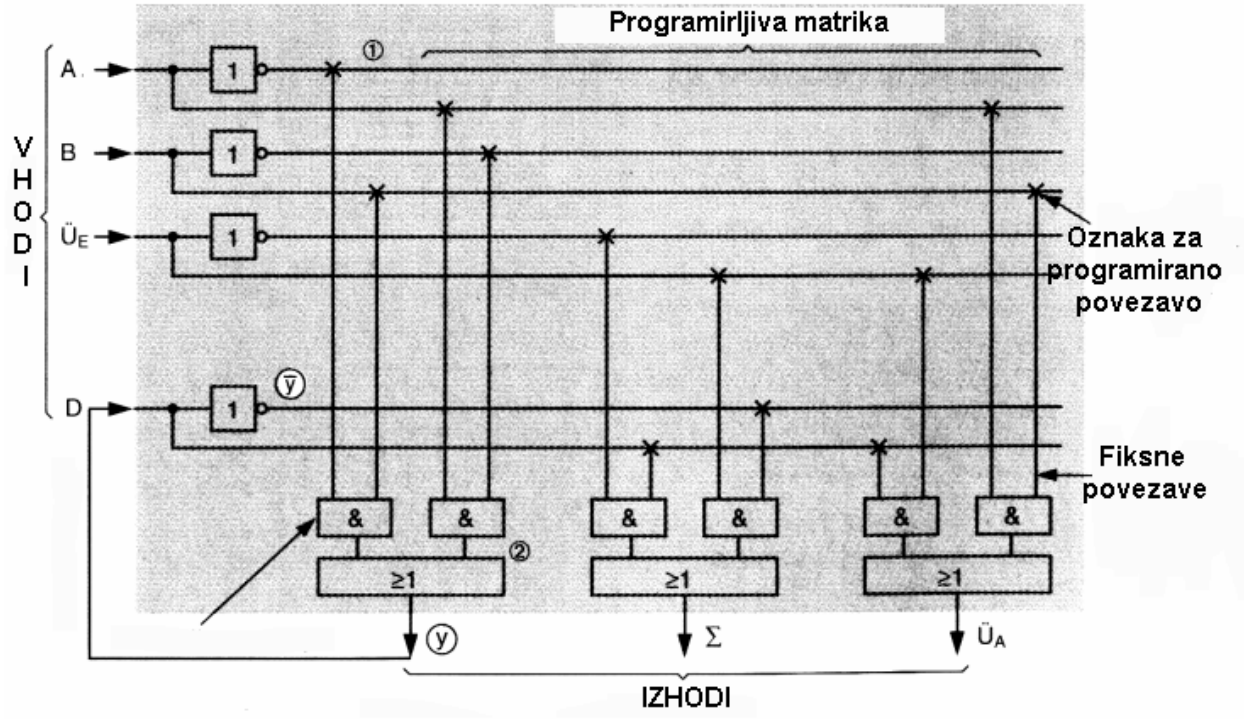


Slika 2.22: AND-OR-XOR arhitektura programirljivega vezja



Slika 2.23: Različni izvedbi logičnega vezja za enako funkcijo

Na sliki 2.23 vidimo, da bi pri realizaciji z navadnim vezjem potrebovali 8-vhodni OR (a), če pa uporabimo kombinacijo AND-OR-XOR pa zadostujeta že dva dvovhodna OR elementa (b). Razlika je očitna!

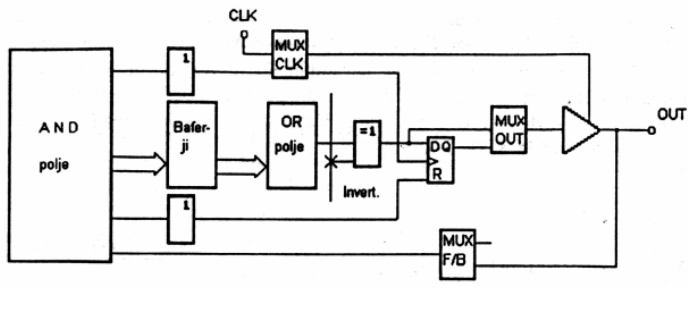


Slika 2.24: Primer programiranih povezav za popolni seštevalnik v izvedbi s PLA vezjem

2.6.2 GAL - PROGRAMIRLJIVA VEZJA

Za razliko od družine PAL vezij, ki so v bipolarni tehnologiji, so GAL vezja v CMOS tehnologiji, kar omogoča še večjo zmogljivost in imajo namesto navadnih spominskih celic tako imenovane makro celice. v makro celici jepoleg spominskih celic še nekaj programirljivih multipleksorjev, ki omogočajo spreminjanje konfiguracije makro celice, kar pomeni, da lahko uporabnik sam izbira arhitekturno zasnovo sekvenčnega dela vezja. Glede na proizvajaca obstajajo določene razlike v arhitekturni zasnovi makro celice.

Makro celica na sliki (*Altera*) ima vgrajene tri multipleksorje, ki omogočajo neodvisno programiranje izhodnih linij, povratnih zvez in taktnih impulzov za vsako celico posebej. Selekcijski vhodi multipleksorjev-MUX so krmiljeni z izhodi EEPROM vezja (na sliki niso prikazani).



Slika 2.25: Arhitektura makro celice firme Altera

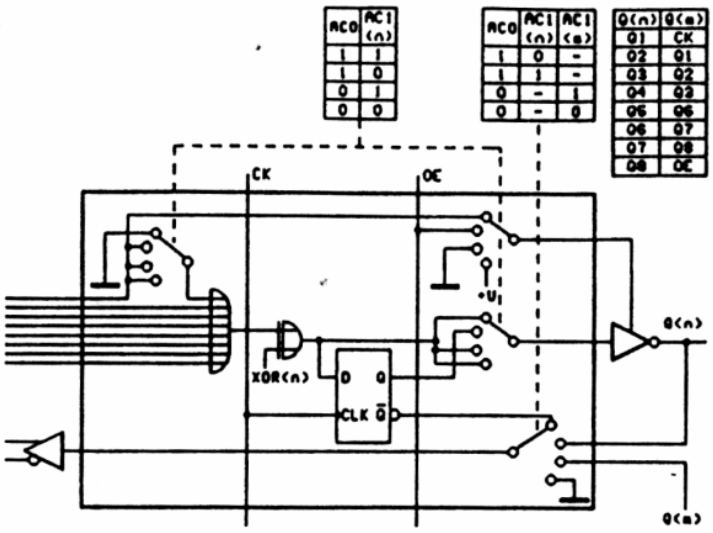
Izhodni MUX *OUT* določa ali bo izhod sekvenčen ali kombinacijski (kombinacijski bo takrat ko bo namesto izhoda izbran vhod v spominsko celico). MUX *F/B* v povratni vezavi odloča o tem ali bo vhod v AND polje priključen na izhod spominske celice ali pa na zunanji signal. Na ta način lahko programiramo zunanje priključke tako, da so vhodi ali pa izhodi in za vsako celico posebej.

Nekatere izvedbe imajo v celici dve povratni vezavi, kar omogoča, da lahko izkoristimo spominsko celico za notranje stanje, pripadajoč priključek (izhod) pa kot neodvisen vhod, kar bistveno poveča izkoriščenost vezja. MUX *CLK* omogoča izbiro urinega impulza med zunanjim ali notranjim - v makro celici generiranem signalu. Ker je drugi urin (taktni) vhod generiran z neko konjunkcijo, je takt lahko tudi kombinacija neodvisnih vhodov ali pa zunanjega taktnega signala.

Režimi nastavitev makro celice

Celico programiramo s pomočjo bitov SYN, AC=, AC1(n) in XOR(n). Bita SIN in AC= imata vpliv na vsa makro celice istočasno, medtem ko lahko bita AC1(n) in XOR(N) določimo za vsako celico posebej.

Izhodi AND vezij so priključeni na OR vezje z 8 vhodi.Eden od izhodov je lahko priključen tudi na trinivojski inverter na izhodu. Stanje bitovAC= in AC! vpliva na položaj »preklopnikov«. XOR bit vpliva na XOR vezje: 0 stanje prepušča, 1 pa invertira. SYN bit v stanju 0omogoči povezave CK in OE. $Q_{(n)}$ predstavlja izhod makro celice, $Q_{(m)}$ pa je izhod sosednje makro celice (glej tabelo na sliki 2.26).

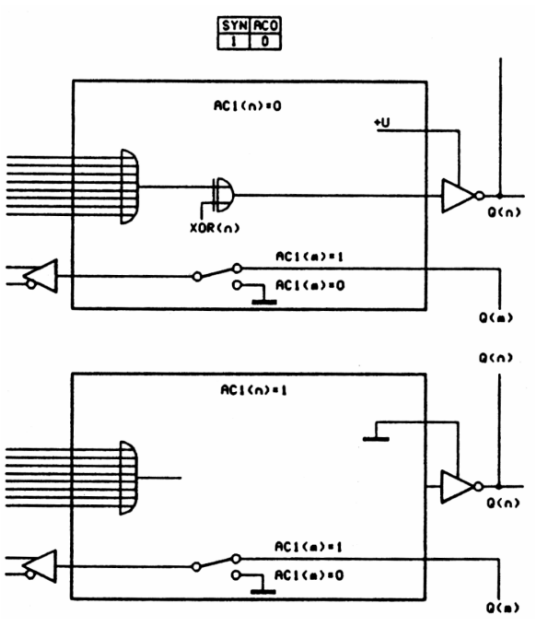


Slika 2.26: Primer izhodne logične makro celice

Normalni vhod ali izhod.

Pri tem načinu je stanje izhoda odvisno od stanj na vhodih in na povratnih vezavah. Izhod pa je lahko tudi onemogočen, zato se v tem primeru izhodna sponka uporabi kot vhod in je priključena na vhod povratne vezave naslednje makro celice.

Bit SYN je na logični1, bit AC= pa na logični 0. XOR bit invertira izhod OR vezja, če je na 1, sicer pa prepušča. Če je je bit AC! na 0, je izhod omogočen, če pa je AC1 na 1, je onemogočen in ga lahko uporabimo kot vhod. Stanje bita AC1 sosednje celice določa povezavo vhoda povratne vezave na maso ali na izhod sosednje makro celice. Sosednja makro celica je zgornja za makrocelice 2,3 in 4, ter spodnja za celice 5,6 in 7. Vhod povratne vezave skrajnih makro celic je priključen na sponko CK ali OE. Za dodatne vhode uporabimo najprej zunanje makro celice.

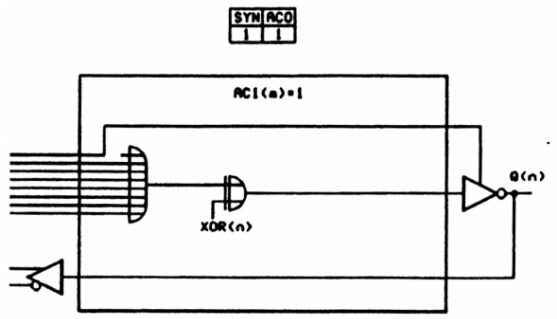


Slika 2.27: Makro celica v režimu : normalni izhod (zgoraj) normalni vhod (spodaj)

Kontrolirani izhod

V tem načinu je eden od vhodov OR vezja uporabljen za kontrolo izhoda (ali je izhod aktiven ali pa v stanju visoke impendance). Ostalih 7 vhodov OR vezja pa vpliva na logično stanje izhoda.

Biti SYN, AC= in AC! so na logični 1. XOR bit invertira izhod OR vezja, če je na 1, sicer pa normalno prepušča. Izhodni inverter je priključen na vhod povratne vezave, razen pri skrajnih makro celicah, kjer je povratna vezava priključena na vhod CK oziroma OE.

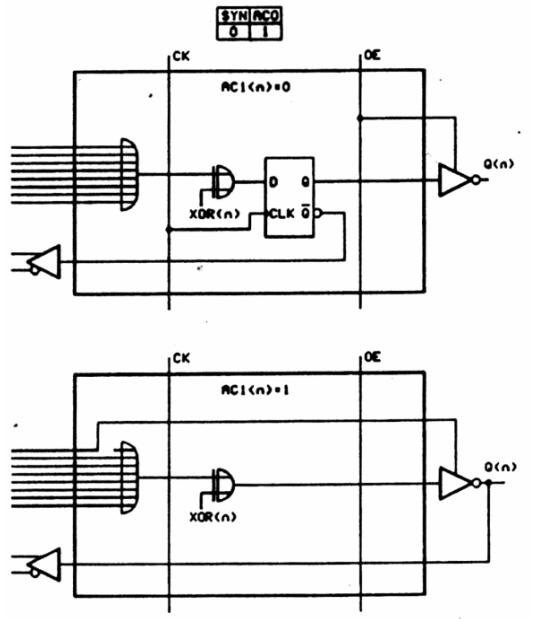


Slika 2.28: Makro celica v režimu: kontrolirani izhod

Registrirani ali kontrolirani izhod

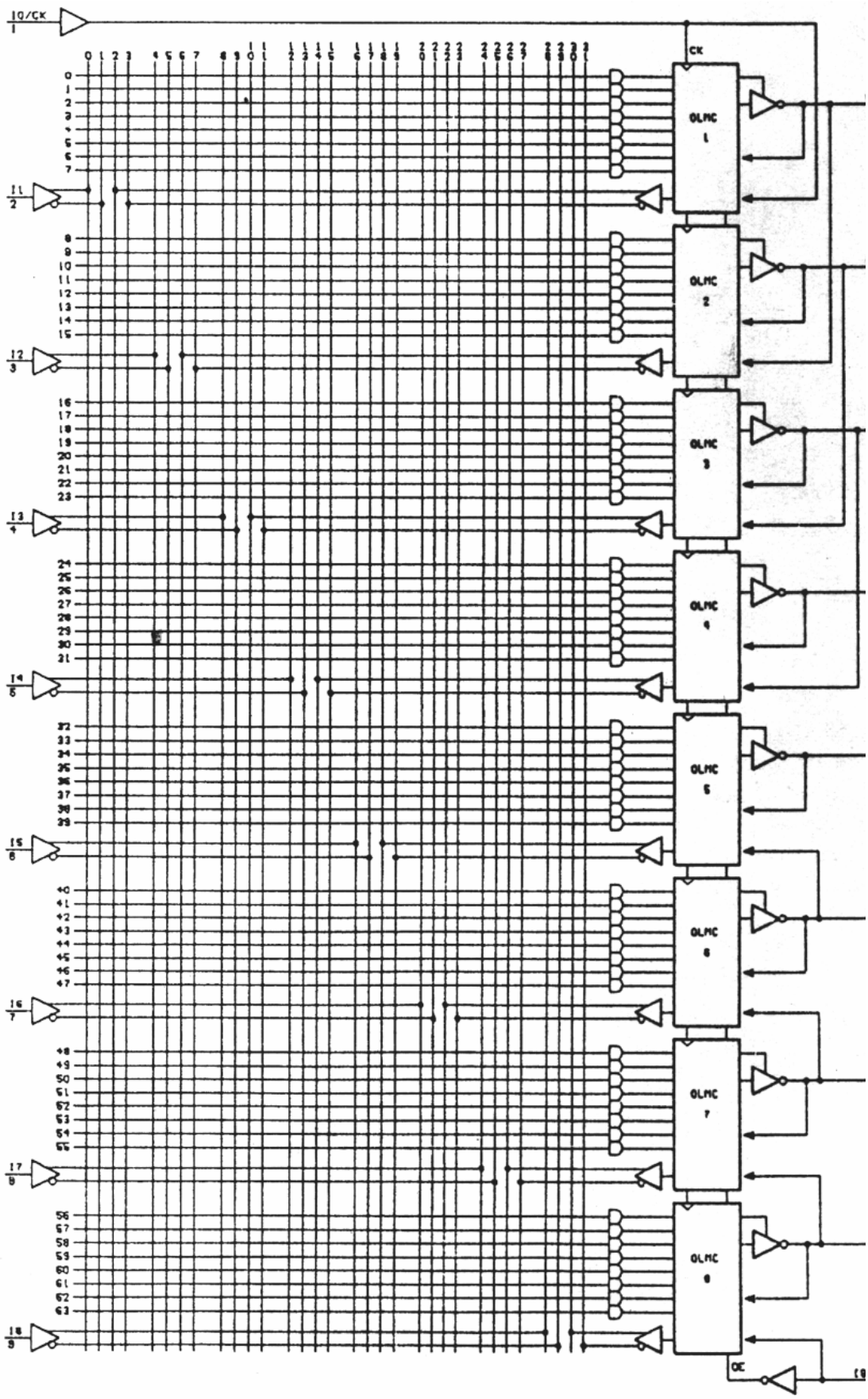
Pri registriranem izhodu se se stanje izhoda spremeni pri prehodu vhoda CK iz 0 na 1 in če je izhod omogočen s pomočjo vhoda OE. Kontrolirani izhod pa deluje enako kot v prejšnjem primeru.

Če je bit AC! v stanju 1, deluje izhod enako kot v prejšnjem primeru. Če pa je bit AC1 na 0, je izhod registriran. To pomeni, da se informacija iz OR vezja in XOR vezja prenese v D flip -flop ob pozitivnem pragu na vhodu CK(clock), ter se ne spremeni do naslednje spremembe iz 0 na 1 na vhodu CK. Stanje D-flip flopa se prenese na izhod, če je signal OE na 1, oziroma če je pin 11 na 0. Sicer je izhod v stanju visoke impendance. Tudi v tem primeru je lahko XOR uporabljen za eventuelno potrebo po invertiranju.



Slika 2.29: Registrirani izhod (zgoraj), kontrolirani izhod (spodaj)

PRILOGA 6: Primer arhitekture notranjih povezav v GAL vezju (GAL 16V8)



2.7.1.1 Programiranje GAL oziroma PALCE vezij

GAL vezja lahko progamiramo s pomočjo ustreznega programatorja, katerega priključimo na računalnik običajno preko paralelnega vodila LPT1. Na računalniku mora biti naložena tudi pripadajoča programska oprema. Programatorje in programsko opremo ponudniki programirljivih vezij neprestano posodabljajo, kar omogoča uporabniku vedno lažje delo. Za primer si oglejmo postopek programiranja s pomočjo programa, ki deluje še v DOS okolju.

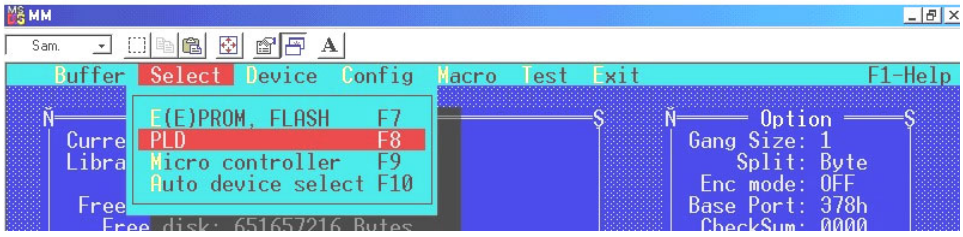
S pomočjo »Mega Max« programatorja je možno programirati različne vrste programibilnih vezij kot so logična polja, mikrokontrolerji in EPROM-i različnih proizvajalcev. V nadaljevanju bo opisan postopek programiranja GAL vezij tipa PALCE.

Ker deluje pripadajoči program v DOS okolju, ne moremo uporabiti miške za izbiranje ukazov ali zaslonskih oken znotraj programskega okna temveč s pomočjo tipke TAB, smernimi tipkami in tipko ESC, s katero zapustimo predhodni meni. Program lahko napišemo v obliki tekstovne datoteke po pravilih za program z nekim urejevalnikom besedila (npr. WordPad). V tekstovni datoteki so lahko pod ustrezno oznako zapisani tudimkomentarji, ki pa se kasneje ne prevajajo v jezik, ki ga programator razume. Komentar je zgolj za lažje kasnejše popravke oz. za pomoč pri orientaciji v programu. Napisani tekstovni program se nato s pomočjo GAL assemblerja prevede v obliko (JEDEC datoteka), ki jo programator »razume« in katera definira vsa mesta, ki morajo biti »programirana«

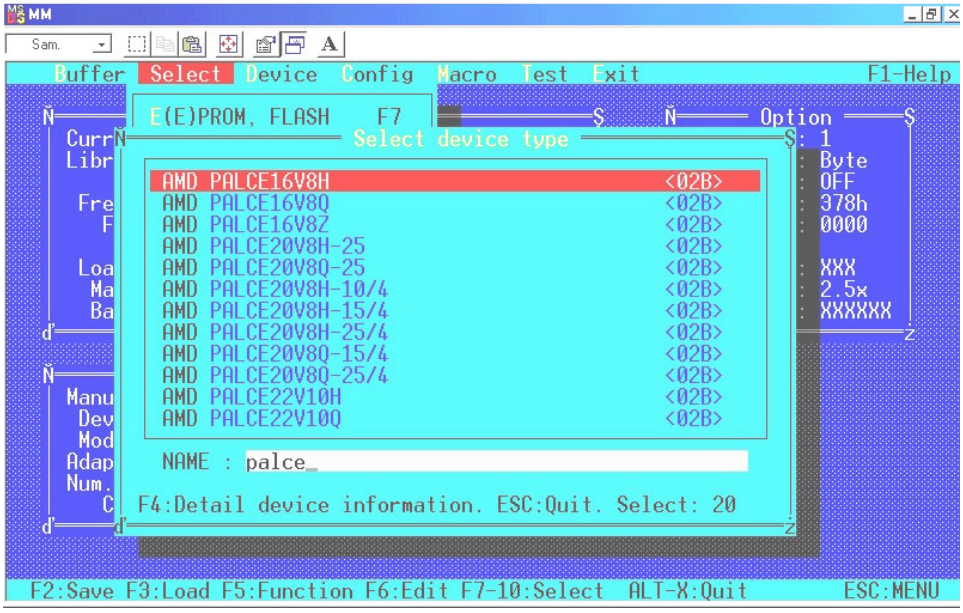
.

Postopek programiranja je sledeč:

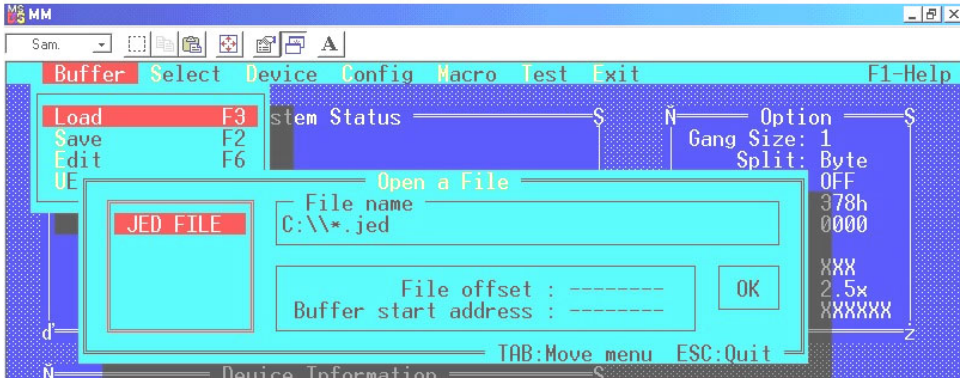
1. Najprej je potrebno v meniju **Select/PLD** izbrati ustezen tip komponente oz. **PLD (P**rogrammable **L**ogic **D**evice) npr. **PALCE16V8H**, kot kaže slika spodaj.



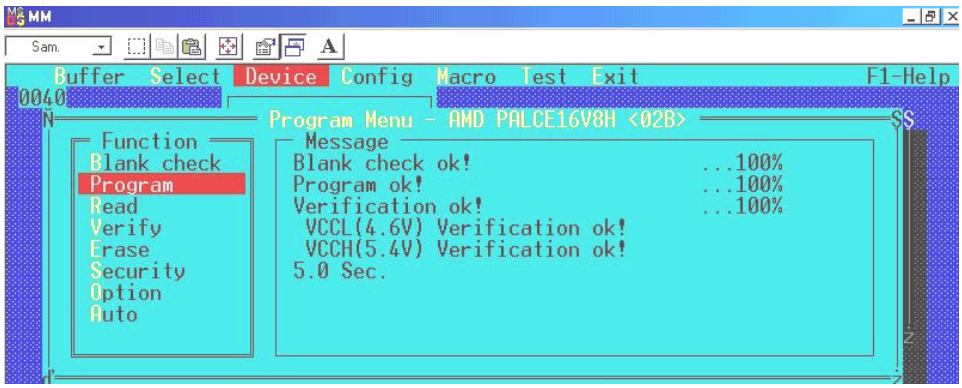
2. V programator je potrebno vstaviti ustrezen vtični modul, ki je potreben za to skupino vezij. Oznaka modula se vidi v desnem delu zaslonskega okna (Module Name) <02B> na sliki spodaj.



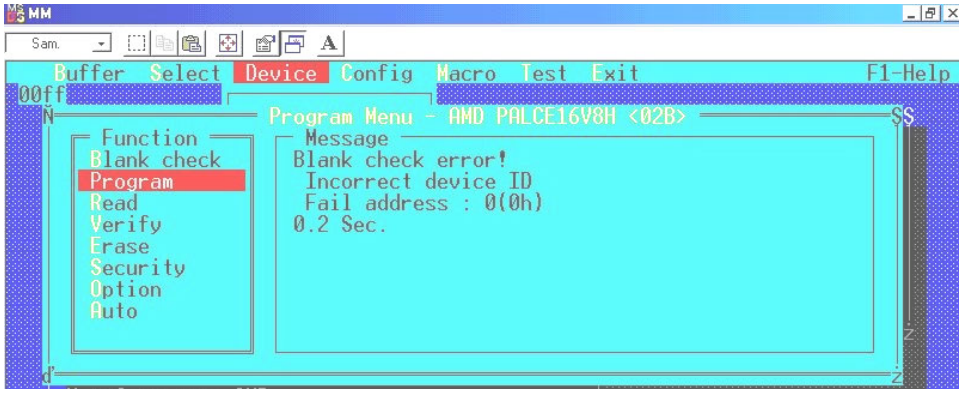
3. Zatem z ukazom **Buffer/Load** naložimo vsebino prevedenega programa v JEDEC datoteko, v začasni pomnilnik. Postopek programiranje lahko pričnemo z ukazom **Device/Function/Program.**



Pri postopku programiranju program najprej samodejno preveri, če je vsebina integriranega vezja prazna, potem ga programira in na koncu izvede še verifikacijo oz. preveri, če se programirana vsebina v vezju ujema z vsebino v začasnem pomnilniku.



V primeru da je integrirano vezje že bilo uporabljeno oz. že vsebuje program, je tega potrebno predhodno izbrisati z ukazom **Device/Function/Erase**. Pred programiranjem je možno tudi preveriti ali je »vsebina« integriranega vezja prazna z ukazom **Device/Function/Blank check**.. Poskus programiranja integriranega vezja, ki ni »prazno« bo program zavrnil z obvestilom **Blank check error!**, kar je razvidno iz slike spodaj. V primeru, da se želimo prepričati, če je vezje pravilno programirano uporabimo ukaz: **Device/Function/Verify.** Lahko si ogledamo tudi matrično polje ki pripada notranjosti iz katerega so vidne programirane lokacije.



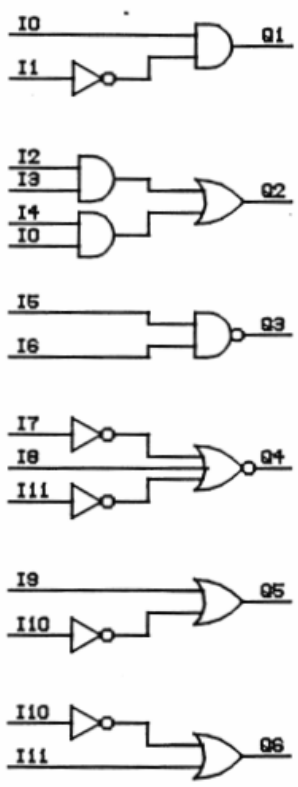
Programiranje drugih programirljivih vezij poteka na podoben način.

2.7.1.2 Primeri enostavnih programov

1. ***Program za logično vezje z normalnimi izhodi***

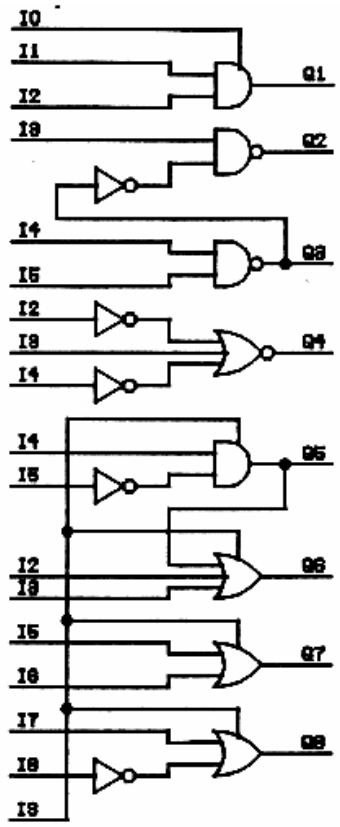
V tem primeru sta izhoda Q8 in Q7 (jih ne potrebujemo!) uporabljena kot vhoda I10 in I11, vendar zato v tem načinu niso možne povratne vezave iz teh izhodov.

```
Normalni izhodi      ;Header programa

CHIP norm GAL16V8    ;Deklaracijski blok

;pin   1   2   3   4   5   6   7   8   9   10
       I0  I1  I2  I3  I4  I5  I6  I7  I8  GND

;pin  11  12  13  14  15  16  17  18  19  20
       I9 I10 I11  Q6  Q5  Q4  Q3  Q2  Q1  VCC

ŽUES NORM0194        ;user electronic signature

EQUATIONS            ;Funkcijski blok
     Q1 = I0*/I1
     Q2 = I2*I3 + I4*I0
     /Q3 = I5 + I6
     /Q4 = /I7 + I8 + /I11
     Q5 = I9*/I10
     Q6 = /I10 + I11
```



Slika 2.29:Programirano vezje glede na pripadajoči program

2. Logično vezje z kontroliranimi izhodi in pripadajoči program



Slika 2.30: Programirano vezje glede na pripadajoči program

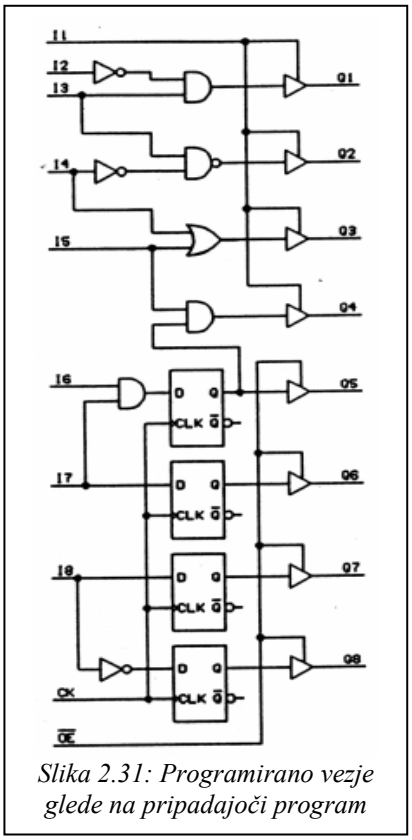
V tem načinu so izhodi kontrolirani, kar pomeni, da so lahko v stanju visoke impedance. posamezen izhod je omogočen z programiranim vhodom (I0 in I9). V tem načinu lahko izhode uporabimo tudi za povratno vezavo, razen izhodov Q1 in Q8, ker sta na povratni vezavi skrajnih makro celic, že priključena signala I0 (CK) in I9 (OE).

```
Kontrolirani izhodi    ;Header programa

CHIP kontr GAL16V8     ;Deklaracijski blok

;pin 1   2   3   4   5   6   7   8   9   10
    I0  I1  I2  I3  I4  I5  I6  I7  I8  GND

;pin 11  12  13  14  15  16  17  18  19  20
     I9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  VCC

ŽUES KONT0194      ;user electronic signature

EQUATIONS          ;Funkcijski blok
     Q1 = I1 * I2
     /Q2 = I3*/Q3
     /Q3 = I4 * I5
     /Q4 = /I2 + I3 + /I4
     Q5 = I4*/I5
     Q6 = I2 + I3*Q4
     Q7 = I5 + I6
     Q8 = I7 + /I8
     Q1.OE = I0
     Q5.OE = I9
     Q6.OE = I9
     Q7.OE = I9
     Q8.OE = I9
```

3. Primer programa za <u>logično vezje z registriranimi izhodi</u>

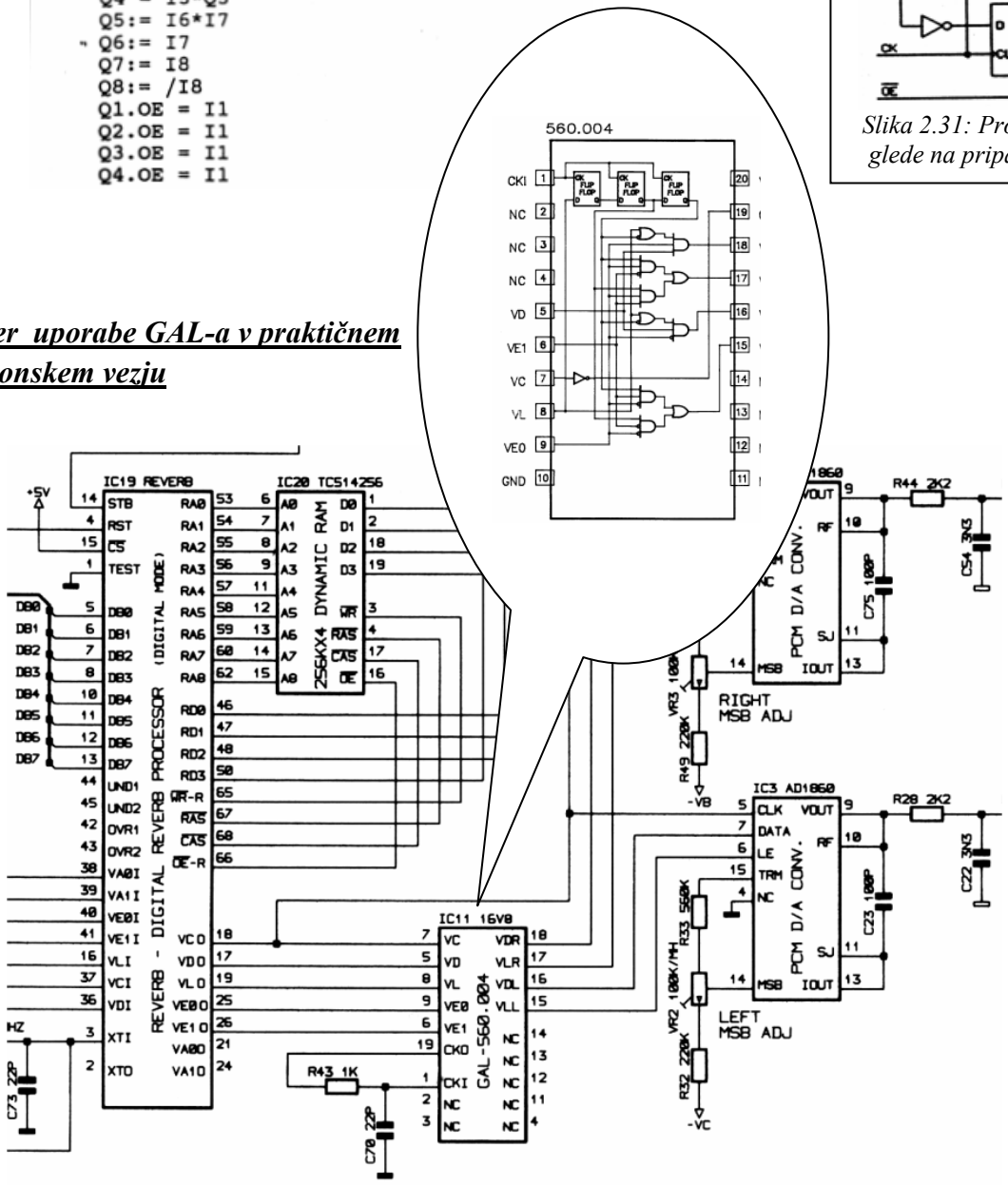
Pri registriranih izhodih se podatek vpiše v D-flip flop ob prehodu CK iz 0 v 1.Registrirane izhode omogočimo tako, da na vhod u OE logična 0. Priključek 1 (CK) je rezerviran za proženje kontroliranih izhodov, priključek 11 (OE) pa za omogočitev registriranih izhodov.

```
Registrirani izhodi      ;Header programa

CHIP reg GAL16V8         ;Deklaracijski blok

;pin   1   2   3   4   5   6   7   8   9
       CK  I1  I2  I3  I4  I5  I6  I7  I8  G

;pin  11  12  13  14  15  16  17  18  19
     /OE  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  V

ŽUES REG0194          ;user electronic signat

EQUATIONS             ;Funkcijski blok
     Q1 = /I2 * I3
     /Q2 = I3*/I4
     Q3 = I4 + I5
     Q4 = I5*Q5
     Q5:= I6*I7
     Q6:= I7
     Q7:= I8
     Q8:= /I8
     Q1.OE = I1
     Q2.OE = I1
     Q3.OE = I1
     Q4.OE = I1
```



Slika 2.31: Programirano vezje glede na pripadajoči program

<u>Primer uporabe GAL-a v praktičnem elektronskem vezju</u>



Primera programiranja GAL-a v okviru vaj!

1.Primer:

*Primerjalnik števila glede na **LIHO** in **SODO***

```
*IDENTIFICATION
    Kombi_6_vhodi;

*TYPE
    GAL16V8;

*PINS

    % binarni vhodi za številčni podatek%

    A  =  2,
    B  =  3,
    C  =  4,
    D  =  5,
    E  =  6,
    F  =  7,

    % Izhodi funkcij za sodo oz. liho število  %

    a  =  12,   % Sodo %
    b  =  13;   % Liho %


*BOOLEAN-EQUATIONS

    a = A & B & /C & /D & E & /F + A & B & /C & D & /E & /F + A & B & /C & D & E & /F + A & B & C & /D & /E & /F + A & B & C & /D & E & /F + A & B & C & D & /E & /F;

    b = A & B & /C & /D & E & F + A & B & /C & D & /E & F + A & B & /C & D & E & F + A & B & C & /D & /E & F + A & B & C & /D & E & F;

*END
```

2.Primer:

***Prekodirnik** iz BCD koda v 7-segmentni kod*

```
*IDENTIFICATION
    bin_to_7;

*TYPE
    GAL16V8;

*PINS

    % dvojiški vhodi%

    A  =  2,
    B  =  3,
    C  =  4,
    D  =  5,

    % Izhodi Sedem segmentnih LED-prikazovalnikov s skupno anodo  %

    /a  =  18,
    /b  =  17,
    /c  =  16,
    /d  =  15,
    /e  =  14,
    /f  =  13,
    /g  =  12;


*BOOLEAN-EQUATIONS

    a = /A & /C + B & /D + B & C + /A & D
      + /B & /C & D + A & C & /D;

    b = /A & /C + /C & /D + A & B & /D
      + A & /B & D + /A & /B & /D;

    /c = /A & B & /C & /D + B & C & D + /A & C & D;

    d = /B & D + /A & /C & /D + A & B & /C
      + A & /B & C + /A & B & C;

    e = /A & /C + B & D + C & D + /A & B;

    f = /A & /B + B & D + /C & D + /A & C + /B & C & /D;

    g = /A & B + B & /C + A & D + /C & D + /B & C & /D;

*END
```