

URS - Četvrti rok 2017.

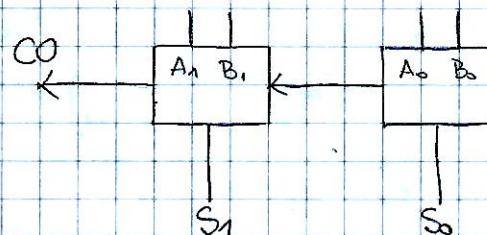
• PALASM

Zadatak 1. (MI 2016./2017.)

U sintaksi PALASM alata napisati izraze koji opisuju potpuno zbrojalo za dva dvobitna ulazna argumenta $[A_1, A_0] + [B_1, B_0]$ u sumu $[S_1, S_0]$ i prijenos CO.

Zbog čega nije moguće realizirati potpuno zbrojalo velike širine (npr. 6 bita) na sklopu GAL16V8?

* potpuno zbrojalo = 2 poluzbrojala



TABLICA:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 i prijenos 1

--- PIN DECLARATION ---

PIN	2,3	$A[1..0]$	COMBINATORIAL ; INPUT
PIN	4,5	$B[1..0]$	COMBINATORIAL; INPUT
PIN	12,13	$S[1..0]$	COMBINATORIAL; OUTPUT
PIN	14	CO	COMBINATORIAL; OUTPUT
PIN	10	GND	;
PIN	20	VCC	;

1. Zbrojajuje bitova niže tečine

$A_0 \ B_0$	S_0	C_0
00	0	0
01	1	0
10	1	0
11	0	1

$\Rightarrow S_0$ dobije se funkcijom XOR

$$\Rightarrow C_0 = A_0 * B_0$$

ESAN

2. Zbrojajuje bitova više tečine + carry

$A_1 \ B_1 \ C_0$	S_1	C_1
000	0	0
001	1	m_0
010	1	m_1
011	0	m_2
100	1	m_3
101	0	m_4
110	0	m_5
111	1	m_6

$$\Rightarrow S_1 = A_1 \text{ XOR } B_1 \text{ XOR } C_0$$

$$\Rightarrow C_1 = m_3 + m_5 + m_6 + m_7$$

$$= /A_1 * B_1 * C_0 + A_1 * /B_1 * C_0$$

$$+ A_1 * B_1 * /C_0 + A_1 * B_1 * C_0$$

$$= /C_0 * A_1 * B_1$$

$$+ C_0 * (A_1 + B_1)$$

* C_1 se može ili direktno pogledati tablicu ili računski

$$\begin{aligned}
 & /C_0 * A_1 * B_1 + C_0 * [/A_1 * B_1 + A_1 * /B_1 + A_1 * B_1] \\
 & = /C_0 * A_1 * B_1 + C_0 * [B_1 * (A_1 + A_1) + A_1 * B_1] \\
 & = /C_0 * A_1 * B_1 + C_0 * [(B_1 + A_1) * (B_1 + /B_1)] \\
 & = /C_0 * A_1 * B_1 + C_0 * (B_1 + A_1)
 \end{aligned}$$

--- EQUATIONS ---

$$S[0] = A[0] :+ : B[0]$$

$$S[1] = A[1] :+ : B[1] :+ : (A[0] * B[0])$$

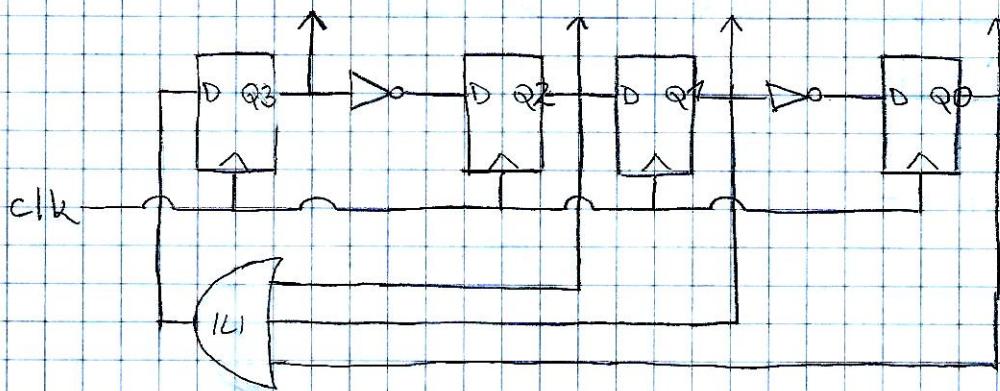
$$C0 = / (A[0] * B[0]) * A[1] * B[1]$$

$$+ (A[0] * B[0]) * (A[1] + B[1])$$

- Na sklopu GAL16V8 ne može se realizirati potpuno zbrojalo velike širine zato što sklop nema dovoljno ulaza, a ni sklop u "lui matrici" nema dovoljan broj produktnih članova.

Zadatak 2. (MI PRIMER)

Napisati deklaracije signala $Q[3..0]$ te odgovarajući EQUATION blok. Ako je početno stanje registra $Q[3..0] = \#b\ 1001$ (ne ugraditi u kod) izračunati kroz koja stanja prolazi sklop.



* D bistabil = flip flop on uzima ono što je na ulazu i prenosi to na izlaz

--- PIN DECLARATION ---

PIN	2..5:13;10:Q[3..0]	REGISTERED ; OUTPUT
PIN	15;8 :CLK	COMBINATORIAL ; INPUT
PIN	11	/OE ;
PIN	10	GND ;
PIN	20	VCC ;

--- EQUATIONS --

$$Q[0] = /Q[1]$$

$$Q[1] = Q[2]$$

$$Q[2] = /Q[3]$$

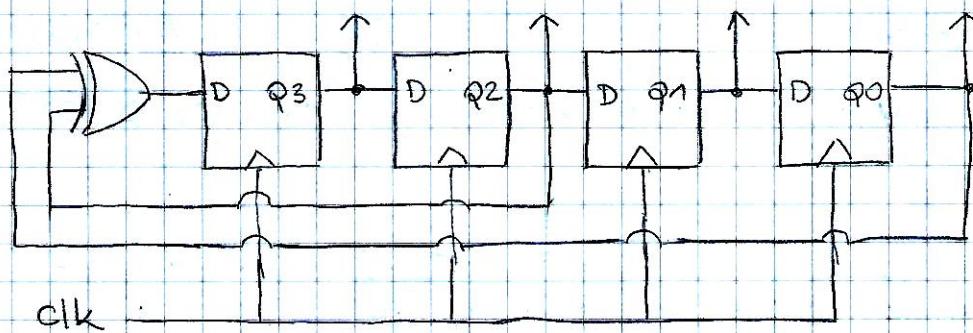
$$Q[3] = Q[2] + Q[1] + Q[0]$$

* ako je početno stanje 1001 onda su sledeća:

$$1001 \rightarrow 1001 \rightarrow 1001 \rightarrow \dots$$

Zadatak 3.

Sljčan problem - 4 bitni posmračni register



--- PIN DECLARATION ---

PIN	1	CLK	COMBINATORIAL ; INPUT
PIN	10	GND	
PIN	20	VCC	
PIN	11	/OE	
PIN	2..5	Q[3..0]	REGISTERED ; OUTPUT

--- EQUATIONS ---

$$Q[0] = Q[1]$$

$$Q[1] = Q[2]$$

$$Q[2] = Q[3]$$

$$Q[3] = Q[2] \oplus Q[0]$$

NAPOMENA:

\oplus = XOR

- Ako je početno stanje 1001 onda su sljedeća:

$$1001 \rightarrow 1100 \rightarrow 1110 \rightarrow 1111 \rightarrow 0111 \rightarrow 0011$$

$$\rightarrow 1001 \rightarrow \text{ i opet sve u knjig}$$

Zadatak 4. (MI 2016./2017.)

Implementirajte state machine za 2 bitno brojce sposobno

brojati i odbravljavati (dakle stanja su brojevi: 0-00, 1-01, 2-10, 3-11; brojci može ići gore / dolje ovisno o signalu UP ili DOWN). Stanja se trebaju ispisivati na 3 diode, tj. 3 izlaza ($0 \rightarrow$ sve ugašene, $1-1.$ upaljena, itd.). Dodatno postoji i signal INV, koji kada je aktivan inverte logiku osvjetljavanja dioda (sve upaljene $\rightarrow 0$, 1 ugašena $\rightarrow 1$, itd.).

SLJEDEĆE stanje UVJEK ovise o ulazu !!!

*

MOORE - izlazi su funkcija samo trenutnog stanja

- izlazi su trenutni

MEALY - izlazi su funkcija trenutnog stanja i signala ulaza

- izlazi kasne

* Radimo samo sinkrone

1) Odrediti ulazne i izlazne linije:

UP - ulaz

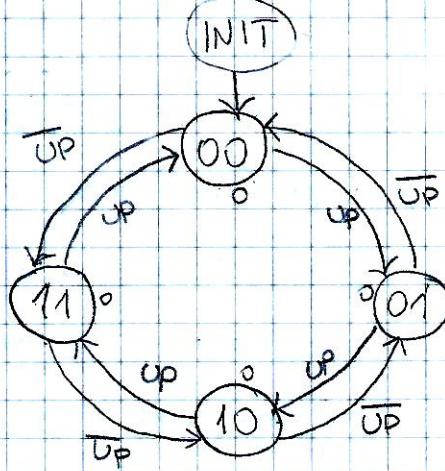
LD [2..0] - izlazi

INV - ulaz

2) Odrediti tip automata - izlazi ovise o ulazima \Rightarrow MEALY

--- PIN DECLARATION ---

PIN	1	CLK	COMBINATORIAL ; INPUT
PIN	2	UP_IN	COMBINATORIAL ; INPUT
PIN	3	INV_IN	COMBINATORIAL ; INPUT
PIN	12..14	LD[2..0]	REGISTERED ; OUTPUT
PIN	10	GND	;
PIN	20	VCC	;
PIN	11	IOE	;
PIN	15	UP	REGISTERED ; OUTPUT
PIN	16	INV	REGISTERED ; OUTPUT



--- EQUATIONS ---

STATE

MEALY-MACHINE

$$INIT = /LD[0] * /LD[1] * /LD[2]$$

$$ST0 = /LD[0] * /LD[1] * /LD[2]$$

$$ST1 = LD[0] * /LD[1] * /LD[2]$$

$$ST2 = LD[0] * LD[1] * /LD[2]$$

$$ST3 = LD[0] * LD[1] * LD[2]$$

Napomena:

PAL'CE 16V8 ima 8
bistabilna

ISPRAVITI !

DEFAULT BRANCH INIT

INIT := VCC \rightarrow ST0

ST0 := /UP \rightarrow ST3

+ \rightarrow ST1

ST1 := /UP \rightarrow ST0

+ \rightarrow ST2

ST2 := /UP \rightarrow ST1

+ \rightarrow ST3

ST3 := /UP \rightarrow ST2

+ \rightarrow ST0

$$\text{INIT_OUTF} := \text{INV} \rightarrow /LD[0] * /LD[1] * /LD[2]$$

$$+ \rightarrow /LD[0] * /LD[1] * LD[0]$$

$$\text{ST0_OUTF} := \text{INV} \rightarrow LD[0] * LD[1] * LD[2]$$

$$+ \rightarrow LD[0] * LD[1] * /LD[2]$$

$$\text{ST1_OUTF} := \text{INV} \rightarrow /LD[0] * LD[1] * LD[2]$$

$$+ \rightarrow LD[0] * LD[1] * /LD[2]$$

$$\text{ST2_OUTF} := \text{INV} \rightarrow /LD[0] * /LD[1] * LD[2]$$

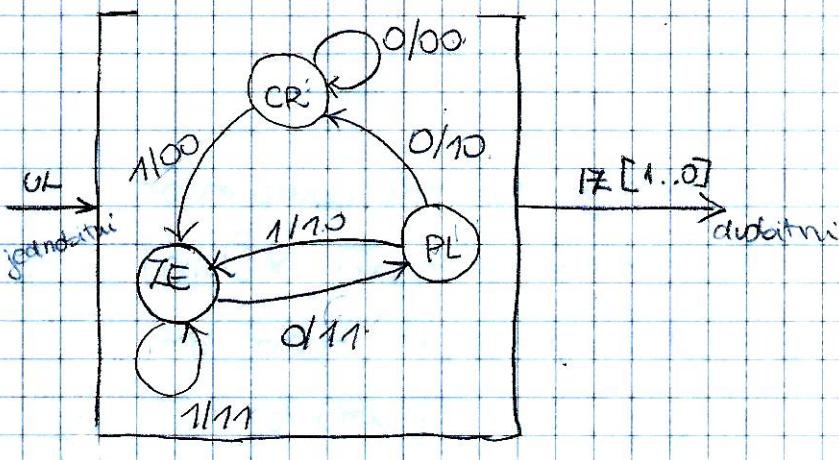
$$+ \rightarrow LD[0] * LD[1] * /LD[2]$$

$$\text{ST3_OUTF} := \text{INV} \rightarrow /LD[0] * /LD[1] * /LD[2]$$

$$+ \rightarrow LD[0] * LD[1] * LD[2]$$

Zadatok 5.

Zadan je automat s 1 ulazom i 2 izlaza.



O kojem se automatu radi?

Raspisemo tablicu:

TRENUTNO STANJE	ULAZ	SLEDEĆE STANJE	IZLAZ
CR	0	CR	00
CR	1	ZE	00
ZE	0	PL	11
ZE	1	ZE	11
PL	0	CR	10
PL	1	ZE	10

MOORE!

← - FAULT

$$CR = /IZ[1] * /IZ[0] ; 00$$

$$PL = IZ[1] * /IZ[0] ; 10$$

$$ZE = IZ[1] * IZ[0] ; 11$$

$$INIT = /IZ[1] * /IZ[0]$$

STATE

MOORE-MACHINE

*treba bi negativne
biti definirane

DEFAULT BRANCH INIT

$$INIT := VCC \rightarrow CR$$

$$CR := /UL \rightarrow CR$$

$$+ \rightarrow ZE$$

$$PL := /UL \rightarrow CR$$

$$+ \rightarrow ZE$$

$$ZE := /UL \rightarrow PL$$

$$+ \rightarrow ZE$$

Zadatak 6. (2008./2009.)

Napiši izraze u sintaksi PALASM-a koji opisuju dvočitni komparator brojeva u dvojnom komplementu. Ulazni vektori neka su $X[1..0]$ i $Y[1..0]$, a izlaz neka je IZ . Za slučaj $X > Y$, IZ treba biti logička nula, a inače jedinica. Napišite samo EQUATIONS blok naredbi.

* Kod zapisa 2k najviši bit pokazuje predznak:

0 - pozitivan

1 - negativan

X_1	X_0	Y_1	Y_0	I/F
0	0	0	0	0
0	0	0	1	1 ✓
0	0	1	0	1 ✓
0	0	1	1	1 ✓
0	1	0	0	1 ✓
0	1	0	1	0
0	1	1	0	1 ✓
0	1	1	1	1 ✓
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1 ✓
1	1	1	1	0

$$I/F = X_1 * Y_1 + X_1 * Y_1 * X_0 * Y_0 + X_1 * Y_1 * X_0 * Y_0$$

PAZI: kod ovih 2k

brojeva posebno parci

koji je već, a koji manji

\Rightarrow RASPREŠI!

$$\begin{aligned} -2 &= 10 \\ -1 &= 11 \end{aligned}$$

$$I/F = X[1] * Y[1] + X[0] * Y[0] * (X[1] * Y[1] + X[1] * Y[1])$$

Zadatak 7. (14. 2008./2009.)

Implementirati automat. 4 bitni redač $Q[3..0]$ koji redom preprima vrijednosti #b 1001, #b 0011, #b 0111, #b 1111, #b 110, #b 1100. Inicijalno $INIT = #b 0000$. Realiziraj ga kao Mooreov. Nije potrebno koristiti .OUTF nego dodjeli vrijednosti izlaznim bitovima ostvariti STATE ASSIGNMENT izraze.

—PIN DECLARATION—

PIN 1 CLK

COMBINATORIAL ; INPUT

PIN 10 GND

)

PIN 20 VCC

;

PIN 11 IOE

;

PIN 12..15 Q[3..0]

REGISTERED ; OUTPUT

--- EQUATIONS ---

STATE

NOORE - MACHINE

$$\text{INIT} := /Q[3] * /Q[2] * /Q[1] * /Q[0]$$

$$ST1 := Q[3] * /Q[2] * /Q[1] * Q[0]$$

$$ST2 := /Q[3] * /Q[2] * Q[1] * \bar{Q}[0]$$

$$ST3 := /Q[3] * Q[2] * Q[1] * Q[0]$$

$$ST4 := Q[3] * Q[2] * Q[1] * Q[0]$$

$$ST5 := Q[3] * Q[2] * Q[1] * /Q[0]$$

$$ST6 := \bar{Q}[3] * Q[2] * /Q[1] * /Q[0]$$

DEFAULT BRANCH IN IT

$$\text{INIT} := VCC \rightarrow ST1$$

$$ST1 := \rightarrow ST2$$

$$ST2 := \rightarrow ST3$$

$$ST3 := \rightarrow ST4$$

$$ST4 := \rightarrow ST5$$

$$ST5 := \rightarrow ST6$$

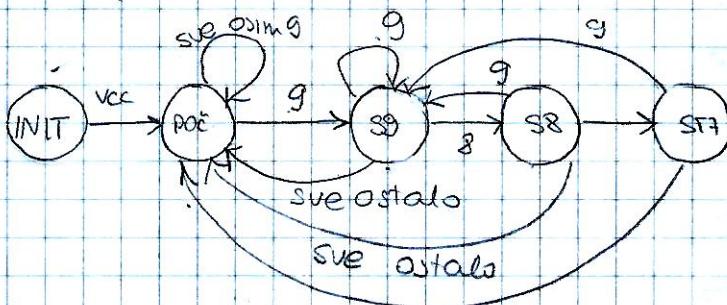
$$ST6 := \rightarrow ST1$$

Upozorenje: ako nije zadano (ali moralo bi biti) onda napraviti da se stanje koje nema definiran prijelaz vraća u početno stanje (ne INIT)

Zadatak 8. (ZIR 2011/2012.)

U PALASM-u opisati parser ulaznih numeričkih podataka.

Na liniji D[3..0] dolaze se 4 bitni binarni brojevi od 0 do 15, odnosno 0 do F. Prije svakog ulaznog brida tabla postavlja se novi broj. Segmenti A, G i D su segmentarnog pokazivača odgovaraju signalima SA, SG i SD. Potrebno je detektirati pojavljivanje niza 9, 8, 7. Ako se pojavi cijeli niz treba se upaliti SA, SG i SD. Ako detektiramo prva dva člana treba upaliti SA i SG, a ako detektiramo samo prvi treba upaliti samo SA. Ukoliko se ne detektiraju 1 ili 2 člana niza ne pojavi broj koji nastavlja niz trebati ugasiti sve segmente. Koristiti sintaksu za opis automata s konačnim brojem stanja i to za opis Mooreovog automata.



* Uzeto u obzir da može doći npr. 9 8 9 8 7

(napravljeno drugaći nego što piše u zadatku)

--- PIN DECLARATION ---

PIN 1	CLK	COMBINATIONAL ; INPUT
PIN 10	GND	;
PIN 20	VCC	;
PIN 11	/OE	;
PIN 2..5	D[3..0]	COMBINATIONAL ; INPUT
PIN 12	SA	REGISTERED ; OUTPUT
PIN 13	SG	REGISTERED ; OUTPUT
PIN 14	SD	REGISTERED ; OUTPUT

--- EQUATIONS ---

STATE

MOORE-MACHINE

$$INIT := /SA * /SG * /SD$$

$$POC := /SA * /SG * /SD$$

$$SG := SA * /SG * /SD$$

$$S8 := SA * SG * /SD$$

$$S7 := SA * SG * SD$$

DEFAULT BRANCH INIT.

$$INIT := VCC \rightarrow POC$$

$$\begin{aligned} POC' := & \text{DEVETKA} \rightarrow SG \\ & + \rightarrow POC \end{aligned}$$

$$\begin{aligned} SG' := & \text{DEVETKA} \rightarrow SG \\ & + \text{OSMICA} \rightarrow S8 \\ & + \rightarrow POC \end{aligned}$$

$$\begin{aligned} S8' := & \text{DEVETKA} \rightarrow SG \\ & + \text{SEDMICA} + S7 \\ & + \rightarrow POC \end{aligned}$$

$$\begin{aligned} S7' := & \text{DEVETKA} \rightarrow SG \\ & + \rightarrow POC \end{aligned}$$

CONDITIONS

$$\text{DEVETKA} = D[3] * /D[2] * /D[1] * D[0]$$

$$\text{OSMICA} = D[3] * /D[2] * /D[1] * /D[0]$$

$$\text{SEDMICA} = /D[3] * D[2] * D[1] * D[0]$$

Zadatak 9. (1. M1 2008./2009.)

Mooreov automat:

stanje	ST1	ST0
CRVEN	0	0
ZELEN	0	1
ŽUT	1	0

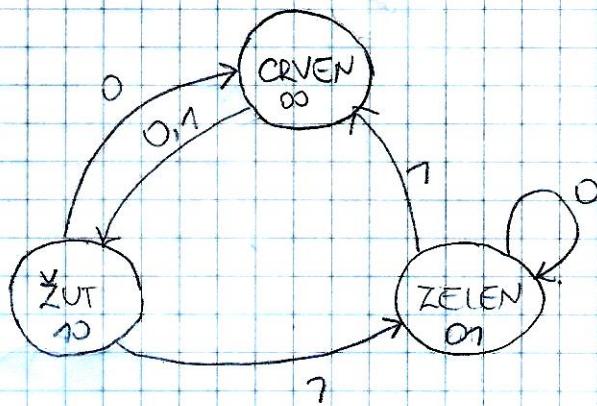
Prijedat u nov stanje ovisi o ulaznoj varijabli UL, a definirane su sa:

$$ST1 := 1ST1 * 1ST0$$

$$ST0 := 1ST1 * ST0 * UL +$$

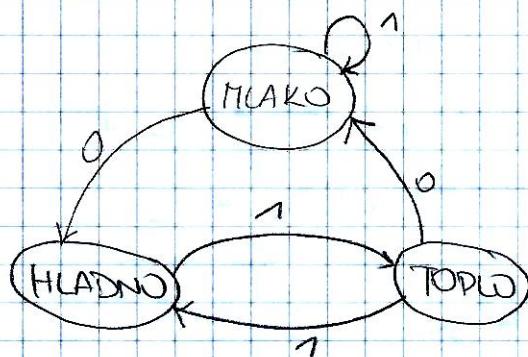
$$ST1 * ST0 * UL$$

TRENUTNO STANJE	ULAZ	INDUCE STANJE
00	0	10 Ž
00	1	10 Ž
01	0	01 2
01	1	00 C
10	0	00 C
10	1	01 2



Zadatak 10.

Mooreov automat. Ima 1-bitni ulaz UL. Opisati automat u jeziku PACASM. Nije potrebno računati dodjelu bitova stanja već samo jednačine prijelaza.



INIT := VCC \rightarrow MLAKO

MLAKO := UL \rightarrow MLAKO
+ \rightarrow HLADNO

TOPLO := UL \rightarrow HLADNO
+ \rightarrow MLAKO

HLADNO := UL \rightarrow TOPLO

Naponske razine

Objasnjenje
čitanje
instrukcija
iz
grafova

NAPONSKE RAZINE (TTL i CMOS)

Zadatak 1. (2008./2009.)

Potrebno je na \overline{Q} izlaz D-bistabila SN7474 spojiti čim veći broj istih takvih bistabila na odgovarajuće D ulaze. Koji je maximalni mogući broj takvih bistabila koji opterećuju \overline{Q} izlaz, koji i daje zadavaju ulazne i izlazne staticke karakteristike za sve sklopove (vezane za odnose ulaznih i izlaznih napona i struja u obje logičke ravine). Staticke karakteristike dane u prilogu. Odgovor obrankišti odgovarajućim nejednakostima.

NAPONI:

IZLAZ \longrightarrow ULAZ

$$U_{O\bar{H}} = [2,4, 3,6] \quad U_{I\bar{H}} = 2$$

$$U_{O\bar{L}} = [0,2, 0,6] \quad U_{I\bar{L}} = 0,8$$

Ujeti da sklop radi. i u najgornim uvjetima:

- $U_{O\bar{H}} > U_{I\bar{H}}$ ✓
- $U_{O\bar{L}} < U_{I\bar{L}}$ ✓

Spoj je moguć.

STRUJE:

$$\frac{I_{O\bar{H}}}{I_{I\bar{H}}} = \frac{-0,4 \text{ mA}}{40 \mu\text{A}} = -10$$

$$\frac{I_{I\bar{L}}}{I_C} = \frac{16 \text{ mA}}{-1,6 \text{ mA}} = -10$$

Moguće je spojiti 10 bistabila.

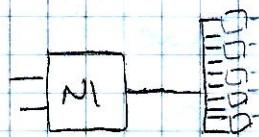
$$U_{GSV} = U_{O\bar{H}} - U_{I\bar{H}} = 0,4$$

$$U_{GSN} = U_{I\bar{L}} - U_{O\bar{L}} = 0,4$$

$$U_{GS} = \min(U_{GSV}, U_{GSN}) = 0,4$$

Zadatak 2. (2008/2009.)

Na izlaz dvoulažnih NI vrata SN7400 spojeni su oba ulaza 5 istih takvih NI vrata (dakle ukupno 10 ulaza). Koliko iznose izlazne struje visoke i niske razine na izlazu pogonskih NI vrata za ovaj spoj. Da li je ovakav spoj obrnutjen, te ako jest, kolika iznose granice smetnji niske i visoke razine. Statičke karakteristike dane u prilogu. Odgovor obrazložiti odgovarajućim izrazima.



NAPOMI:

IZLAZ :

$$U_{OH} = 2.4$$

$$U_{OL} = 0.4$$

ULAZ :

$$U_{IH} = 2$$

$$U_{IL} = 0.8$$

*

~~DA LI JE OVAJ SPOJ OBRAZLOŽITI~~

Da bi spoj bio moguć mora vrijediti:

$$U_{OH} > U_{IH} \quad \checkmark$$

$$U_{OL} < U_{IL} \quad \checkmark$$

\Rightarrow spoj je moguć

STRUJE:

$$\frac{I_{OH}}{I_{IH}} = \frac{-0.4 \text{ mA}}{40 \mu\text{A}} = -10$$

$$\frac{I_{OL}}{I_{IL}} = \frac{16 \text{ mA}}{-1.6 \mu\text{A}} = -10$$

Na izlaz NI sklopa može se spojiti 10 ulaza - znači 5 NI sklopova

$$U_{GSV} = U_{OH} - U_{IH} = 0.4$$

$$U_{GSN} = U_{IL} - U_{OL} = 0.4$$

$$U_{GS} = \min (U_{GSV}, U_{GSN}) = 0.4$$

Zadatak 3. (MI 2016/2017.)

Na ulaz GAL16V8 sklopa želimo spojiti izlaz logičkih vrata 74LS00. Tablice s DC karakteristikama su dane u nastavku. Potrebno je:

- dopasiti po kojim podatcima iz tablice se vidi da li možemo ostvariti ovaj spoj, te skicirati napomene razine za dobu logička stanja.
- Postaviti nejednadjelbe po kojima vidimo može li se ostvariti spoj.
- Izračunati koliko ulaza GAL sklopa može pogoniti jedan ulaz 74LS00?

U zadaci bi trebalo biti zadano V_{CC}
iz kojeg zaključujem
(min,max) iz tablice
ako nije zadano
gleđajući nagnosti
stavljanje i ortansko



NAPOMI:

IZLAZ:

$$U_{OH} = 2.5$$

$$U_{OL} = 0.5$$

ULAZ:

$$U_{IH} = 2$$

$$U_{IL} = 0.8$$

STRUJE:

$$I_{OH} = -0.4 \text{ mA}$$

$$I_{IL} = 8 \text{ mA}$$

$$I_{IH} = 10 \mu\text{A}$$

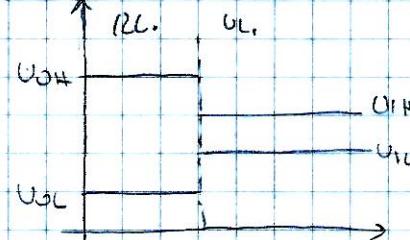
$$I_{IL} = -100 \mu\text{A}$$

Da bi spoj bio moguć mora vrijediti:

$$U_{OH} > U_{IH} \quad \checkmark$$

$$U_{OL} < U_{IL} \quad \checkmark$$

⇒ Spoj je moguć



$$\left| \frac{I_{OH}}{I_{IH}} \right| = 40$$

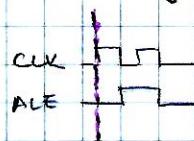
$$\left| \frac{I_{OL}}{I_{IL}} \right| = 8 \quad \times$$

moguće je spojiti 8

VREMENSKI DUGRAMI

Objašnjuje:

1. povlačimo crtkane linije na prvi rastući brid prije svakog ALE



(To su poluciklusi)

2. krećemo s desne strane prema lijevoj (brže je)

3. ispisujemo D i A za svaku crtkanu liniju

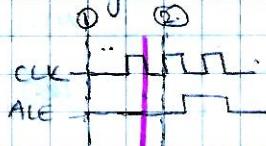
4. tražimo pojavu gdje su na 2 vrastopne crtkane linije

ISTI D i A

5. \Rightarrow pronašli smo jednobajtnu jednosciklusnu naredbu

(obj lijevi se; zato što zamenjuje prvi dohvati optoka)

6. na prvi padajući brid lijevo od druge povučemo punu liniju. To je naš početak strojnog ciklusa



7. Uzimavamo punu liniju na isti način ali za svaku DRUGU crtkanu liniju (1 strojni ciklus = 2 poluciklusa = 12 CLK)

ASEMBLER 8051

• Zadatak 1. (M1 2016/2017.)

Za sustav s mikrokontrolerom 8051 napisati program u asembleru koji svakih $50\ \mu\text{s}$ pročita podatak sa skupa priključka P1 i upiše ga u vanjsku podatkovnu memoriju na adresi FFFEh. Čitanje i upis podataka potrebno je izvesti u potprogramu koji poslužuje prekid brojila T0. Pretpostaviti da MC radi na taktu frekvencije 12 MHz.

Način brojila 2:

$$12\ \text{MHz} = 12 = 1\ \mu\text{Hz}$$

$$T = 1\ \mu\text{s}$$

u TMO ($2^8 - 256$) trebamo upisati $256 - 50 = 206$

CSEG AT 00h

jmp glavni

; prekid

CSEG AT 0Bh

mov A, P1

movx @DPTR, A

reti

glavni: mov DPTR, #FFFEh

mov TMOD, #0000010b

mov TMO, # 206 d

mov TLO, # 206 d

setb TCON.4

setb IE.1

setb IE.4

petja jump petja

} brojila

} prekidi

Zadatak 2. (LjR 2014./2015.)

Napišiasm za 8051 koji broji koliko puta se
195 d pojavljuje u gornjih 128 bajtova interne memorije.

Rezultat napiši u varijablu memoriju na adresu 2000h.

Program izvršiti jednom. Potrebno koristiti DJNZ i CJNE.

CSEG AT 00h

```
MOV A, #0h
MOV DPTR, #2000h
MOV R1, #128d
MOV R2, #FFh
```

* indirektno
adresiranje

procitaj: MOV R0, @R2 !

cjne R0, #195, oduzmi
inc A

oduzmi: dec R2

```
DJNZ R1, procitaj
MOVX @DPTR, A
```

kraj jump kraj

END

* uvijek moraju imati beskonačnu

petku jer bi inače program nastavio
čitati sljedeće lokacije

Zadatak 3. (JIR 20.)

Pregrodati koliko se puta broj 45 pojavljuje u prvih 200 lokacija vanjske podatkovne memorije. Rezultat upisati u skup priključaka P1. Program izvršiti samo jednom.

CSEG AT 00h

mov A, #001

mov D PTR, #00h

mov R0, #200h

ucitaj : mov X R1, @D PTR

CJNE R1, #45d, odurni

inc A

odurni: inc D PTR

DJNZ R0, ucitaj

MOV P1, A

END

* CJNE i DJNZ rade s

<Byte> i zato ne moremo

direktno koristiti D PTR

Zadatak 4. (MI 2015./2016. zima)

Napisati asmu za 8051 koji svako 50 µs poveća za 1 paritatak na adresi 1000h u vanjskoj podatkovnoj memoriji. Nakon što podatak dosegne 100, potrebno je ponovo krenuti od 0. Povećanje sadržaja memorije izvesti u potprogramu koji pozurjuje prekid brojila T0. Pretpostaviti da na početku rada na 1000h stoji 0. MC radi na $f = 12 \text{ MHz}$.

$$f = 12 \text{ MHz}$$

$$1T = 12 \text{ clk} \Rightarrow 12 \text{ MHz} : 12 = 1 \mu\text{s}$$

$$T = \frac{1}{f} = 1 \mu\text{s}$$

Način rada 2:

$$\text{THO} = 2^8 - 50 = 206$$

$$\text{TLO} = 206$$

CSEG AT 00h

jmp glavni

i prekid

CSEG AT 0Bh

movx A, @DPTR

cjne A, #100d, povećaj

mov A, #0

jmp kraj

povećaj: inc A

kraj: movx @DPTR, A

reti

; glavni

mov DPTR, #1000h

mov TMOD, #00000010b

mov THO, #206d

mov TLO, #206d

setb TCON.4

setb IE.1

setb IE.7

Petlja: jmp petlja

END

Zadatak 5. (ZIR 2012.)

* ASSEMBLER

Na 8051 spojene su dvije 8-bitne VJ iz kojih se čita jedinicu u vanjskom podatkovnom prostoru. Prva na adresi 0xFFFF, druga na 0xFFE. Osim VJ, na P1.0 spojena je svjetleća dioda koja svijetli dovođenjem visoke razine napona na priključak μC. Potrebno je neprekidno čitati podatke iz VJ i ako je podatak iz prve veći ili jednak od podatka iz druge upaliti svjetleću diodu. Napisati program u asembleru.

CSEG AT 00h

petlja: clr P1.0

MOV DPTR, #FFFF

MOVX A, @DPTR

MOV R0, A

MOV DPTR, #FFFE

MOVX A, @DPTR

MOV R1, A

CJNE R0, R1, misu

misu: JC petlja

upali: setb P1.0

jmp petlja

CJNE dest, src, label

CJNE postavlja carry flag:

dest > src C=0

dest < src C=1

nepotrebno!
BOJE SA SUB

PAŽI pročitaš 1. drugi podatak, onda
prije i naredba

SUBB A, R2 → treće postaviti

C=1 ako je R2>A

Zadataci u C-u (8051)

Zadatak 1. (ZI 2016./2017. letnji)

Na računalni sustav s MC 8051 spojen je 8 bitni analogno-digitalni pretvornik koji ima sljedeće priblijavičke:

TIP	OPIS
DO-D7 Izlazi	Sadrži vrijedi postatak za vrijeme kada je READY=0
START ulaz	Rastući broj pokreće analogno-digitalnu pretvorbu
READY izlaz	Automatski se postavlja u 0 nakon što je pretvorba završena, te u 1 nakon rastućeg broja na priblijavičku START.

Podatkovne linije pretvornika spojene su na skup priblijavičaka P1 mikrokontrolera. Priblijavičak READY spojen je na priblijavičak MC P3.2 koji ujedno predstavlja ulaz za vanjski prekid 0.

Priblijavičak START spojen je na P3.3.

Potrebito je napisati podršku u C-u koja u prekiduoj fiji vanjskog prekida 0 čita podatak s pretvornika i upisuje ga u vanjsku podatkovnu memoriju na lokaciju 0x3000. Vanjski prekid treba posluživati na pojavu broja. Nakon što je podatak pročitan, potrebno je ponovno pokrenuti pretvorbu.

```

#include <stdio.h>
#include "reg51.h"
void prekidna (void);

sbit ready = P3^2      → nepotrebno
sbit start = P3^3

volatile unsigned int xdata *izlaz;
izlaz = 0x3000;

```

```

void main (void) {
    TCON = 0x01;   → prekid EX0 se okida na prijelaz
    IE = 0x81;

    start = 0;     } osigurava prvo pokretanje pretvarbe
    start = 1;     } (0-1-0 je zato što moramo osigurati
    start = 0;     } rastuci brod) (1)

    while (1);
}

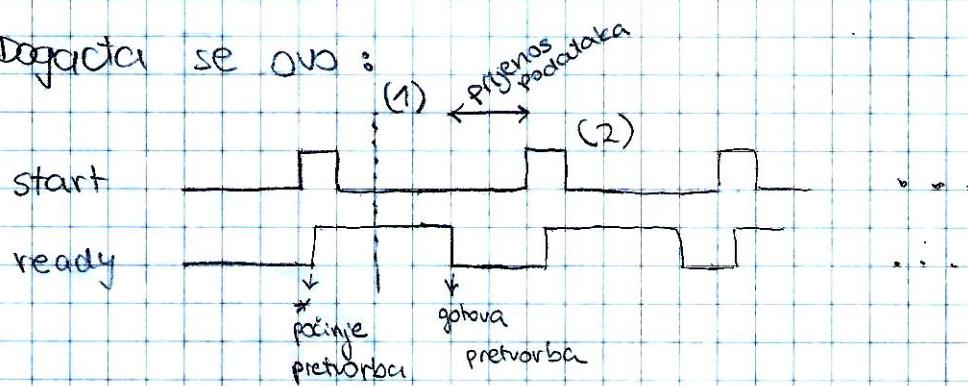
void prekidna (void) interrupt 0 {
    *izlaz = P1;

    start = 1;     } gotovi smo s prijenosom podataka i
    start = 0;     } trebamo pokrenuti novu pretvarbu
}

```

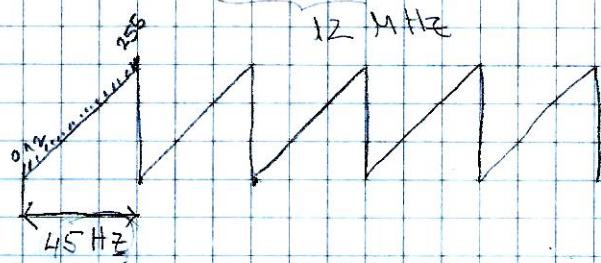
(2)

Dogadja se ovo:



Zadatak 2. (iz teme)

Na računalni sustav s μC 8051 spajen je 8 bitni D/A pretvornik koji se može u vanjskom memorijskom prostoru na adresi 0xFFFF. Napisati u C-u fju koja upisom niza vrijednosti 0, 1, 2, 3, ..., 255 u pretvornik generira pilastri valni oblik. Frekvencija pilastog signala treba iznositi 45 Hz. Upis u pretvornik treba realizirati pomoću prekidne fje koja poslužuje prekid brojila TO. Prepostaviti da frekv. faju μC iznosi 11.0592 MHz.



$$T = 1 / 45 \text{ Hz} = 0.022 \text{ s} \rightarrow \text{vrijeme za sve podatke}$$

$$256 \text{ podataka} \cdot 8 \text{ bita} = 2048 \text{ bita}$$

$$\text{brzina prijenosa} = \frac{2048 \text{ bita}}{0.022 \text{ s}} \cdot \frac{45.045}{45.045} = \frac{92.252 \text{ bit/s}}{1 \text{ s}}$$

$$\text{brzina prijenosa} = 92.252 \text{ bit/s}$$

$$0.022 = 22 \text{ ms} = 22.000 \mu\text{s}$$

$$\begin{array}{r} 2^{16} = 65536 \\ - 22000 \\ \hline \end{array}$$

$$43536 = 0x AAA0$$

```
# include <stdio.h>
```

```
# include "reg51.h"
```

```
Void prekidna (void);
```

```
Volatile Unsigned Mt xdata *pretvornik;
```

```
pretvornik = 0xFFFF;
```

```
Void main (void){
```

```
    TMOD = 0x01;
```

```
    TH0 = 0x AA;
```

```
    TL0 = 0x 10;
```

```
    IE = 0x 82;
```

```
    TR0 = 1;      for
```

```
    while (1);
```

```
}
```

```
Void prekidna (void) interrupt 1 {
```

```
    TH0 = 0xAA;
```

```
    TL0 = 0x10;
```

```
    TR0 = 1;      ?
```

```
    for (i=0 ; i< 256 ; i++) {
```

```
        *pretvornik = i;
```

```
}
```

```
}
```

Zadatak 3. (WIR 2014./2015)

Napišite C program za 8051 koji će na skupu priključaka P₀ generirati pravobutni napon perioda 50 ms. f = 12 MHz.

Rješiti pomoću prekidca i brojila. Program se vrati beskonačno.

```
#include <stdio.h>
```

$$1 \text{ clk} = 1 \mu\text{s}$$

```
#include "reg51.h"
```

$$50 \mu\text{s} = 50000 \mu\text{s}$$

```
void prekidna (void);
```

$$2^{16} = 65536 \rightarrow \text{stane u brojcu}$$

```
void main (void) {
```

$$65536 - 50000 = 15536$$

```
    TMOD = 0x01;
```

$$= 0x3C | B0$$

```
    TH0 = 0x3C;
```

TH0 TL0

```
    TL0 = 0xB0;
```

```
    IE = 0x82;
```

```
    TR0 = 1
```

```
    P1 = 0x00;
```

```
    while (1);
```

```
}
```

```
void prekidna (void) interrupt 1 {
```

```
    TH0 = 0x3C;
```

```
    TL0 = 0xB0;
```

```
    if (P1 == 0x00) {
```

```
        P1 = 0xFF;
```

```
    } else {
```

```
        P1 = 0x00;
```

```
}
```

Zadatak 4. (ZIR 2012.)

Na 8051 spojena je tipka kod koje je ishtrovanje rješeno sklopovski. Spojena je na P1.0. Potrebno je svakim pritiskom tipke invertirati stanje priključka 1 tog skupa 300 µs nabor što je tipka pritisnuta. Pretpostaviti da se tipka prihvata u intervalima mnogo većim od 300 µs. Za generiranje vremenskog intervala potrebno je koristiti prekid koji daje brojilo T0. ($f = 12 \text{ MHz}$)

Napisati program i odgovarajući prekidnu funkciju u C-u.

Brojilo: način rada 1:

$$f = 12 \text{ MHz} : 12 = 1 \text{ MHz} \Rightarrow T = 1/f = 1\mu\text{s}$$

$$2^6 = 65536 - 300 = 65236 \xrightarrow{\text{HEXA}} 0x \text{ FE} \{ \text{D4} \\ \text{TMO} | \text{TLO}$$

```
#include <stdio.h>
```

```
#include "reg51.h"
```

```
void prekidna(void);
```

```
sbit invertiraj = P1^1;
```

```
sbit tipka = P1^0;
```

```
void main (void) {
```

```
    TMOD = 0x01;
```

~~```
 TH0 = 0xFE,
```~~~~```
    TL0 = 0xD4;
```~~

```
    IE = 0x82;
```

```
    while (1) {
```

```
        if (tipka == 1) {
```

```
            TH0 = 1
```

TR0 = 0

```
void prekidna(void) interrupt 1 {
```

```
    if (invertiraj == 1) {
```

```
        invertiraj = 0;
```

```
    } else {
```

```
        invertiraj = 1;
```

```
}
```

```
}
```

FALI:

```
TH0 = 0xFE
```

```
TL0 = 0xD4;
```

Zadatak 5. (C)

Na 8051 spojene su dvije 8-bitne vj. Prva je spojena na P1, ona daje podatak. Druga prima podatak i vidi se na vanjskoj memoriji na adresi 0xFOOF. Treba svakog 50ms pročitati podatak iz prve i upisati ga u drugu. Za vrijeme koristi T0. $f = 12 \text{ MHz}$.

```
#include <stdio.h>
```

```
#include "reg51.h"
```

```
void prekidua(void);
```

```
volatile unsigned int xdata *primi;
```

```
primi = 0xFOOF;
```

```
void main(void) {
```

```
    TMOD = 0x01;
```

```
    TH0 = 0x3C;
```

```
    TL0 = 0x80;
```

```
    IE = 0x82;
```

```
    TR0 = 1;
```

```
    while (1);
```

```
}
```

$$2^{16} - 50000 = 15536$$

3C BC

```
void prekidua(void) {
```

```
    TH0 = 0x3C;
```

```
    TL0 = 0x80;
```

```
*primi = P1;
```

```
}
```

Zadatak 6. (21.2012. zima)

Na 8051 spojen je AD pretvarač na adresi 0xFFFF.

Treba neprestano čitati podatak s njega i ako se podatak promijeni ispisati "pročitan novi podatak".

Serijski treba ispisivati brzinom 9600 kbit/s.

ASSEMBLER:

CSEG AT 00h

; inicijalizacija brojila

mov TMOD, #20h

mov TH1, #FDh

mov TL1, #FDh

setb TCON.6

; inicijalizacija serije

mov PCON, #00h

clr SCON.7

setb SCON.6

setb SCON.4

mov DPTR, #FFFF

movx A, @DPTR

mov R0, A

provjera: mov A, @DPTR

CJNE A, R0, skoci

jmp provjera

skoci: mov R0, A

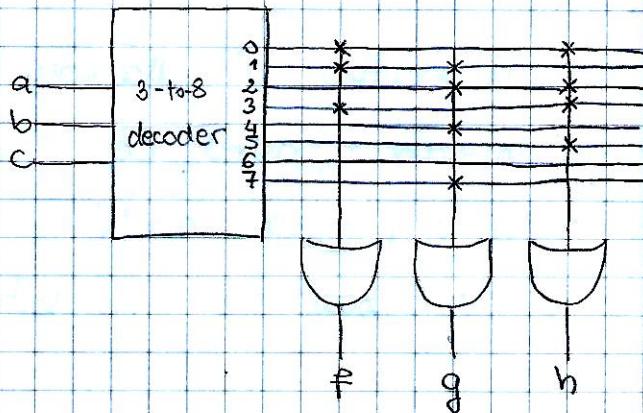
URS - zadaci -

[2017.]

Zadatak 1.

Zadane su funkcije $f = \sum m(0,1,3)$, $g = \sum m(1,2,4,7)$ i $h = \sum m(0,2,5,7)$. Implementirati pomoću PROM-a sa adresnim dekoderom (fiksnim) i poljem OR sklopova.

* najviši uniterm = 7 ($2^3 = 8$ trebamo dekoder 3-to-8)



Zadatak 2.

process (a)

variable c : std-logic

c := '0'

begin

c := a;

b <= c;

end process;

a, b su signali tipa std-logic, početne vrijednosti '0'.

Kolika je vrijednost signala b nakon što se dogodila promjena (event) na signalu a s '0' na '1' te ponovo s '1' na '0'. Objasni!

Razlika je u tome što se varijable odmah pridjele vrijednost, a signalu tek nakon izlaska iz procesa!

- $a = '0' \rightarrow '1' \Rightarrow b = 1$. U slasku u proces varijabla c odmah poprima vrijednost '1', a true i b
- $a = '1' \rightarrow '0' \Rightarrow b = 0$. Isti razlog.

Zadatak 3.

```

signal c : std_logic := '0';
...
process (a)
begin
  c <= a;
  b <= c;
end process;

```

Kolika je vrijednost signala b nakon što se dogodila promjena (event) na signalu a s vrijednosti '0' na '1' te ponovo s '1' na '0'.

Objasni!

* PAH što piše na početku!

U ovom slučaju c je signal.

- $a = '0' \rightarrow '1' \Rightarrow b = '0'$. c je na početku '0'. Uklonom u proces on neće promijeniti svoju vrijednost, ostat će '0' pa će zato i b biti '0'. Nakon izlaska iz procesa $c = '1'$.
 - $a = '1' \rightarrow '0' \Rightarrow b = '1'$ c je '1', zato će i b biti '1'.
- Izlaskom iz procesa $c = '0'$.

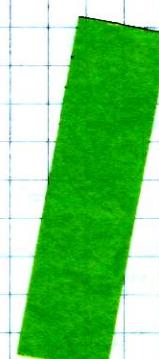
Zadatak 4.

Korišćenjem kombinacijske logike napišite VHDL kod (entitet i arh.) za izvedbu multipleksora 8 na 1 pri čemu je defaultna vrijednost na izlazu ona s ulaza (0).

```

entity Mux is
  port (
    in_data : in std_logic_vector (7 downto 0);
    selector : in std_logic_vector (2 downto 0);
    IZLAZ : out std_logic;
    );
  end MUX;
  enable : in std_logic;

```

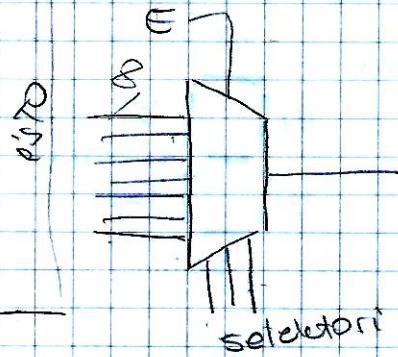


- architecture ARH of MUX is begin

with selector select

```

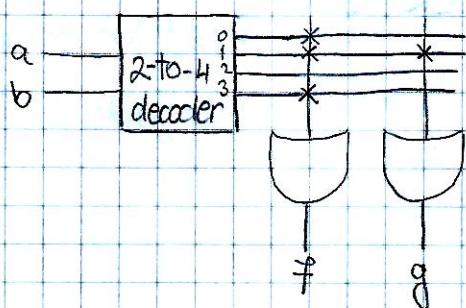
    idar2 <= in-data(7)      when "111"; and enable = '1'
    in-data(6)              when "110";
    in-data(5)              when "101";
    in-data(4)              when "100";
    in-data(3)              when "011";
    in-data(2)              when "010";
    in-data(1)              when "001";
    in-data(0)              when others;
    when enable = '0';
end ARH;
  
```



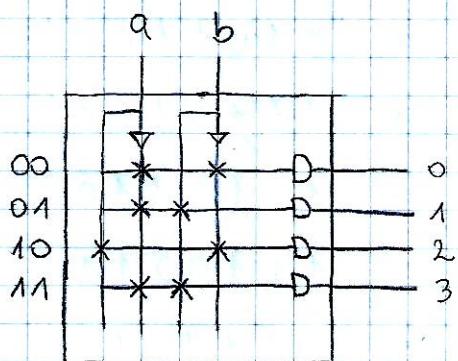
Zadatak 5.

Implementirati funkcije $f = \sum m(0,1,3)$; $g = \sum m(1)$ u PROM sklopu s 2 ulaza. Skicirati shemu sklopa.

| A | B | f | g |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |



Shema dekodera:

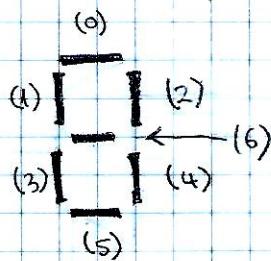


Zadatak 6.

Ispisati u VHDL-u komponentu (entitet i arhitekturu) za 7-segmentni pokaznik konstenjem kombinacijske procesne logike.

Ulez je signal tipa int koji prihvata vrijednosti od 0-9, a izlaz je std-logic-vector veličine 7 koji predstavlja kod za 7 segmentni pokaznik kako je prikazano na slici:

izlazi:



SLJED NAREDBI BITAN

entity DECODER is

port(

- number : in integer range 0 to 9;

 output : out std-logic-vector (6 downto 0);

);

end DECODER;

architecture ARH of DECODER is

begin

process (number)

begin

case number is

. when 1 => output <= "0010100";

. when 2 => output <= "1101101";

. when 3 => output <= "1110101";

. when 4 => output <= "1010110";

. when 5 => output <= "1110011";

. when 6 => output <= "1111011";

. when 7 => output <= "0010101";

. when 8 => output <= "1111111";

* ne treba povući signal ili
varijabla?

6 5 4 3 2 1 0

+ 0 0 1 0 1 0 0

1 1 0 1 1 0 1

1 1 1 0 1 0 1

1 0 1 0 1 1 0

1 1 1 0 0 1 1

1 1 1 1 0 1 1

0 0 1 0 1 0 1

1 1 1 1 1 1 1

```

when 9 ⇒ output ← '1110111'
when others ⇒ output ← '0000000'
end case;
end process;
end ARIK;

```

Zadatak 7.

Za PicoBlaze procesor napisati sljedeći prekidni potprogram.

Na svaki zahtjev za prekid potrebno je očitati stavlje vanjske jedinice koja je spojena na IN-PORT. Na dobivenom podatku ispituje se parnost: ako je paran potrebno je povećati brojač parnih podataka u ScratchPad memoriji na adresi hex(10), a ukoliko je neparan povećati brojač neparnih podataka koji se nalazi u ScratchPad memoriji na adresi hex(11).

| | |
|---|---|
| ADDRESS 000
CONSTANT IN-PORT, <u>00</u>
ENABLE INTERRUPT
; brojaci
LOAD S1, 00
LOAD S2, 00
CEKA) JUMP CEKA) | PREKID INPUT S0, IN-PORT ; spremi
AND S0, 01 ; maska
COMPARE S0, 00 ; usporedi
JUMP Z, PARAN
; prepostavljam
; da je tribalo
; bit zadano
; u NEPARAN ADD S1, 01
; zadatku
STORE S1, 10
RETURN

PARAN ADD S2, 01
STORE S2, 11
RETURN

; prekidna adresa |
|---|---|

* brojevi se pišu u hexadekatskom obliku

* nepotrebni jer AND samo po sebi postavlja zastavice

Zadatak 8.

U VHDL-u projektirati 16-bitni brojač sa asinkronim resetom.

Brojač na ulazu ima signal širine 4 bita koji definira podatak kojim se trenutna vrijednost brojača inkrementira.

Brojač omogućuje brojanje naprijed i natrag koji se kontrolira pomoću priključka UP-DOWN ('1'-inkrementira, '0'-dekrementira) te tako, brojač na ulazu ima ENABLE priključak, koji omogućuje brojanje ako je postavljen u '1'. Brojač ima i asinkroni reset koji sve izlazne signale postavlja u vrijednost '0'.

```
entity BROJAC is
    port (
        povecaj : in std-logic-vector (3 downto 0);
        UP-DOWN : in std-logic;
        ENABLE : in std-logic;
        RESET : in std-logic;
        COUNTER : out std-logic-vector (15 downto 0);
        CLK : in std-logic
    );
end BROJAC;
```

```
architecture ARH of BROJAC is
```

```
begin
```

```
variable count := std-logic-vector (15 downto 0);
```

```
variable dodaj := std-logic-vector (3 downto 0);
```

```
dodaj := povecaj;
```

```
process (RESET, CLK, ENABLE, UP-DOWN)
```

```
begin
```

```
if RESET = '1' then
```

```
count = "0000 0000 0000 0000";
```

```
endif
```

PONOV SINKR.
ASINKR. RESET

treba biti
unutar
procesa

```

if CLK'event() and CLK='1' then
    if ENABLE = '1' then
        if UPDOWN = '1' then
            count = count + dodaj;
        elsif UPDOWN = '0' then
            count = count - dodaj;
        endif
    endif
    counter <= count;    nemog zaboraviti !
end process;
end ARIT;

```

Eadatak 9.

Fadana je tablica vrijednosti za enkoderski sklop. Potrebno je implementirati sklop u VHDL-u pomoću kombinacijske logike.

| ULAZ | IZLAZ |
|-------|-------------------|
| 0 0 0 | - 0 0 0 0 0 0 1 |
| 0 0 1 | - 0 0 0 0 0 0 1 0 |
| 0 1 0 | - 0 0 0 0 0 1 0 0 |
| 0 1 1 | - 0 0 0 0 1 0 0 0 |
| 1 0 0 | - 0 0 0 1 0 0 0 0 |
| 1 0 1 | - 0 0 1 0 0 0 0 0 |
| 1 1 0 | - 0 1 0 0 0 0 0 0 |
| 1 1 1 | - 1 0 0 0 0 0 0 0 |

SLJED NAREDBI NIVE BITAN

entity dekoder is

port (

ulaz : in std_logic_vector (2 downto 0);

izlaz : out std_logic_vector (7 downto 0);

);

end dekoder;

architecture ARH of decoder is
begin

with ulaz select

```
    idlaz <= "000000001" when "000";
    "000000010" when "001";
    "000001000" when "010";
    "000010000" when "011";
    "000100000" when "100";
    "001000000" when "101";
    "010000000" when "110";
    "100000000" when "111";
    NULL when others;
```

end ARH;

Zadatak 10.

Na PicoBlaze spojena je 1 ulazna VJ (port-id 0x80) i 2 izlazne VJ (0x60 i 0x40) te vremenski sklop koji je spojen na prekidnu liniju PicoBlaze procesora. Na svaki prekid dobioven od prekidne jedinice treba pročitati podatak s ulazne jedinice i ispitati paritet. Ako je broj jedinica paran podatak se šalje na vanjsku jedinicu s port-id 0x60, a u suprotnom na 0x40. Program se održava beskonačno.

ADDRESS 000

CONSTANT PORT-ID1, 80 ; ulazna

CONSTANT PORT-ID2, 60 ; parni

CONSTANT PORT-ID3, 40 ; neparni

[PARITET = broj jedinica]

ENABLE INTERRUPT

PETLJA JUMP PETLJA

```

PREKID      INPUT S0 , PORT_ID1
TEST        S0,FF
JUMP        C , NEPARAN
PARAN      OUTPUT S0 , PORT_ID2
RETURNI
NEPARAN    OUTPUT S0 , PORT_ID3
RETURNI
; prekidna adresa
ADRESS 3FF
JUMP PREKID

```

NAREDBA TEST

TEST sx, kk
ili
TEST sx, sy

→ uzima podatke i s njima radi operaciju AND.
Rezultat spremi u privremeni registar

ZASTAVICE

C - postavlja se u '1' ako je broj jedinica u privremenom registru NEPARAN

Z - postavlja se ako su svi bitovi rezultata jednakci '0'

Zadatak 11.

U VHDL-u projektirati kontrolnu jedinicu za semafor pomoću stroga stanja kako je to pokazano na predavanjima (dva procesa).

Ulaz u jedinicu je signal vremenskog signala učtenja, a izlazi (njih 3) se koriste za paljenje signalizacijskih svjetala (crveno, žuto, zeleno). Stroj mijenja stanje tako da se dobije sljedeći redoslijed svjetala :

1. crvena
2. crvena i žuta
3. zelena
4. žuta



Sklop ima i asinkroni reset koji postavlja stanje u 1. crvena.

* signalizacijska svjetla

| | | |
|---|---|---|
| 2 | 1 | 0 |
| C | X | Z |

library IEEE

use IEEE.std_logic_1164.all;

entity SEMAFOR is

port (

CLK: in std_logic;

RESET: in std_logic;

SVJETLO: out std_logic_vector (2 downto 0);

)

end SEMAFOR;

architecture ART of SEMAFOR is

type STATE_TYPE is (S_CRVENO, S_CRVENO_ZUTO, S_ZUTO, S_ZELENO)

signal trenutno: STATE_TYPE : S_CRVENO;

signal sljedece: STATE_TYPE : S_CRVENO;

begin

process (CLK, trenutno)

begin

if CLK'event() and CLK = '1' then

if trenutno = S_CRVENO then

SVJETLO = '100';

* Uspješno

rješenje s

case

naredbom!

elif trenutno = S_CRVENO_ZUTO then

SVJETLO = '110';

sljedece <= S_ZELENO;

elif trenutno = S_ZELENO then

SVJETLO = '001';

sljedece <= S_ZUTO;

if trenutno = S_ZUTO then
svjetlo = '010'
sljedece <= S_CRVENO
endif;
endif;
end process;

process (CLK, RESET)
begin
if RESET = '1' then
trenutno = S_CRVENO ;
elsif CLK'event and CLK = '0' then
trenutno <= sljedece ;
endif;
end process;
end AR#;

Zadatak 12.

U VHDL-u je potrebno projektirati komponentu koja ispituje jednakost dva ulazna signala bit po bit. Signali su tipa std-logic koja je definirana pomocu generic ključne riječi u deklaraciji entiteta, a izlaz je tipa std-logic koji ima vrijednost '1' ukoliko su signali jednaki, a '0' ako nisu.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity KOMPARATOR is generic ( WIDTH : integer := 8 );
port (
    ulaz1: in std-logic-vector ( WIDTH-1 downto 0 );
    ulaz2: in std-logic-vector ( WIDTH-1 downto 0 );
    izlaz: out std-logic );
end KOMPARATOR;
```

```
architecture ARH of KOMPARATOR is
```

```
begin
```

```
process (ulaz1, ulaz2)
```

```
variable temp: bit;
```

```
temp = '1';
```

```
begin
```

```
for i in n'range loop
```

```
if ulaz1(i) != ulaz2(i) then
```

```
temp = '0';
```

```
exit;
```

```
endif;
```

```
end loop;
```

izlaz <= temp;

end process;

end ARH;

Zadatak 13.

U VHDL-u projektirati komponentu koja računa broj vodećih nula u ulaznom signalu tipa std-logic veličine 32 bita. Izlazni signal je tipa std-logic veličine 5 bitova.

```
library IEEE;
```

```
use IEEE. std-logic-1164.all
```

```
entity NULE is
```

```
port (
```

```
    ulaz : in std-logic-vector (31 downto 0);
```

```
    izlaz : out std-logic-vector (4 downto 0);
```

```
);
```

```
end NULE;
```

```
architecture ARH of NULE is
```

```
begin
```

```
process (ulaz)
```

```
variable counter : std-logic-vector (4 downto 0);
```

```
counter := '00000'
```

```
for i in 0 to 31 loop
```

```
    if ulaz(31-i) = '0' then
```

```
        counter = counter + 1;
```

```
    else
```

```
        exit;
```

```
    end loop;
```

```
    izlaz <= counter;
```

```
*
```

```
* end process;  
end ARH;
```

Zadatak 14.

U VHDL-u treba implementirati sljedeću logičku funkciju na razvojnoj pločici Spartan 3E Starter Kit: preklopniči SW2-SW0 su ulazi, signali LD3-LD1 predstavljaju izlaze logičke funkcije, koji su spojeni na LED diode.

| Izlaz | Funkcija |
|-------|---------------------------------|
| LD1 | SW0 or SW1 |
| LD2 | SW1 and SW2 |
| LD3 | (SW0 xor SW1) and (SW0 xor SW2) |

```
library IEEE;  
use IEEE.std-library-1164.all;
```

```
entity LEDICE is  
port(  
    SW : in std-logic-vector (2 downto 0);  
    LD : out std-logic-vector (2 downto 0)  
);  
end LEDICE;
```

```
architecture ARH of LEDICE is  
begin  
    LD(0) <= SW(0) or SW(1);  
    LD(1) <= SW(1) and SW(2);  
    LD(2) <= (SW(0) xor SW(1)) and (SW(0) xor SW(2));  
end ARH;
```

Zadatak 15.

- Na PicoBlaze su spojene 2 ulazne VJ (port-id = 0x20, 0x40) te 1 izlazna VJ (port-id 0x60) te vremenski sklop koji je spojen na prekidnu liniju. Na svaki prekid čitaju se podaci s ulaznih VJ. (Podaci su u 8-bitnom 2k formatu)
- Na prvi prekid čita podatak s VJ1, a na drugi s VJ2. Na treći prekid potrebno je poslati podatak s VJ3. Program se vrati beskonечно. Ako je podatak s VJ1 neparan, onda se na podatak s VJ2 šalje nepromijenjen na VJ3. Ako je podatak VJ1 neparan, onda se na VJ3 šalje negirana vrijednost podatka priuđenog sa VJ2.

a) PROGRAM ZA PROCESOR

ADDRESS 000

CONSTANT VJ1, 20

* 0A2

tu ide hexa vrijednost
(ne treba 0x)

CONSTANT VJ2, 40

CONSTANT VJ3, 60

ENABLE INTERRUPT

LOAD S0, 00 ; brojač prekida

ADDRESS 3FF

ADD S0, 01 ; povećaj brojač

XOR S0, 01

JUMP Z, PRVI

XOR S0, 02

JUMP Z, DRUGI

JUMP TRECI

PRVI: INPUT S1, VJ1
RETURN1

DRUGI: INPUT S2, VJ2
RETURN2

TRECI: AND S1, 01

JUMP Z, PARAN

NEPARAN: XOR S2, FF

ADD S2, 01

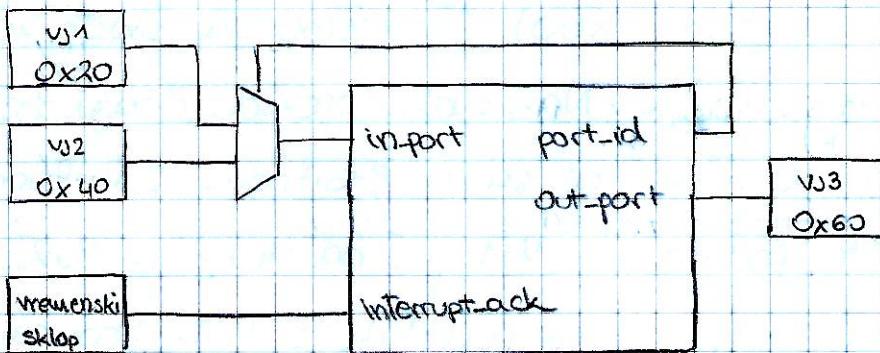
} DVOJNI KOMPLEMENT

PARAN: STORE S2, VJ3

LOAD

RETURN1

b) Nacrtati blok shemu povezivanja komponenti



c) Napisati sve procese koji će obradivati vanjske jedinice i prekidove jedinicu. Celokupni kod za top-level entitet i arhitekturu.
- zadavne deklaracije

end component;

component vanjska_izlaz is
port (data: out std_logic;
clk: in std_logic;

begin

 preceding process (clk)

begin

 if clk'event and clk='1' then

 if

)

)

)

Zadatak 16. - POGLEDATI UPUTU ZA 2. CIKLUS LABORATORIJE

Na računalni sustan s μC 8051 spojena je tipka kod koje je istitrovavajuće gješeno sklopovski. Tipka je spojena na pribljučak 2 skupa P3 koji predstavlja ulaz za vanjski prekid 0. Potrebno je svakim pritiskom tipke invertirati stanje pribljučka 1 istog skupa, 300 μs nakon što je tipka pritisnuta. Za generiranje vremenskog intervala potrebno je koristiti vanjski prekid 0 i prekid koji daje brojilo T0, uz pretpostavku da je $f = 12 \text{ MHz}$. Napisati program i odgovarajuće prekide u C++u. Pretpostaviti da se tipka pritiska u intervalima mnogo većim od 300 μs.

NAČIN 1:

```
#include <stdio.h>
#include <reg51.h>
void prekid (void);
void brojilo (void);
sbit tipka = P3^2;
sbit izlaz = P3^1;
```

$$f = 12 \text{ MHz}$$

$$12 \text{ perioda} = 1 \text{ strojni ciklus}$$

$$12 \cdot T = 12 \cdot \frac{1}{f} = 1 \mu\text{s}$$

nama treba 300 μs

$$\text{max stanje brojila} = 2^6 = 65536$$

```
void main (void) {
    TMOD = 0x01;
    IE = 0x83;
    while(1);
}
```

$$\Rightarrow 65536 - 300 = 65236 = \text{FE D4}_{(16)}$$

| |
|----------------------|
| T _{H0} = FE |
| T _{L0} = D4 |

* broji se prema FF

```
void prekid (void) interrupt 0 {
    TH0 = 0xFE;
    TL0 = 0xD4;
    TR0 = 1;
}
```

```
Void brojIco(void) {
```

```
    TR0 = 0;
```

```
    if (izlaz2 == 0)
```

```
        izlaz2 = 1;
```

```
    else
```

```
        izlaz2 = 0;
```

```
}
```

Zadatak 17.

Na računalni sustav s MC 8051 spojen je AD pretvarač na adresi 0x FFFF. Treba neprestano čitati podatak s AD pretvarača i ako se on promjeni ispisati preko serijske veze "Procitan novi podatak".

Serijski treba ispisivati brzinom 9 600 kbit/s.

```
#include < stdio.h >
```

```
#include "reg51.h"
```

```
volatile unsigned char xdata *yj = 0xFFFF;
```

```
int main (void) {
```

```
    data char temp;
```

```
    TMOD = 0x20;
```

```
    TH1 = 0xFD;
```

```
    TL1 = 0xFD;
```

```
    TR1 = 1;
```

```
    SCON = 0x52;
```

```
    do {
```

```
        if (temp != *yj) {
```

```
            printf ("Procitan novi podatak \r\n");
```

```
            temp = *yj;
```

```
} while (1);
```

Zadatak 18.

Na PicoBlaze spojene su dvije ulazne VJ (0×20 , 0×40) i jedna izlazna (0×60). Na svaki signal s tipkala spojenog na najniži bit na adresi 0×21 treba pročitati podatak s ulaznih jedinica (8-bitni 2k format). Na prvi signal čita se podatak s VJ1, a na drugi signal s VJ2. Na treći signal treba podatak poslati na VJ3 te postupak ponavljati beskonačno. Ukoliko je podatak poslan s VJ1 paran onda se podatak s VJ2 šalje nepromijenjen na VJ3. Ako je podatak neparan odna se šalje negirana vrijednost podatka s VJ2.

a) program za procesor

```
ADDRESS 000
CONSTANT VJ1, 20
CONSTANT VJ2, 40
CONSTANT VJ3, 60
CONSTANT TIPKALO, 21
LOAD $0,00
PETLJA LOAD S1,TIPKALO
        COMPARE S1, 01
        JUMP NZ PETLJA
        ADD $0, 01
        COMPARE $0, 01
        JUMP Z PRVI
        COMPARE $0,02
        JUMP Z DRUGI
        JUMP TRECI
```

} petlja čeka signal
s tipkala

*treba li mojda maska?

(radi ovog 'najniži bit') *popavi*

PRVI INPUT S2, VJ1

JUMP PETLJA

DRUGI INPUT S3, VJ2

JUMP PETLJA

TRECI AND S2, 01

JUMP ZA PARAN

NEPARAN XOR S3, FF

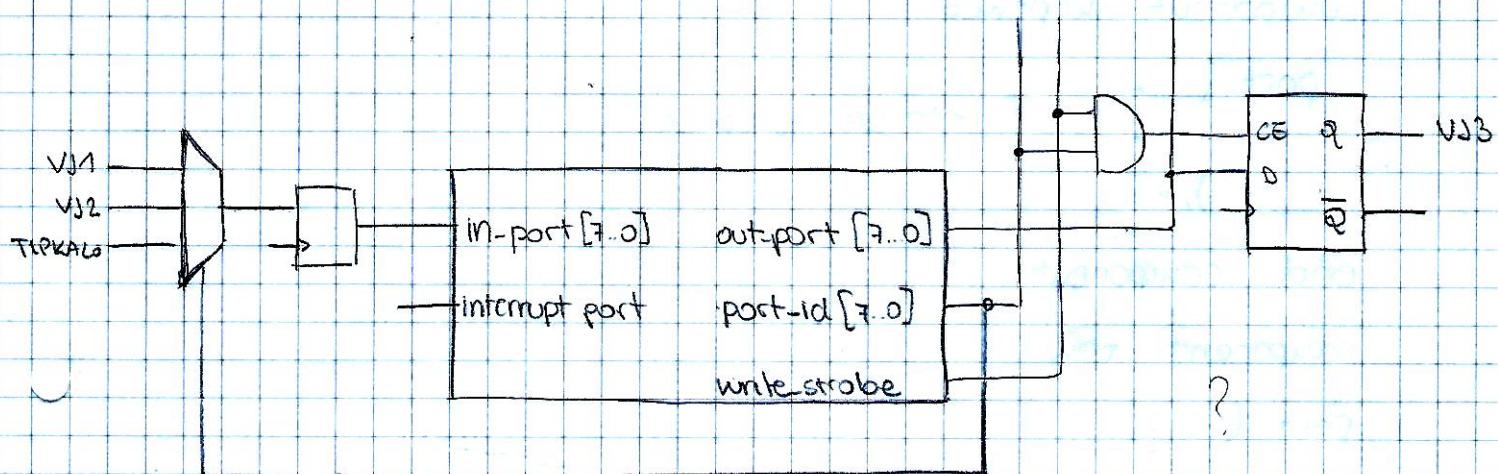
ADD S3, 01

PARAN OUTPUT S3, VJ3

LOAD S0, 00

JUMP PETLJA

b) shema



c) VHDL - zadane su deklaracije komponenti "kpsm3" i "rom"

```
entity top-level is
port (
    data-vj1 : in std-logic-vector (7 downto 0);
    data-vj2 : in std-logic-vector (7 downto 0);
    data-vj3 : out std-logic-vector (7 downto 0);
    tip_kalo : in std-logic;
    clk : in std-logic;
);
end top-level;
```

architecture ARH of top-level is

~~begin~~

component kpsm3

```
port ( : *to se prepiše
);

```

end component;

component rom

```
port ( : )
;
```

end component

signal address-signal

signal instruction-signal

signal port_id-signal

signal write_strobe-signal

signal out_port-signal

signal read_strobe-signal

signal in_port-signal

signal interrupt-signal

signal interrupt-ack-signal

signal reset-signal

begin

processor : kcprom3 port map

(address \Rightarrow address-signal

:

);

program : rom port map

(address \Rightarrow address-signal,

instruction \Rightarrow instruction-signal

clk \Rightarrow clk

);

vlazne-jedinice : process(clk)

begin

if clk'event and clk = '1' then

if port_id = '0010 0000' then (20)

in-port-signal <= data_vj1;

elif port_id = '0100 0000' then (40)

in-port-signal <= data_vj2;

elif port_id = '0100 0001' then

in-port-signal <= tipka0;

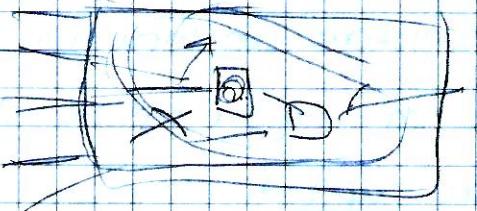
else

in-port-signal <= "xxxx xxxx";

endif;

endif;

end vlazne-jedinice;



* nastavak
na str
16. \rightarrow

Zadatak 19.

Na računalni sustav s μC 8051 spojeni su 8-bitni AD pretvarač i DA pretvarač koji se vide u vanjskom memoriskom prostoru na adresama 0xFFFF i 0xFFE. Na ulaz za vanjski prekid 0 spojen je izvor tabla uvrštenih podataka. Treba napisati program u C-u te odgovarajuću prekidnu funkciju koja kad dođe zahvat za prekid pročita podatak iz AD pretvarača i upiše ga u DA-prevarač.

```
#include <stdio.h>
```

```
#include "reg51.h"
```

```
void prekid (void);
```

```
void main (void) {
```

```
    IE = 0x81;
```

```
    while (1);
```

```
}
```

```
void prekid (void) interrupt 0 {
```

```
    volatile unsigned char xdata *ulaz, *izlaz;
```

```
    ulaz = 0xFFFF;
```

```
    izlaz = 0xFFE;
```

```
    *izlaz = *ulaz;
```

```
}
```

Zadatak 20.

Napisati program u C-u koji na priključku P1 skupa P1 daje pravokutni signal perioda 200 μs. Pretpostaviti da je MC radi na taktu frekvencije 12 MHz.

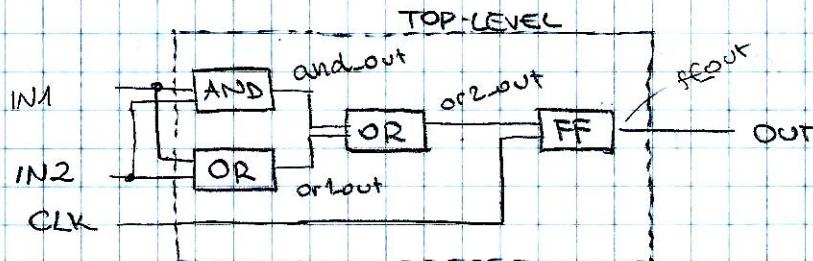
```
#include <stdio.h>
#include "reg51.h"
void prekid(void);
sbit prikljucak = P1^0;
void main (void) {
    TMOD= 0x01,
    TH0 = 0xFF;
    TL0 = 0x38;
    IE = 0x82;
    TR0 = 1;
```

$$2^{16} - 200 = 65336 \\ = FF38$$

```
void prekid (void) interrupt 1 {
    TR0=0;
    if (prikljucak == 0) prikljucak = 1;
    else prikljucak = 0;
}
```

Zadatak 21.

VHDL. Logičke blokove AND i OR potrebno je zasebno definirati entitetom i arhitekturom te ih koristiti kao komponente u top-level entitetu. Komponenta FF predstavlja flip-flop register i potrebno je opisati pomoću procesa u arhitekturi top-level entiteta.



* ne znau jer
slika dobra,
- takva je na f2

entity and1 is

```
port (
    x : in std-logic;
    y : in std-logic;
    z : out std-logic;
);
```

end and1;

architecture beh_and1 of and1 is

begin

$z \leq '1'$ when $x='1'$ and $y='1'$ else ' 0 ';

end beh_and1;

entity or1 is

```
port (
    x : in std-logic;
    y : in std-logic;
    z : out std-logic;
);
```

end or1;

entity flip-flop is

```
port (
    x : in std-logic;
    clk : in std-logic;
    y : out std-logic;
);
end flip-flop;
```

architecture beh_ff of flip-flop is

begin

process (clk)

begin

if clk'event and clk = '1' then

$y \leftarrow x$

end if;

end process;

end architecture;

architecture beh-or1 is

begin

$Z \leq '1'$ when $x='1'$ or $y='1'$ else '0';

end beh-or1;

entity top-level is

```
port ( in1 : in std-logic;  
      in2 : in std-logic;  
      clk : in std-logic;  
      idaz : out std-logic;  
 );
```

end top-level;

architecture beh-toplevel of top-level is

component and1

```
port ( x : in std-logic;  
      y : in std-logic;  
      z : out std-logic;  
 );
```

end component;

component or1

```
port ( x : in std-logic;  
      y : in std-logic;  
      z : out std-logic;  
 );
```

end component;

component flip-flop

```
port ( x : in std-logic;  
      clk : in std-logic;  
      y : out std-logic  
 );
```

signal and-out, or1-out, or2-out, : std-logic;

begin

and-1: and1 port map ($x \Rightarrow \text{in } 1$,
 $y \Rightarrow \text{in } 2$,
 $z \Rightarrow \text{and-out}$);

or1: or1 port map ($x \Rightarrow \text{in } 1$,
 $y \Rightarrow \text{in } 2$,
 $z \Rightarrow \text{or1-out}$);

or2: or1 port map ($x \Rightarrow \text{and-out}$,
 $y \Rightarrow \text{or1-out}$,
 $z \Rightarrow \text{or2-out}'$)

process (clk)

begin

if clk'event and clk='1'

then z = or2-out;

end if;

end process;

end beh-top-level;

ff1: flip-flop port map (

$x \Rightarrow$

clk \Rightarrow clk;

$y \Rightarrow f$

Zadatak 22.

Na PicoBlaze spojena je ulazna VJ1 (port-id = 0x80) i izlazna VJ2 (0x40) te LCD jedinica (0x60). Potrebno je isprojektirati sustav: Sa vanjske jedinice VJ1 se čitaju 8 bitni podaci u 2k formatu. Ako je priuđeni podatak negativan, potrebno je na LCD ispisati "NEG", a ako je pozitivan ili nula na LCD treba napisati "POZ". Apsolutnu vrijednost podatka treba poslati na VJ2.

ADDRESS 000

CONSTANT VJ1

VJ2

LCD

P

O

Z

N

E

G

CONSTANT clrdisp, ON ; izčiđuje LCD-a

INTERRUPT ENABLE

PET1JA JUMP PET1JA

PREKID INPUT S0, VJ1



LOAD S1, S0

*kad su podaci u

AND S1, S0

2k → najviši bit

JUMP F, POZ

je bit za predznak

NEG

LOAD S2, clrdisp

1-neg
0-pož

OUTPUT S2, LCD

LOAD S2, N

OUTPUT S2, LCD

LOAD S2, E

OUTPUT S2, LCD

LOAD S2, G

OUTPUT S2, LCD

SUB S0, 01

XOR S0, FF

} pretvaraće u poz broj

OUTPUT S0, VJ2

RETURN1

POZ

LOAD S2, clrdisp

OUTPUT S2, LCD

LOAD S2, P

OUTPUT S2, LCD

LOAD S2, O

OUTPUT S2, LCD

LOAD S2, Z

OUTPUT S2, LCD

OUTPUT S0, VJ2

RETURN1

ADDRESS 3FF

JUMP PREKID

Nastavak 18.

```
izlazna_vj : process(clk)
begin
    if clk'event and clk='1' then
        if write_strobe='1' then
            if port_id = '0110 0000' then (60)
                data_vj3 <= out_port_signal;
            endif;
        endif;
    endif;
end izlazna_vj;
end beh_top_level
```