ZADATAK 1.

Na računalni sustav s mikrokontrolerom 8051 spojena je tipka kod koje je istitravanje riješeno sklopovski. Tipka je spojena na priključak 2 skupa priključaka P3 koji predstavlja ulaz za vanjski prekid 0. Potrebno je svakim pritiskom tipke invertirati stanje priključka 1 istog skupa, 300 us nakon što je tipka pritisnuta. Za generiranje vremenskog intervala potrebno je koristiti vanjski prekid 0 i prekid koji daje brojilo T0, uz pretpostavku da frekvencija takta iznosi 12 MHZ. Napisati program i odgovarajuće prekidne funkcije u C jeziku. Pretpostaviti da se tipka pritiska u intervalima mnogo većim od 300 us.

# include <stdio.h>

# include <reg51.h>

void prekid(void);

void brojilo(void);

sbit tipka = P3^2;

sbit izlaz = P3^1;

void main(void){

TMOD = 0x01;

IE = 0x83;

while(1);

}

void prekid(void) interrupt 0{

TH0 = 0xFE;

TL0 = 0xD4;

TR0 = 1;

}

void brojilo(void) interrupt 1{

TR0 = 0;

if(izlaz == 0)

izlaz = 1;

else

izlaz = 0;}

ZADATAK 5.

U programskom jeziku VHDL potrebno je implementirati sljedeću logičku funkciju na razvojnoj pločici Spartan-3E Starter Kit: Preklopnici SW2-SW0 predstavljaju ulaze u logičku funkciju, dok signali LD3-LD1 predstavljaju izlaze logičke funkcije, koji su spojeni na LED diode, kako je prikazano u tablici.

IZLAZ FUNKCIJA

LD1 SW0 or SW1

LD2 SW2 or SW2

LD3 (SW0 xor SW1) and (SW2 xor SW2)

library IEEE

use IEEE.std\_logic\_1164.all

entity THIS\_IS\_SPAAAAARTA is

port(

SW: in std\_logic\_vector(2 downto 0);

LD: out std\_logic\_vector(2 downto 0);

);

end THIS\_IS\_SPAAAAARTA;

architecture ARHITEKTURA of THIS\_IS\_SPAAAAARTA is

begin

LD(1)<=SW(0) or SW(1)

LD(2)<=SW(2) or SW(2)

LD(3)<=(SW(0) xor SW(1)) and (SW(2) xor SW(2))

end ARHITEKTURA;

ZADATAK 6.

U VHDL-u projektirati 16-bitni brojač (counter) sa asinkronim resetom. Brojač na ulazu ima signal širine 4 bita koji definira prodatak kojim se trenutna vrijednost brojača inkrementira. Brojač omogućuje brojanje naprijed i natrag koji se kontrolira pomoću priključka UP\_DOWN('1' - inkrementiranje, '0' - dekrementiranje). Isto tako, brojač na ulazu ima ENABLE priključak, koji omogućuje brojanje ukoliko je postaljen u '1', a inače neće doći do brojanja. Brojač ima i asinkroni reset koji sve izlazne signale postavlja u vrijednost '0'.

library IEEE

use IEEE.std\_logic\_1164.all

entity COUNTER is

port(

brojac: out std\_logic\_vector(15 downto 0);

dodaj: in std\_logic\_vector(3 downto 0);

UP\_DOWN: in std\_logic;

ENABLE: in std\_logic;

RESET: in std\_logic;

CLK: in std\_logic);

end COUNTER;

architecture ARHITEKTURA of COUNTER is

variable count := std\_logic\_vector(15 downto 0);

variable ulaz := std\_logic\_vector(3 downto 0);

ulaz := dodaj;

begin

process(RESET, CLK, ENABLE, UP\_DOWN)

begin

if RESET = '1' then

count := '0000 0000 0000 0000';

end if;

if CLK'event and CLK = '1' then

if ENABLE = '1':

if UP\_DOWN = '1' then

count = count + ulaz;

elif UP\_DOWN = '0' then

count = count - ulaz;

end if;

end if;

end if;

brojac <= count

end process;

end ARHITEKTURA;

ZADATAK 13.

U VHDL-u projektirati kontrolnu jedinicu za semafor pomoću stroja stanja kako je to pokazano na predavanjima (dva procesa). Ulaz u jedinicu je signal vremenskog vođenja, a izlazi (njih 3) se koriste za paljenje signalizacijskih svjetala (crvena, žuta, zelena). Stroj mijenja stanja tako da se dobije sljedeći redoslijed svjetala: 1. Crvena, 2. Crvena i Žuta, 3. Zelana, 3. Žuta, te tako iznova. Isto tako, sklop ima asinkroni reset, koji postavlja stanje u 1. Crvena.

use IEEE

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 ARHITEKTURA of SEMAFOR is

type STATE\_TYPE is (S\_CRVENO, S\_CRVENO\_ZUTO, S\_ZELENO, S\_ZUTO);

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';

sljedece <= S\_CRVENO\_ZUTO;

elif trenutno = S\_CRVENO\_ZUTO then

SVJETLO = '110';

sljedece <= S\_ZELENO;

elif trenutno = S\_ZELENO then

SVJETLO = '001';

sljedece <= S\_ZUTO;

elif trenutno = S\_ZUTO then

SVJETLO = '010'

sljedece <= S\_CRVENO

end if;

end if;

end process;

process(CLK, RESET)

begin

if RESET = '1' then

trenutno = S\_CRVENO;

elif CLK'event and CLK = '0' then

trenutno <= sljedece;

end if;

end process;

end ARHITEKTURA;

ZADATAK 14.

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

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

entity ISPITAJ is generic(n:integer:8)

port(

sig1 : in std\_logic\_vector(n-1 downto 0);

sig2 : in std\_logic\_vector(n-1 downto 0);

izlaz: out std\_logic;

);

end ISPITAJ;

architecture ARHITEKTURA of ISPITAJ is

begin

process(sig1, sig2)

variable temp : bit

temp = '0'

begin

for i in n'range loop

if(sig1(i) = sig2(i)) then

temp = '1';

else

temp = '0';

exit;

end if;

end loop;

izlaz<=temp;

end process;

end ARHITEKTURA;

ZADATAK 15.

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 ARHITEKTURA of NULE is

variable brojac : std\_logic(4 downto 0);

brojac:='00000'

begin

process(ulaz,izlaz)

begin

for i in 0 to 31 loop

if(ulaz[31-i]='0') then

brojac = brojac + 1

else

exit

end if

end loop;

izlaz<=brojac;

end process;

end ARHITEKTURA;

Zadatak 7. (15 bodova)

Na PicoBlaze procesor spojene su dvije ulazne vanjske jedinice (VJ1, port id: 0x20,

VJ2 , port id: 0x40) i jedna izlazna jedinica(VJ3, port id: 0x60), te vremenski sklop

koji je spojen na prekidnu liniju PicoBlaze procesora. Potrebno je rješiti sljedeći

zadatak: Na svaki prekid dobiven od prekidne jedinice treba pročitati podatak sa

ulaznih jedinica (podaci su u 8-bitnom 2'k formatu). Na prvi prekid čita se podatak

sa VJ1, a na sljedeći prekid sa VJ2. Na sljedeći prekid potrebno je poslati podatak

na VJ3, te postupak ponavljati beskonačno. Ukoliko je podatak primljen na VJ1 paran,

onda se podatak sa VJ2 šalje nepromijenjen na VJ3. Ako je podatak sa VJ1 neparan, onda

se na VJ3 šalje negirana vrijednost podatka primljenog na VJ2. Potrebno je napisati

sve procese u VHDL-u koji će obrađivati vanjske jedinice i prekidnu jedinicu, te program

za procesor, cjelokupni programski kod u VHDL-u za top-level entitet i arhitekturu te

nacrtat blok-shemu povezivanja komponenti.

ADRESS 000

CONSTANT VJ1, 20

CONSTANT VJ2, 40

CONSTANT VJ3, 60

LOAD s0, 00 ; Brojač prekida

ENABLE INTERRUPT

JUMP petlja

petlja: JUMP petlja

prekid\_prvi:

INPUT s1, VJ1; čitajmo sa VJ1

prekid\_drugi:

INPUT s2, VJ2; čitajmo sa VJ2

prekid\_treci:

AND s1, 01;

JUMP Z, paran

neparan:

SUB s2,1

XOR s2, FF

STORE s2, VJ3

RETURNI

paran:

STORE s2, VJ3

RETURNI

ADRESS 3FF

ADD s0, 01; povecaj brojac prekida za jedan

xor s0, 01

JUMP Z, prekid\_prvi

xor s0, 02

JUMP Z, prekid\_drugi

; za bilo koji prekid osim prvog i drugog ispisujemo! (nema provjere uvjeta)

JUMP prekid\_treci

architecture PONASANJE of SKLOP is

signal data\_vanjska1 : std\_logic\_vector(7 downto 0);

signal data\_vanjska2 : std\_logic\_vector(7 downto 0);

signal data\_vanjska3 : std\_logic\_vector(7 downto 0);

component vanjska\_ulaz is port( --struktura

data : in std\_logic\_vector(7 downto 0);

clk: in std\_logic\_vector

);

component vanjska\_izlaz is port(

data : out std\_logic\_vector(7 downto 0);

clk: in std\_logic\_vector

);

begin

prekidna: process(clk)

begin

if clk'event and clk = '1' then

if interrupt='1' then --desio se prekid

interrupt='1'

endif

end process prekidna;

vanjska\_jedinica\_1: vanjska\_ulaz

port map(

data => data\_vanjska\_1,

clk => clk

);

vanjska\_jedinica\_2: vanjska\_ulaz

port map(

data => data\_vanjska\_2,

clk => clk

);

vanjska\_jedinica\_3: vanjska\_izlaz

port map(

data => data\_vanjska\_3,

clk => clk

);

ulazne\_vj: process(clk)

begin

if clk'event and clk = '1' then

if port\_id = '00100000' then --ako je port 20

in\_port <= data\_vanjska\_1;

elif port\_id = '01000000' then --ako je port 40

in\_port <= data\_vanjska\_2;

end if;

end if;

end process ulazne\_vj;

izlazne\_vj: process(clk)

begin

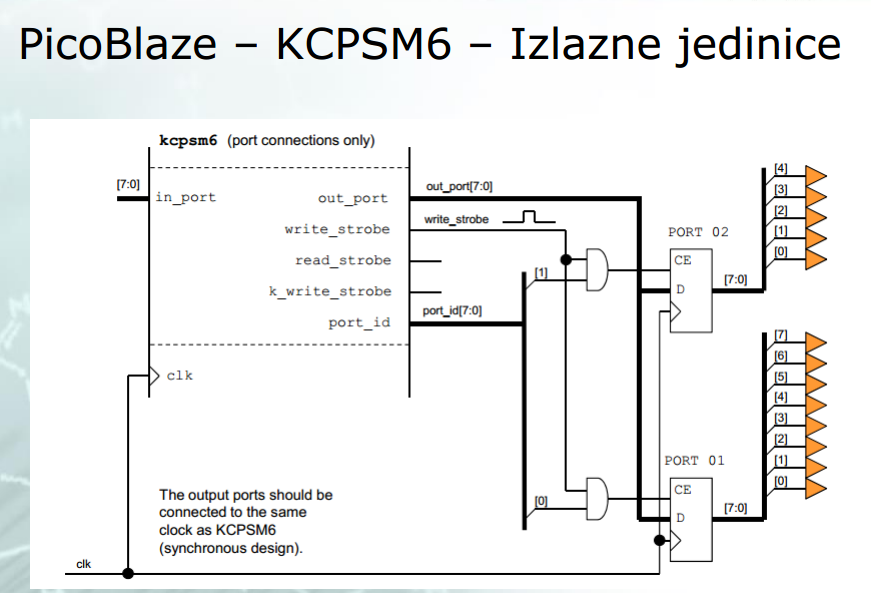
if clk'event and clk = '1' then

data\_vanjska\_3 <= out\_port;

end if;

end process izlazne\_vj;

end architecture PONASANJE;



output\_ports: process(clk)

begin

if clk'event and clk = '1' then

if write\_strobe = '1' then

if port\_id(0) = '1' then

led <= out\_port;

end if;

if port\_id(1) = '1' then

led\_n <= out\_port(0);

led\_e <= out\_port(1);

led\_s <= out\_port(2);

led\_w <= out\_port(3);

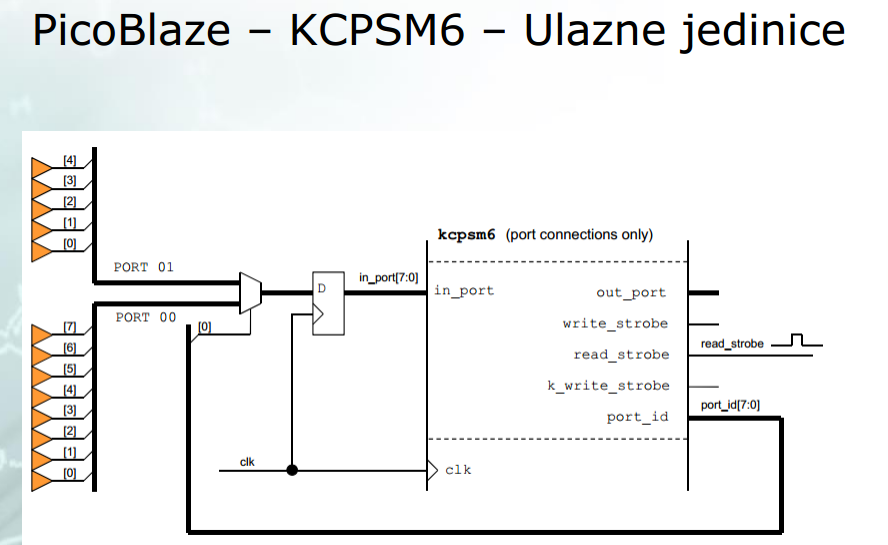
led\_c <= out\_port(4);

end if;

end if;

end if;

end process output\_ports;



input\_ports: process(clk)

begin

if clk'event and clk = '1' then

case port\_id(0) is

when '0' => in\_port <= dip\_switch;

when '1' => in\_port(0) <= push\_n;

in\_port(1) <= push\_e;

in\_port(2) <= push\_s;

in\_port(3) <= push\_w;

in\_port(4) <= push\_c;

when others => in\_port <= „XXXXXXXX“;

end case;

end if;

end process input\_ports;

prekidna: process(clk)

begin

if clk'event and clk = '1' then

if interrupt\_ack = '1' then

interrupt <= '0';

elif interrupt\_event\_1 = '1' then

interrupt <= '1';

else

interrupt <= interrupt

end if;

end prekidna

ulazne\_vj: process(clk)

begin

if clk'event and clk = '1' then

if port\_id = '00100000' (20) then

in\_port <= '10010110' --provjerit!

elif port\_id = '01000000' (40) then

in\_port <= '10010110' --provjerit!

end if;

end if;

end ulazne\_vj;

izlazna\_vj: process(clk)

begin

if clk'event and clk = '1' then

if write\_strobe = '1' then

if port\_id='0110000' then (60)

--piši na VJ3

out\_port <= in\_port;

end if;

end if;

end if;

end izlazna\_vj;

library IEEE;

use IEEE.std\_logic\_1164.all;

entity ENTITET is

port(

clk: in std\_logic;

ulaz1: in integer;

ulaz2: in integer;

)

output\_ports: process(clk)

begin

if clk'event and clk='1' then

if write\_strobe = '1' then

if port\_id = '20' then

VJ3 <= out\_port;

end if;

end if;

end if;

end process output\_ports;