Logički analizator

Logički analizator je mjerni instrument sličan digitalnom osciloskopu. Osnovna razlika je u njegovoj prilagođenosti promatranju signala u digitalnom sustavu. Obično nas prilikom analize rada digitalnog sustava zanima veći broj signala (osciloskopi obično nude 2-4 kanala), ali nas ne zanima točna amplituda signala. Kako digitalni signali poprimaju isključivo vrijednost "0" ili "1", odnosno "nisku" ili "visoku" razinu, rezolucija (broj različitih vrijednosti koje uređaj može izmjeriti, odnosno broj bitova kojima se te vrijednosti mogu prikazati) potrebna za prikaz pojedinog digitalnog signala je 1 bit (odnosno 2 vrijednosti). Stoga logički analizator ima veliki broj kanala (mjernih linija) – tipično 40, 80 ili više, pri čemu se izmjerena vrijednost prikazuje kao "0" ili "1" ovisno da li je manja ili veća od zadanog praga.

Osciloskop se koristi:

- kad se želi promatrati i najmanja promjena napona signala
- kad je potrebno precizno izmjeriti vremenski interval između dva događaja

Logički analizator se koristi:

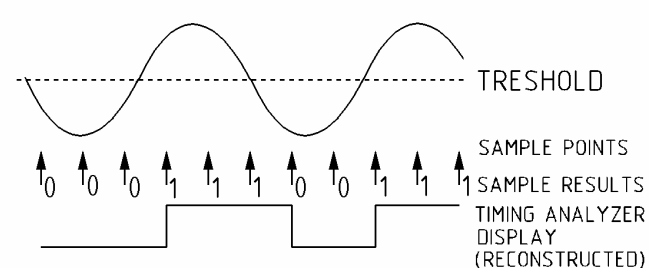
- kad je potrebno promatrati mnogo signala istovremeno
- kad je potrebno detektirati određeni uzorak jedinica i nula na nekoliko linija

Logički analizator omogućava različite prikaze izmjerenih podataka. Primjerice, moguće je definirati skupove signala koji se zajednički prikazuju u binarnom ili heksadekadnom sustavu, a moguće je pojedinim kombinacijama vrijednosti grupe signala dodijeliti i simboličke vrijednosti. Na taj se način može čak i obaviti disasembliranje koda koji izvršava mikroprocesor koji snimamo.

Obično logički analizator može raditi u dva različita načina rada: kao *vremenski analizator* ili kao *analizator stanja*.

Vremenski analizator

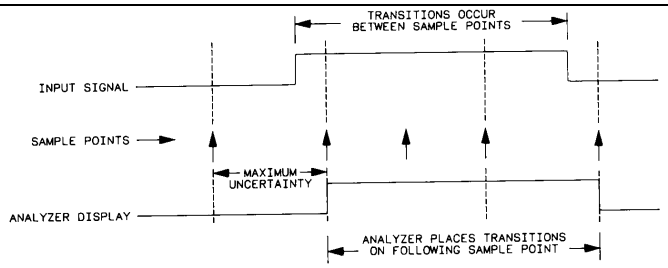
Vremenski analizator prikazuje podatke kao osciloskop (x-os predstavlja vrijeme, a y-os amplitudu signala). Ulazni se signal ispituje i na temelju unaprijed zadanog praga određuje je li visok ili nizak (ako je amplituda signala iznad praga, bit će prikazan kao "1", inače kao "0").



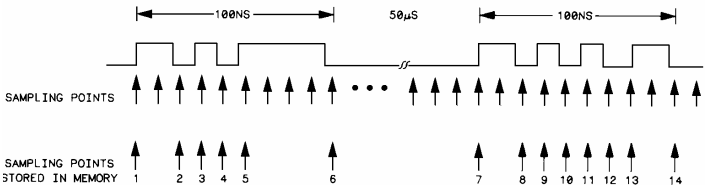
Formira se lista jedinica i nula koja predstavlja prikaz ulaznog signala (nema vrijednosti između 0 i 1). Lista se sprema u memoriju i koristi za rekonstrukciju ulaznog signala (slično osciloskopu, samo s jednim bitom okomite rezolucije). S jednim bitom rezolucije mogu se prikazati samo dva stanja- visoko ili nisko.

Ako je u trenutku ispitivanja linija u jednom od dva stanja ("0" ili "1"), a u slijedećem ispitnom trenutku u suprotnom stanju ("1" ili "0"), analizator zna da se dogodila promjena negdje između ispitnih točaka, ali ne zna točno gdje, pa promjenu prikazuje u drugoj ispitnoj točci - neodređenost vremenskog trenutka promjene.Najveća neodređenost iznosi jedan ispitni period (ako se promjena dogodila odmah iza prethodne ispitne točke).

Ako se uzima kraći ispitni period (više ispitnih točaka), smanjuje se vrijeme koje se može promatrati jer svaka ispitna točka koristi jednu memorijsku lokaciju. Npr, ako je period uzorkovanja 10ns, vremenski analizator s 1k memorije prestao bi prihvaćati podatke nakon 10,24ms = 10ns * 1024.



Tehnika kojom se rješava ovaj problem naziva se uzorkovanje promjena (*engl. transitional sampling*), gdje se spremaju samo točke kojima je prethodila promjena i vrijeme proteklo od zadnje promjene (koriste se dvije memorijske lokacije po promjeni i niti jedna ako nema promjene). Pri tome je "efektivna dubina memorije" jednaka ukupnom vremenu promatranja podijeljenom s periodom uzorkovanja.

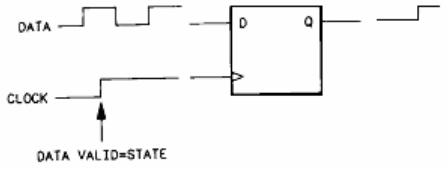


Osciloskop počinje snimati signal nakon okidne točke, dok logički analizator neprekidno hvata podatke, te može prikazati podatke prije okidne točke ("negativno vrijeme") i poslije.

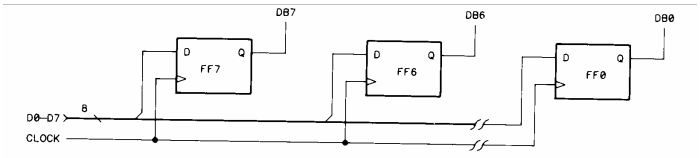
- okidanje na uzorak (uzorak je kombinacija jedinica i nula na ulaznim linijama)
- okidna točka može se postaviti u bin, hex, oct, dec ili ASCII obliku
- okidanje na brid: mnogi su logički sklopovi osjetljivi na brid, pa se mora omogućiti praćenje podataka kad se uređaj pokrene (npr. posmačni registar okidan bridom)

Analizator stanja

"Stanje" logičkog sklopa je uzorak na njegovim linijama kad su podaci stabilni (valjani).



Podatak na D-ulazu je valjan tek na uzlazni brid signala vremenskog vođenja, tj. stanje bistabila definirano je kad se pojavi pozitivan brid signala vremenskog vođenja. Ako se više takvih bistabila spoji na isti signal vremenskog vođenja (CLOCK), svi će prihvatiti podatke sa svojih D-ulaza na pozitivnu promjenu CLOCK-a. Analizator se može konfigurirati tako da snima podatke kad se pojavi pozitivan brid CLOCK-a (bilo koja aktivnost na ulaznim linijama neće se prikazati ako nema te promjene).



Analizator stanja ima sinkrono uzorkovanje jer koristi takt sustava koji se ispituje, dok vremenski analizator ima vlastiti unutarnji takt koji upravlja uzorkovanjem (asinkrono). Vremenski analizator prikazuje podatke vremenskim dijagramom, a analizator stanja daje ispis podataka u obliku liste.

Vježba 5: Sabirnica sustava MC68020 EVM

U ovoj vježbi promatraju se stanja na sabirnici modula baziranog na mikroprocesoru MC68020, za sabirnički ciklus dohvata podatka, te upisa podatka u memoriju. Sabirnica je asinkrona i prijenos se obavlja uz primjenu postupka rukovanja (*engl. handshaking*).

Procesor započinje sabirnički ciklus postavljanjem signala ECS*, a ukoliko je to prvi ciklus prijenosa operanda, postavlja se i signal OCS* (ovo je potrebno ako se operand prenosi u više sabirničkih ciklusa). Signal R/W* označava radi li se o ciklusu upisa ili čitanja podatka. Nakon postavljanja adrese na adresnu sabirnicu, signal AS* označava da je adresa na sabirnici stabilna. Signalima SIZ0/1 određuje se veličina podatka koji se prenosi.

U ciklusu pisanja procesor postavlja podatak na sabirnicu, a zatim signalom DS* označava njegovu valjanost. Sabirnica se drži stabilnom sve dok se ne dobije potvrda prijenosa podataka signalom DSACK0/1*. Kombinacija ovih signala označava koju veličinu podatka je periferija sposobna primiti, iz čega slijedi i broj ciklusa u kojima se obavlja prijenos podatka.

U ciklusu čitanja procesor također postavlja DS*, ali nešto kasnije, i nakon toga čeka odgovor periferije da je spremna dati podatak. I ta dojava se obavlja signalima DSACK0/1*, čija kombinacija označava veličinu podatka koju periferija daje.

DSACK1*	DSACK0*	rezultat
H	H	umeću se stanja čekanja
H	L	širina sab. podataka 8 bita
L	H	širina sab. podataka 16 bita
L	L	širina sab. podataka 32 bita

SIZ1	SIZ0	veličina
0	1	oktet
1	0	riječ
1	1	tri okteta
0	0	duga riječ

Rad u laboratoriju:

1) Konfigurirati logički analizator za vremensko snimanje stanja na sabirnici. Pri tome koristiti priloženi raspored linija priključenih na analizator (tablica). U tablici je označeno na koji priključak analizatora i na koju nožicu priključka su priključeni pojedini sabirnički signali.

Tablica I: Raspored signala na priključcima analizatora

	pin	signal	opis
POD 1	1	CLK	(Clock)
	2	R/W	(Read/Write)
	13	DS	(Data Strobe)
	4	AS	(Address Strobe)
	5	SIZ0	(Size)
	6	SIZ1	
	7	ECS	(External Cycle Start)
	8	OCS	(Operand Cycle Start)
	9	DBEN	(Data Buffer Enable)
	11	DSACK0	
	12	DSACK1	(Data Transfer and Size Acknowledge)
	0, 10, 14-15	NC	(Not Connected)
POD 2:		A0 - A15	(donji dio adresne sabirnice)
POD 3:		A16 - A31	(gornji dio adresne sabirnice)
POD 4:		D0 - D15	(donji dio sabirnice podataka)
POD 5:		D16 - D31	(gornji dio sabirnice podataka)

Prilikom snimanja stanja na sabirnici potrebno je promatrati sljedeće signale:

CLK signal vremenskog vođenja
ECS početak vanjskog sabirničkog ciklusa
OCS označava prvi sabirnički ciklus prijenosa operanda
AS označava da je na sabirnici valjana adresa
DS označava da procesor postavio valjan podatak na sabirnicu - pisanje, ili da se od vanjskog uređaja očekuje da postavi podatak na sabirnicu - čitanje
R/W visoko - ciklus čitanja, nisko ciklus pisanja
DBEN omogućava vanjski sabirnički međuspremnik
SIZ0/1
označava broj okteta koje još treba prenijeti u tekućem ciklusu
DSACK0/1
odgovor procesoru da je završena operacija prijenosa podataka
A0-A31
adresna sabirnica
D0-D31
sabirnica podataka

2) Korištenjem monitor programa MC68020 modula (kratak opis naredbi priložen) napisati kratak programski odsječak kojim se dohvaća podatak iz memorije. Neka se program vrti u petlji, a izlazak iz programa se obavlja resetiranjem EVM modula. Primjer takvog programa:

```
400600      move.l$400700.l,d2
            bra   $400600
```

(Na lokaciju s koje se čita prethodno upisati prepoznatljiv podatak.)

3) Postaviti okidnu sekvencu logičkog analizatora i snimiti stanja na sabirnici u toku izvođenja petlje. Analizirati dobivene valne oblike, te prepoznati pojedine faze izvođenja instrukcija. Skicirati stanja na sabirnici za jedan tipični sabirnički ciklus dohvata podatka.

4) Analogno koraku 2) napisati kratku programsku petlju u kojoj se vrši upis podatka u memoriju.

```
400600      move.l#$12345678,$400700.l
            bra   $400600
```

5) Snimiti stanja na sabirnici i analizirati dobivene valne oblike. Potrebno je paziti na sabirnički ciklus upisa podatka u memoriju i skicirati stanja na sabirnici u tom ciklusu.

6) Ponoviti korake 2) do 5) (čitanje i upis) za slijedeće slučajeve prijenosa podataka:

   a) prijenos podatka duljine 8 bita (oktet)
   b) prijenos podatka duljine 16 bita (riječ)
   c) prijenos podatka duljine 32 bita (duga riječ) koristeći neparnu adresu kao izvor odnosno odredište operanda.

Dodatak: Pregled osnovnih naredbi monitor programa MC68020

An ispis sadržaja adresnog registra An, i eventualna izmjena Novi sadržaj se unosi u heksa kodu, nakon čega treba pritisnuti tipku <ENTER>. Pritisak na <ENTER> bez prethodnog unosa ostavlja stari sadržaj u registru.

AE ASCII entry, CTRL-<term> to exit, CTRL-N=00 upis ASCII stringa u memoriju. Nakon unosa početne adrese, zahtijeva se unos niza znakova.

<BACKSPACE> omogućava brisanje znakova unazad.
<CTRL C> sprema $8D (CR sa postavljenim bitom 7).
<CTRL L> sprema $8A (LF sa postavljenim bitom 7)
<CTRL N> sprema $0 ('\0')
<CTRL T> završetak unosa (može se podesiti naredbom ST)

AS line-by-line assembler

Assembler prihvaća standardne Motoroline mnemonike, i 6 pseudo-naredbi. Pretpostavljena početna adresa je $400600, a to je ujedno i najniža iskoristiva adresa u RAM-u. Izmjena početne adrese moguća je korištenjem pseudo-naredbe "$nnnnnn". Komentari se prihvaćaju, uz uvjet da postoji bar jedna praznina iza naredbe. Linija koja počinje sa "*" također se shvaća kao komentar.

Pseudo-naredbe:

$nnnnnn unosi se nova adresa za smještanje koda

ASCII unos niza znakova - nakon naredbe slijedi razmak, a zatim niz znakova, zaključen s <ENTER>

ASCIZ unesenom nizu znakova dodaje se zaključni nul-karakter ("\0")

DATA unos okteta
pr. DATA 0,1,$10,32
- podaci se poravnavaju na granicu riječi !

END završetak unosa asemblerskog koda

*Napomena: ne postoji mogućnost korištenja labela - odredište grananja se mora odrediti kao apsolutna adresa, ili kao pomak u odnosu na programsko brojilo (BRA *+28)*

Bn - breakpoint display/set, n = 1 to 4

Moguće je koristiti do 4 prekidne točke. Unos nove adrese postavlja novu vrijednost prekidne točke. Unos adrese 0 uklanja prekidnu točku. Pritisak na <ENTER> ostavlja prekidnu točku na starom mjestu.

CM - compare two blocks of memory

Usporedba dvaju blokova memorije. Zadaje se početna i zvršna adresa prvog bloka, te početna adresa drugog bloka memorije. Ispisuju se lokacije na kojima je ustanovljena razlika. Ako nema tog ispisa, nije ustanovljena ni jedna razlika.

CN - continue execution after break

Uklanja se prekidna točka na kojoj je zaustavljeno izvršavanje programa, postavlja se prava instrukcija i od nje započinje izvršavanje.

CV - convert between hex and decimal

Pretvara decimalni broj (do 8 znamenaka) u heksadecimalni, ili heksadecimalni (započinje sa 'X') u decimalni. Najveći broj koji se može pretvoriti je 99999999, odnosno 5F5E0FE heksadecimalno.

Dn - Ispis sadržaja registra podataka Dn, i eventualna izmjena Novi sadržaj se unosi u heksa kodu, nakon čega

treba pritisnuti tipku <ENTER>. Pritisak na <ENTER> bez prethodnog unosa ostavlja stari sadržaj u registru.

DS - disassembler
Zadaje se početna i završna adresa bloka koji treba disasemblirati

FI - fill block of memory with word
Zadaje se početna i završna adresa bloka memorije i riječ koja se treba upisati.

GO - go and execute target program
Pokreće se izvođenje programa počevši od zadane adrese. Izvođenje se zaustavlja nailaskom na prekidnu točku, ili pritiskom na tipku RESET na modulu MC68020.

HE - help!

IR - Inicializacija registara D0-7 & A0-6.
U sve registre upisuje se zadana vrijednost. Pritisak na tipku <ENTER> upisuje u sve registre vrijednost 0.

LW - locate data word in memory block
Pretražuje se specificirani blok memorije.

MO - memory open/modify.
Omogućava pregledavanje niza memorijskih lokacija i izmjenjivanje njihovog sadržaja. Format dohvata je riječ (16-bita). Unose se 4 heksa znamenke.
Upravljanje:
<space> spremanje unesene vrijednosti (odnosno stare ukoliko nova nije unesena), i otvaranje nove lokacije.
<strelica prema gore>
nova upisana riječ se pohranjuje na prethodnu otvorenu lokaciju (otvara se prethodna lokacija.
<ENTER> upisana riječ se pohranjuje, a zatim se izlazi iz upisa.

MV - move memory block
Premješta se specificirani blok na zadanu adresu.

RD - prikaz sadržaja registara

SR - pregled ili upis novog sadržaja statusnog registra

SS - single-step
Izvođenje programa instrukciju po instrukciju, počevši od zadane adrese. Pritisak na <ENTER> nastavlja izvođenje.

TR - trace target program
Izvođenje programa instrukciju po instrukciju, ali bez zaustavljanja nakon svake instrukcije (stanje se ispisuje).
<CTRL S> zaustavlja ispis
<DEL> prekida izvođenje
<bilo koja tipka> nastavak

Vježba 6: Sabirnica s vremenski multipleksiranim linijama

U ovoj vježbi promatraju se stanja na sabirnici mikroprocesora Intel 8088. Taj procesor ima neke linije vremenski multipleksirane. Promatrat će se multipleksiranje linija podataka s adresnim linijama.

Rad u laboratoriju:

1) Konfigurirati logički analizator za snimanje stanja na sabirnici. Pri tome koristiti priloženi raspored linija priključenih na analizator (Tablica I). U tablici je označeno na koji priključak analizatora i na koju nožicu priključka su priključeni pojedini sabirnički signali.

Tablica I: Raspored signala na priključcima analizatora

	nožica	signal	opis
POD 1	1-8	A/D0-7	multipleksirane linije podataka i adresa
	9	MEMW*	pisanje memoriju
	10	ALE	omogućavanje zapornog sklopa adrese (address latch enable)
	11	RD*	čitanje
	12	CLK	impuls ritma

Prilikom snimanja stanja na sabirnici potrebno je promatrati sljedeće signale;

- CLK signal vremenskog vođenja
- RD* ciklus čitanja
- MEMW* ciklus pisanja u memoriju
- ALE upis adrese u zaporni sklop (prijelaz H-L)
- multipleksirane adresne i podatkovne linije 0 - 7

Napomena: mikroprocesor I8088 ima 20 adresnih linija, tj. može adresirati 1 M memorijskih lokacija. U ovoj vježbi će se promatrati samo linije A0 - A7, koje su vremenski multipleksirane s linijama podataka D0 - D7.

Signali ALE i MEMW* nisu izvedeni na nožicama mikroprocesora, nego se generiraju u dodatnom sklopovlju (sabirnički upravljački sklop I8288), na osnovi označnih linija procesora.

2) Uz pomoć programa **debug** potrebno je izazvati prijenos poznatog podatka na poznatu adresu, i snimiti stanja na sabirnici.

Uputa:

- U program se ulazi iz DOS-a naredbom: **>debug**
- Memorija je organizirana u segmentima. Radi jednostavnijeg preračunavanja adresa prikladno je koristiti okrugle bazne adrese, npr. adrese u području od 2000:0000 (fizička adresa 20000), jer se fizička adresa dobiva jednostavnim pribrajanjem baze i pomaka, uz posmicanje baze za jedno hex mjesto u lijevo). Zbog snimanja samo donjeg dijela adresne sabirnice, stanje na sabirnici odgovara nižem oktetu pomaka. Da bi se ostvarili dohvati memorije u tom području, potrebno je postaviti registre DS (Data Segment) i CS (Code Segment) na odgovarajuću vrijednost.
- Izmjena sadržaja registra obavlja se naredbom **r**:
  `rcs` ispisuje sadržaj registra cs, a zatim prihvaća novu vrijednost (samo <ENTER> ostavlja staru vrijednost u registru)
  `rds` prikaz i izmjena sadržaja ds registra
- Prikaz dijela memorijskog sadržaja se obavlja naredbom **d**:
  `d2000:10 20` ispisuje sadržaj memorije od adrese 2000:0010 do 2000:0020
- Upis sadržaja u niz memorijskih lokacija obavlja se naredbom **f**:
  `f2000:10 1 8 12 34 56 78`
  upisuje niz okteta 12 34 56 78 na 8 lokacija (niz se ponavlja), počevši od adrese 2000:0010 (l je malo slovo "L")

- Program u assembleru može se upisati korištenjem naredbe **a**:
  `a2000:0` započinje unošenje mnemonika. Nakon pritiska na <ENTER>, prihvaća se mnemonik, ili se prijavljuje greška. Samo <ENTER> izaziva povratak u komandnu liniju debug-a.
- Izvršavanje strojnog programa započinje naredbom **g**:
  `g=2000:0` pokreće program od adrese 2000:0000
- Mnemonici I8088:

`mov ax,20`	upis broja 20 hex u registar a
`mov ax,[20]`	upis u ax sa adrese 20 (u Code Segmentu)
`mov [20],ax`	upis na adresu 20 iz ax
`jmp 20`	skok na adresu 20 tekućeg segmenta

Prijedlog za obavljanje vježbe:

Postaviti cs i ds na vrijednost 3000. Upisati program:

```
3000:0000    mov ax,[20]
             jmp 0
```

Na lokaciju 3000:20 postaviti prepoznatljiv broj (npr. 12 34). Prikazati sadržaj memorijskih lokacija na kojima se nalazi program. Pokrenuti izvođenje programa. Postaviti okidnu riječ analizatora (npr. A/D0-7 = kod prve instrukcije u petlji) i snimiti stanja na sabirnici. Uočiti dohvat instrukcija, te dohvat memorije.

Skicirati stanja na sabirnici za tipičan ciklus dohvata riječi iz memorije. Obratiti pažnju na odnos svih važnih signala i signala vremenskog vođenja. Ciklus signala vrem. vođenja započinje silaznim bridom. Označiti periode takta (T1, T2,...) za sabirnički ciklus dohvata podatka. Prvi period takta je onaj u kojem je aktivan signal ALE.

Nakon obavljenog snimanja stanja ponovno uputiti računalo (CTRL-ALT-DEL). Ponovno ući u DEBUG. Postaviti cs i ds na 3000. Upisati program:

```
3000:0000    mov ax,1234
             mov [20],ax
             jmp 0
```

Pokrenuti program i snimiti stanja na sabirnici kao u prethodnom slučaju. Uočiti dohvate instrukcija, te upis u memoriju. Skicirati i označiti sabirnički ciklus upisa u memoriju.

Vježbe 7 i 8: Programibilni logički sklopovi - Dekoder skanirajuće tastature

GAL sklopovi

GAL (engl. Generic Array Logic) je programabilni logički sklop (PLD) iz skupine PLA izveden u E^2CMOS tehnologiji. To je sklop velike brzine i male potrošnje koji omogućava definiranje željene funkcije svakog pojedinog izlaza. Svaka izlazna logička makro-ćelija, pored standardne I-ILI funkcionalnosti, svojstvene svim PLA, sadrži i bistabil te sklopove s tri stanja koji omogućuju dvosmjernu komunikaciju preko pripadnih izvoda čipa. Proizvođač garantira 100 ciklusa pisanja/brisanja i postojanost podataka upisanih u GAL od 20 godina.

GAL-ovi su podržani sa standardnim programskim alatima razvijenim za PLD-ove. Logička funkcija sklopa kojeg želimo dizajnirati može se zadati pomoću Booleovih logičkih jednadžbi, tablice stanja, pomoću grafičkih primitiva (shematski prikaz)... Za transformaciju tog zapisa u oblik pogodan za GAL-programator koristi se neki od prevodilaca (CUPL, ABEL, DASH, ORCAD, ...) koji će generirati mapu spojeva (engl. fusemap) na osnovu koje se programira GAL. Jedno od vrlo korisnih svojstava koje pružaju neki programski alati za PLD sklopove je mogućnost simulacije rada sklopa na osnovi implementiranih logičkih jednadžbi.

Program CUPL

Program CUPL jedan je od programa iz skupine tzv. PAL asemblera, programa za prevođenje ulaznih tekstualnih opisa funkcija sklopa i stvaranje datoteka koje se koiste za direktno programiranje raznih vrsta programibilnih logičkih sklopova. Program podržava programibilna logička polja (PLA) raznih proizvođača, razne tipove PROM i EPROM čipova i neke vrste FPGA. Jednadžbe i podaci upisuju se korištenjem logičkih operatora, definiranjem korisničkih funkcija, sekvenci, te tablica stanja i podataka. Moguće je i defniranje simboličkih imena, konstanti u binarnom, heksadekadskom i dekadskom formatu. Signali se mogu grupirati u polja i u tom obliku uključivati u jednadžbe. Kao izlaz moguće je generirati datoteku u JEDEC izlaznom formatu koji je standard za uređaje za programiranje PLA.

Sintaksa ulaznih datoteka

Datoteka jednadžbi

- sufiks imena datoteke mora biti.PLD
- zaglavlje unutar datoteke ima oblik:

```
Name        jednostavno dekodiranje;
Partno      0001;
Revision    01;
Date        20/12/02;
Designer    ZEMRIS;
Company     FER;
Location    Zagreb;
Assembly    Primjeri;
```

- komentari se pišu kao u programskom jeziku C (/*...*/)
- pojedine varijable odnosno signali koji se koriste u jednadžbama pridružuju se nožicama čipa naredbom Pin, s tim da se signali mogu grupirati u proizvoljna polja:

```
Pin 2 = IO ;              /* ulazni signal */
Pin 3 = MEM;
Pin 4 = !WR ;             /* invertirani ulazni signal */
Pin[5..6] = [ADR1..0];    /* polje varijabli pridruzeno grupi */
                          /* ulaznih signala */
Pin[16..19] = [SEL3..0];  /* polje izlaznih signala */
                          /* signali su spojeni na izlaze cipa */
```

- Grupe signala mogu se udružiti u polja koja se mogu direktno uključiti u logičke jednadžbe:

```
field adrese = [ADR1..0];    /* grupiranje signala u polje */
```

- kao logički operatori koriste se znakovi: ! (negacija), & (logicko 1), # (ILI), $ (isključivo ILI)
- u izraze se mogu uključiti i grupirani signali

```
SELO = !IO & MEM & !WR & adrese:'b'00 ;  /* binarno zadan podatak */
SEL1 = !IO & MEM & !WR & adrese:'b'Ol ;
SEL2 = !IO & MEM & !WR & adrese:2 ;      /* dekadski zadan podatak */
SEL3 = !10 & MEM & !WR & adrese:3 ;
```

Datoteka test vektora

- ima sufiks `.SI`
- *zaglavlje mora biti isto kao i u datoteci jednadžbi !*
- Nakon zaglavlja defnira se redoslijed signala koji se simuliraju. Razmaci između signala dodaju se naredbom `%n`, gdje *n* označava broj umetnutih razmaka.

```
ORDER: MEM, %3, I0, %2, WR, %2,adrese, %2, SEL3..0 ;
```

- Nakon što je defniran redoslijed signala unose se test vektori:

```
VECTORS:
$msg"MEM IO WR ADR SEL" ;
1 0 0 00 LLLH
1 0 0 01 LLHL
1 0 0 10 LHLL
1 0 0 11 HLLL
X 1 X XX LLLL
```

- za izlazne signale (predviđene rezultate) koriste se oznake L, H i Z. Naredba $msg"..." uzrokuje ispis zadane poruke koja može poslužiti za lakše očitanje rezultata simulacije

Pokretanje programa CUPL

Program CUPL koristi se iz komandne linije i ima slijedeću sintaksu:

```
> cupl [-flags] [library] [device] source
```

Kratki opis sintakse i ugradenih programskih sklopki dobije se zadavanjem naredbe cupl bez argumenata. Osnovna biblioteka koja sadrži sve komponente (PAL, GAL, ROM,...)je cupl.dl.

Primjeri zadavanja naredbi programa CUPL:

```
> cupl -sxf G16V8 dekoder
```

Iz ulaznih datoteka `dekoder.pld` i `dekoder.si` nakon prevođenja i simulacije (programska sklopka `-s`) dobiju se: izlazna datoteka `dekoder.so` koja sadrži rezultate simulacije za čip GAL 16V8, te izlazna datoteka `dekoder.doc`, koja sadrži podatke o korištenim jednadžbama, njihovom razvijenom obliku u produkt termi (sklopka `-x`), raporedu spaljivanja osigurača kao i sliku čipa s oznakama korištenih signala (sklopka `-f`).

```
> cupl -j G16V8 dekoder
```

Programska sklopka `-j` uzrokuje stvaranje izlazne JEDEC datoteke `dekoder.jed` koja se koristi za programiranje čipa GAL 16V8.

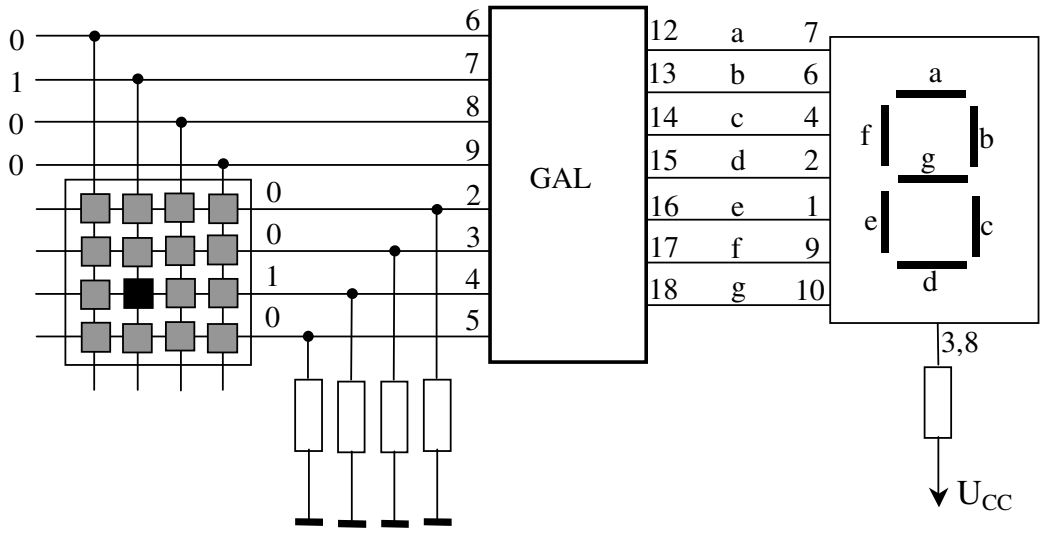
Rad u laboratoriju

Zadatak 1

Na slici 6-1 prikazan je sustav koji dekodira tipku pritisnutu na tipkovnici i prikazuje odgovarajući znak na 7-segmentnoj prikaznoj jedinici. Potrebno je oblikovati sklop (na slici označen kao "GAL") koji će na temelju kombinacije na ulazima upaliti odgovarajuće segmente prikazne jedinice. Sklop se oblikuje definiranjem logičkih jednadžbi koje oblikuju izlazne funkcije u ovisnosti o stanju na ulazima.

Tipkovnica je pasivna (matrica u kojoj se pritiskom na tipku uspostavlja električka veze između odgovarajućeg retka i stupca), pa se utvrđivanje njenog stanja obavlja postupkom "šetajuće jedinice". Na ulaze (npr. stupce) dovodi se kombinacija koja sadrži točno jednu jedinicu (visoku logičku razinu), dok su ostale ulazne linije u niskoj razini. Linije koje odgovaraju recima tipkovnice pritežu se prema masi (*pull-down* otpornicima). Kada nijedna tipka na tastaturi nije pritisnuta, izlazna kombinacija sadrži sve nule. Pritiskanje tipke u selektiranom stupcu (onom na koju je dovedena jedinica) rezultira prosljeđivanjem jedinice na odgovarajuću liniju retka i iz kombinacije na

linijama redaka i stupaca može se ustanoviti koja je tipka pritisnuta. Jasno je da se pritisak tipke u stupcima na kojima nije jedinica ne može ustanoviti. Stoga je nužno ulaznu kombinaciju mijenjati, "šetati" jedinicu i na taj način naizmjenično aktivirati stupac po stupac. To se može obaviti npr. prstenastim brojilom u koje se početno upiše kombinacija s jednom jedinicom.



Slika 6-1. GAL kao dekoder skanirajuće tipkovnice.

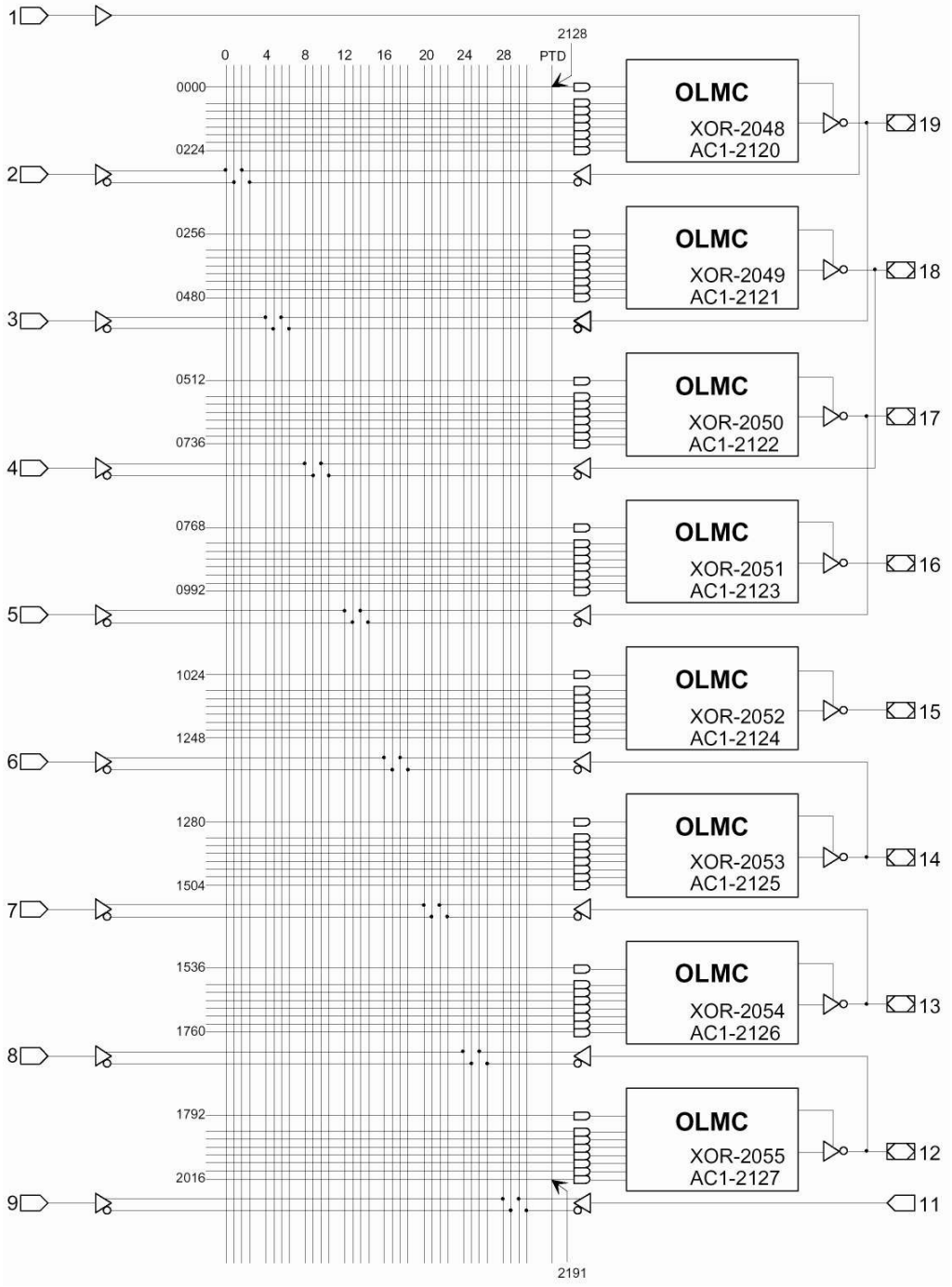
U ovoj vježbi potrebno je oblikovati sklop koji će dekodirati stanja tipkovnice prema gore opisanom postupku, te za svaku pritisnutu tipku upaliti odgovarajuću lampicu 7-segmentne prikazne jedinice. Na vježbama se rabe 7-segmentne prikazne jedinice sa zajedničkom anodom, pa se paljenje pojedinog segmenta postiže postavljanjem niske razine na odgovarajući ulaz prikazne jedinice. Spajanje anode prema napajanju obavlja se preko otpornika koji ograničava struju i štiti svijetleće diode od pregaranja.

Sklop opisati logičkim jednadžbama u obliku pogodnom za PLA sklopove. Na raspolaganju su sklopovi koji omogućuju realizaciju sume 8 produkata, pri čemu svaki produktni član može sadržavati do 16 članova (to označava i oznaka sklopa GAL 16V8). Jednadžbe zapisati u tekstualnu datoteku prilagođenu sintaksi programa CUPL. Struktura i raspored izvoda čipa GAL 16V8 prikazan je na slici 6-2. Napisati odgovarajuće ulazne test-vektore i predviđene izlazne signale za skanirajuću tastaturu i sedmosegmentnu prikaznu jedinicu.

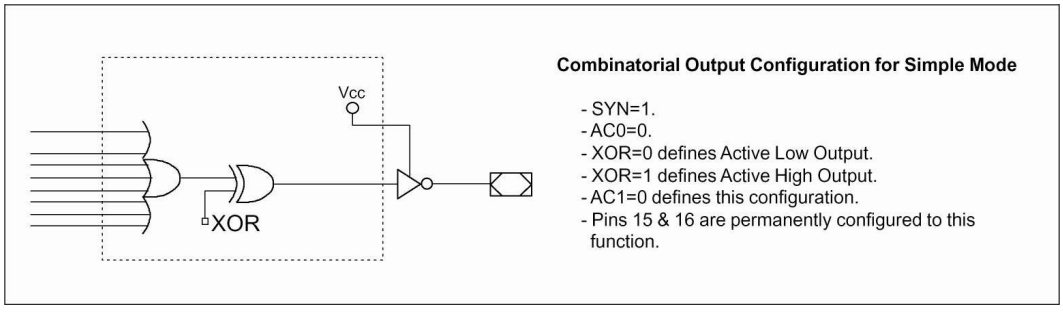
Ispravnost jednadžbi provjeriti simulacijom. Iz testiranih jednadžbi generirati JEDEC datoteku koja služi za programiranje GAL-a odgovarajućim programatorom.

Zadatak 2

Uz pomoć asistenta isprogramirati GAL koristeći JEDEC datoteku dobivenu u prethodnom zadatku. Prema shemi (slika 6-1) sa zadanim rasporedom izvoda čipa spojiti na spojnom polju 16-znamenkastu tastaturu, GAL (kao dekoder) i 7-segmentnu prikaznu jedinicu (LED) sa zajedničkom anodom. Provjeriti ispravnost rada sklopa.



Slika 6-2. Struktura programirljivog sklopa GAL 16V8.



Slika 6-3. Primjer konfiguracije izlaznog logičkog makrobloka (OLMC – *Output Logic Macrocell)*. O konfiguraciji se brine razvojna okolina (u našem slučaju CUPL)