Završni ispit URS – skupljena sva ispitna pitanja s odgovorima
By: Čule

- **Nabrojite osnovne porodice PLD sklopova, navedite njihove nazive?**

PROM (Programmable ROM), PAL (Programmable Array Logic), PLA (Programmable Logic Array), GAL (Generic Array Logic), CPLD (Complex Programmable Logic Device), FPGA (Field Programmable Gate Array).

- **PROM - koje oblike logicke funkcije je moguce (s koliko ulaza i izlaza) implementirati i zašto?**

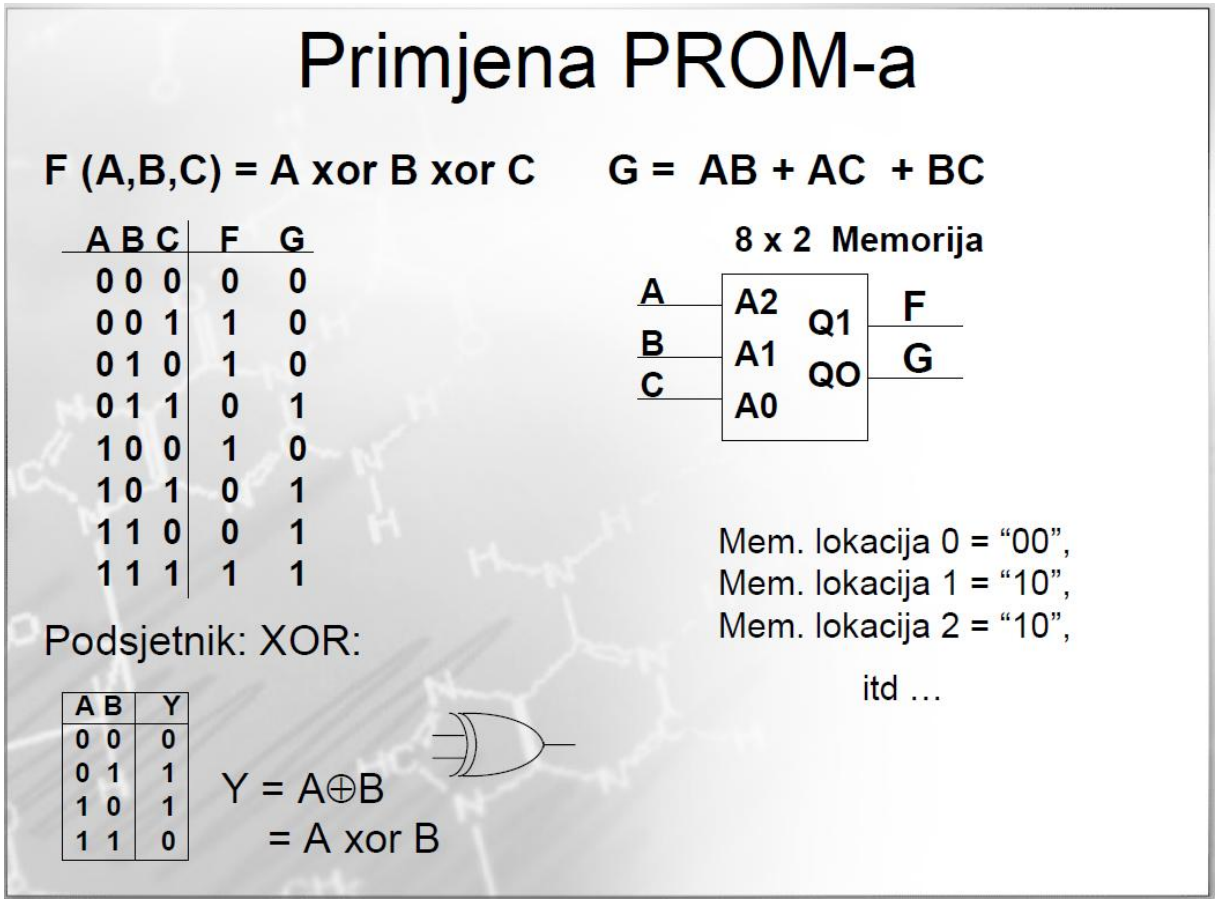
Korištenjem 2k x n PROM-a, mozemo implementirati BILO KOJU logicku funkciju sa k ulaza i n izlaza zato što k-to-2k decoder generira svih 2k mintermova a onda svaki od OR-ova implementira sume mintermova koje predstavljaju trazene logicke funkcije ovisne o k ulaza. Svi mintermovi mogu se implementirati.

- **Koja je najbitnija razlika između GAL i PAL sklopova?**

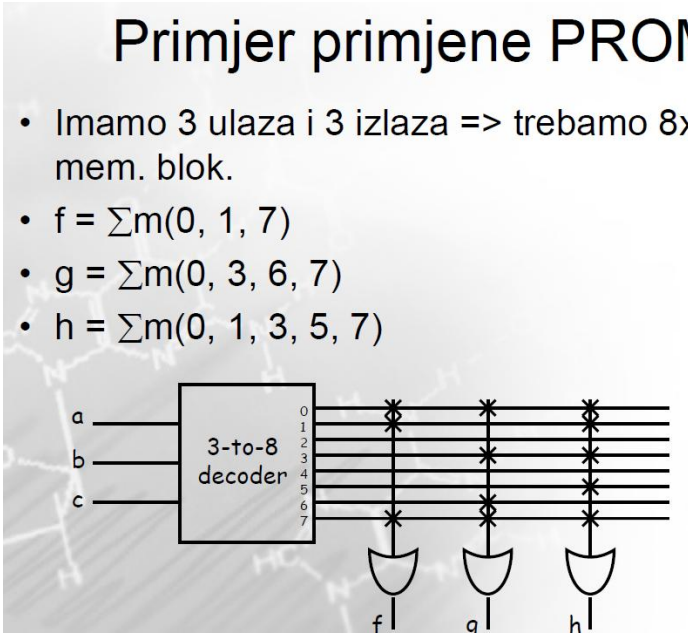
GAL i PAL imaju potpuno isti princip rada. GAL – višestruko programiranje; PAL – samo jednom.

- **Za funkcije:**
  F(A,B,C) = (A xor B xor C) i
  G(A,B,C)= ((A and B) or (A and C) or (B and C)),
  odrediti veličinu memorije i njezin cjelokupni sadržaj.

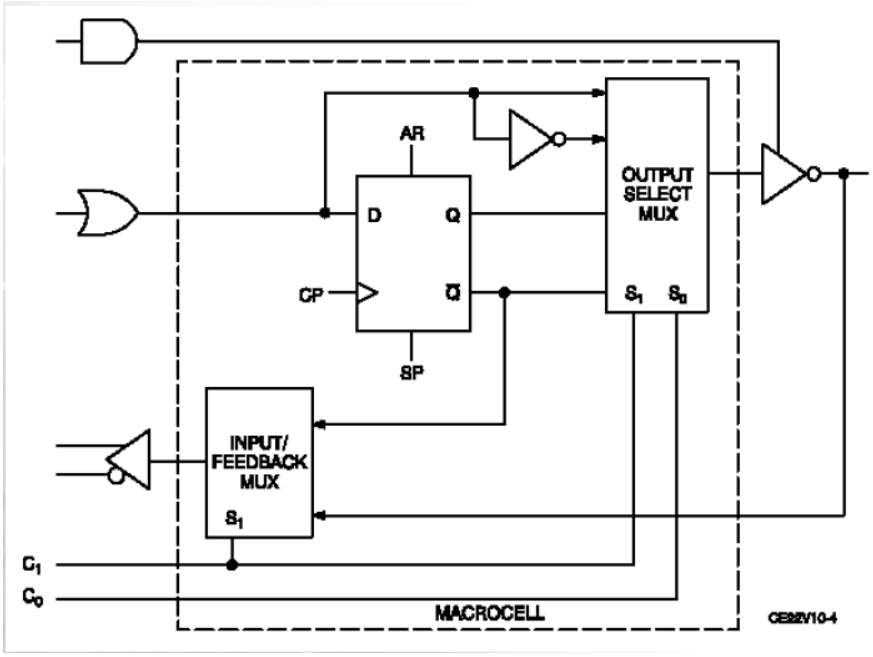
Veličina memorije je 2^3*2=16 sadržaj se određuje raspisivanjem tablica istinitosti za zadane funkcije.



- **Zadane su fie - (f = Em(0,1,3), g = Em(1,2,4,7), h = Em(0,2,3,5)) Implementirati pomocu PROM-a sa adr. dekoderom (fiksnim) i poljem OR sklopova. (takav zadatak)**



- **Nacrtati osnovnu shemu makroćelije u PAL sklopu.**



- **Čemu služi povratna veza u PAL sklopvima?**

Spajanjem izlazne linije na ulaze AND-a stvara se povratna veza. Služi za implementaciju složenih logičkih funkcija

- **Kako se izvode programibilni prekidaci u PLD sklopovima? Objasni osnovne karakteristike!**

Koriste se osirgurači (fuse), tranzistori s plivajućom elektrodom (EPROM, EEPROM), SRAM, Anti-fuse (jednostavna modifikacija CMOS tehnologije, PLICE)

	Ponovo programibilni	Naponski ovisni	Tehnologija
Osigurači (Fuse)	Ne	Ne	Bipolar
EPROM	DA (izvan uređaja)	Ne	UVCMOS
EEPROM	DA (izvan i u uređaju)	Ne	EECMOS
SRAM	DA (u uređaju)	Da	CMOS
Antifuse	Ne	Ne	CMOS+

- **Skicirati i objasniti osn. arhitekturu SPARTAN 3 sklopova (od kojih tipova el. se sastoje i njihova osn. svojstva i fie.)**

Arhitektura spartan 3 sastoji se od 5 osnovnih programibilnih jedinica:

CLB blokovi – sadrže RAM zasnovane funkcijske tablice (LUT) za izvedbu logickih i memorijskih elemenata.

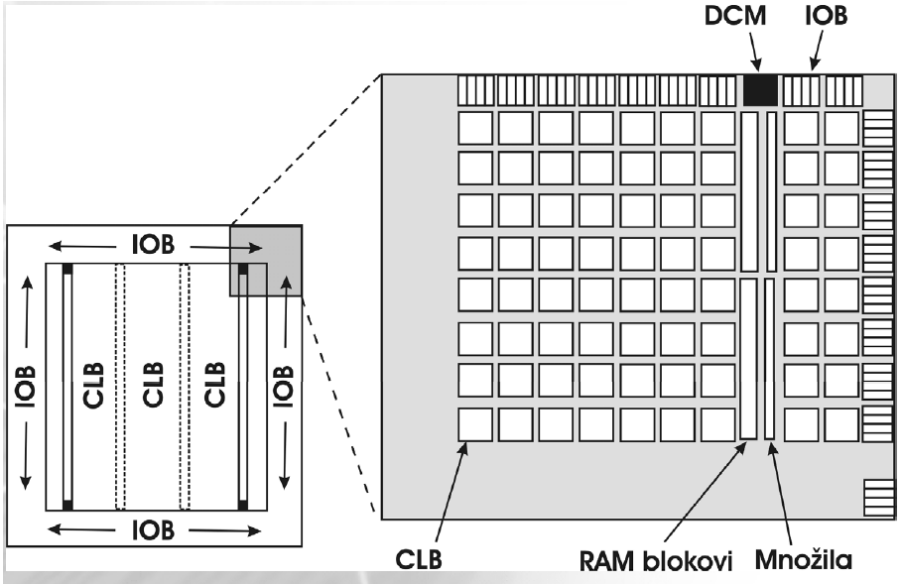
Ulazno izlazni blokovi - kontroliraju protok podataka izmedu ulazno izlaznih pinova i unutarnje logike. Svaki IOB podržava dvosmjerni protok podataka i stanje visoke impedancije. Dostupne su 24 razlicite naponske razine ukljucujuci 6 diferencijalnih standarda. Digitalno kontrolirana impedancija omogucava automatsko terminiranje signala cime uvelike pojednostavnjuje izradu sklopa. – input output blocks

Memorijski blokovi – omogucuju pohranu podataka u 18Kb memorijama

Blokovi za množenje imaju 18 bitne ulaze i racunaju 36 bitni produkt.

Blokovi za digitalnu kontrolu perioda signala vremenskog vodenja sinkroniziraju signal vremenskog vodenja po cijelom sklopu, omogucavaju množenje, dijeljenje i fazni pomak perioda signala vremenskog vodenja. (DCM digital clock manager)

Prsten IOB okružuje pravilno polje CLB-ova. Ovisno o modelu Spartan-3 sklopovi mogu imati jedan stupac, dva stupca ili cetiri stupca RAM memorije, svaki stupac sacinjen je od nekoliko 18Kb RAM blokova, svaki blok memorije povezan je sa blokom za množenje, a blokovi za digitalnu kontrolu perioda signala vremenskog vodenja nalaze se na kraju stupca bloka memorije.



- **Na koje se sve načine može programirati Spartan FPGA sklop? Koji tip programabilnih prekičada se koristi u tim sklopovima?**

Spartan FPGA sklop koristi SRAM memorije kao programibilni prekidač. SRAM memorije ne pamte konfiguraciju bez napajanja. Programiranje Spartan-3 FPGA sklopova obavlja se učitavanjem konfiguracijskih podataka u statičku memoriju koja kontrolira se funkcije elemenata i njihovu povezanost. Prilikom uključivanja FPGA sklopa konfiguracija pohranjena u PROM-u ili nekom drugom mediju pohranjuje se u FPGA sklop. Konfiguracija se može učitati u FPGA sklop na pet različitih načina: dva paralelna načina, dva serijska načina i preko JTAG priključaka.

- **Cemu služi grana visoke impendancije u I/O sklopu Spartan 3 uredaja?**

Grana visoke impedancije omogućava automatsko terminiranje signala. Grana je izvedena uz pomoć digitalno kontrolirane imepdancije koja omogućava dva načina terminiranja signala: paralelno i serijsko terminiranje.

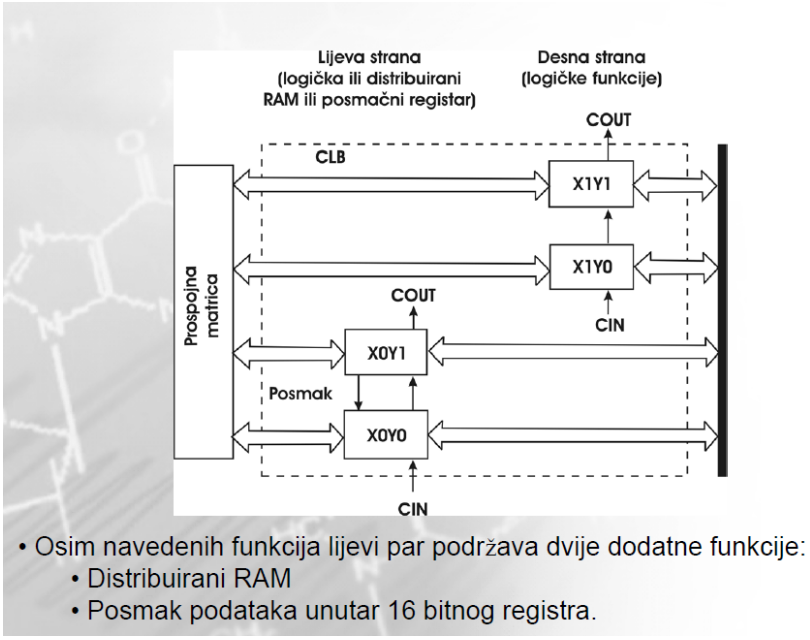
- **Cemu služi DCM blok u Spartan 3 uredaju?**

Blokovi za digitalnu kontrolu perioda signala vremenskog vođenja (DCM) sinkroniziraju signal vremenskog vođenja po cijelom sklopu. Koristi DLL (delay locked loop) povratnu vezu s kašnjenjem. Služi za: stabilizaciju signala vremenskog vođenja, generiranje različitih frekvencija (množenje i dijeljenje perioda signala vremenskog vođenja), fazni pomak signala vremenskog vođenja.

- **Objasnite i skicirati strukturu CLB blokova unutar Spartan 3 uređaja.**

Konfigurabilni logički blokovi (eng. Configurable Logic Blocks – CLB) predstavljaju glavni dio sklopa zadužen za izvedbu slijednih i kombinacijskih funkcija. Svaki CLB sastoji se od 4 međusobno povezana dijela kako je prikazano na slici koja slijedi. Dijelovi su grupirani u parove, a svaki par organiziran je u stupce sa nezavisnim lancem za prijenos (eng. Carry look ahead).
Sva četiri dijela sadrže slijedeće elemente:

- Dva generatora funkcijske logike (lut)
- Dva memorijska elementa
- Multipleksore
- Prijenosni lanca sa dodatnim logičkim sklopovima



- **Što su LUT blokovi, kako se koriste i cemu služe?**

Look-Up Tables su generatori funkcijske logike. Svaki od dva LUT-a unutar jednog dijela ima 4 logička ulaza i 1 izlaz. To nam omogućava izvođenje logičkih funkcija sa 4 varijable. Nadalje, korištenjem multipleksora LUT-ovi se mogu kombinirati unutar samog CLB ili kombinirati sa drugim CLB-ovima čime se mogu dobiti složene logičke funkcije. LUT-ovi se također mogu koristiti kao ROM memorije, distribuirani RAM ili kao 16 bitni posmični registar.

- **KOji tip programibilnih prekidaca se koristi u SPARTAN 3 sklopovima?**

Koristi se SRAM memorija koja ne pamti konfiguraciju bez napajanja. Izveden je u CMOS tehnologiji.

- **Navesti osnovne vrste poveznica u Spartan 3 arhitekturi te njihova osnovna svojstva.**

long lines – 24 signala, rasprostiru se horizontalno i vertikalno, spajaju svaki šesti blok, zbog malog kapaciteta idealni za prenošenje brzih signala.
hex lines – 8 signala, horizontalno i vertikalno, spajaju svaki treci blok, samo jedan blok na kraju signala može upravljati signalima
double lines – 8 signala, svaki drugi blok povezuje vertikalno, horizontalno i dijagonalno.
direct connection - povezuje blok s susjednim blokovima u svim smjerovima. Povezuje blokove s hex, long i double linesima.

- **Što je entitet a što opis arhitekture u VHDL-u?**

Entitet opisuje sučelje sklopa, a arhitektura tijelo i ponašanje sklopa tj funkcionalnost sklopa. Entitet opisuje dakle kako komponenta izgleda prema van dok arhitektura predstavlja unutarnji dizajn i opisuje njeno ponašanje.

- **U VHDLu definirati signal s tipa std_logic koji će nakon globalnog reseta imati vrijednost ‘1’.**

```
signal s : std_logic :='1';
```

- **Razlika izmedu glob. i lok. reseta? Napiši VHDL kod koji za signal tipa std_logic definira stanje '1' u glob. resetu i stanje '0' u lokalnom.**

Globalnim resetom postavljamo inicijalnu vrijednost sklopovlja na izlaz. Pocetna vrijednost se postavlja inicijalno za komponentu i neovisna je o lokalnom resetu. Lokalni reset se koristi tako da napišemo proces koji ce napraviti inicijalizaciju sklopa.

```
signal s : std_logic :='1';

Process (ckl,rst)
begin
     If rst='1' then
          s <='0';
     End if;
End process;
```

- ```
  ...
  signal c:std_logic := '0';
  ...
  process(a)
  begin
  c <= a;
  b <= c;
  end process;
  ```
  Kolika je vrijednost signala b nakon izvođenja gornjeg procesa pri cemu se dogodila promjena (event) na signalu a s vrijednosti '0' na '1' te opet s '1' na '0'?
  process(a)
  variable c:std_logic := '0';
  ```

begin
c := a;
b <= c;
end process;
Kolika je vrijednost signala b nakon Sto se dogodila promjena (event) na signalu a sa vrijednosti '0' na '1' te ponovo sa '1' na '0' . Pojasniti!

Razlika je u tome što se varijabli pridjeli odmah vrijednost, a signalu tek nakon izlaska iz procesa. Tako se u prvom zadatku kod rastućeg brida signala a signal c ne promijeni odmah u prvoj naredbi (c<=a) tj. ostane 0.Signalu b se pridruži stara vrijednost signala c, tj. 0. Kod padajućeg brida signala a ponavlja se ista stvar, c-u se ne pridijeli odmah vrijednost 0 nego ostane u 1 pa je b=1
Kod drugog zadatka ide onako kako si rekao..prvo ide sve u 1 pa onda sve u 0.

- **Razlika izmedu signala i varijabli u VHDL-u.**

Signal predstavlja izvod sklopa i tako se i koristi, dok varijablu rabimo radi pridržavanja parcijalnih rezultata, kao indekse u petljama, a mogu i predstavljati vodove sklopa. Signale definiramo u dijelu za definiciju arhitekture i moguce ih je koristiti bilo gdje u arhitekturi. Varijable se definiraju unutar procesa ili podprograma i mogu se koristiti samo unutar navedenog procesa ili potprograma. Signalu se vrijednost pridružuje <= a varijabli :=.

- **Koja je razlika između kombinacijske i procesne logike i na koje se dvije vrste logike može podijeliti procesna logika?**

Kombinacijska logika – jednoznačna pridruživanja, redoslijed logike nije bitan. Kod procesne logike pridruživanje se radi određenim redosljedom. Zadnja pridružena vrijednost unutar procesne logike poništava prethodnu pridruženu vrijednost. Procesna logika može biti kombinacijska i slijedna. Kombinacijska u kodu nema process ova druga ima.

- **Objasnite memorijski sustav Picoblaze procesora (programskog i podatkovnog)? S obzirom na memoriju u koju aritekturu pripada?**

Dvije odvojene memorije: programska (1024 18 bitne memorijske lokacije sa kojih se samo čita) i podatkovna memorije – scrath pad memorija 64 8 bitne memorijske lokacije za opću namjenu + 16 8 bitnih registara opće namjene. Harvard arhitektura

- **Korištenjem kombinacijske logike napišite VHDL kod (entitet i arhitekturu) za izvedbu multipleksora 8 na 1, pri čemu je defaultna vrijednost na izlazu ona sa ulaza(0)?**

```
Entity bhv is
Port(in_data : in std_logic_vector(7 downto 0);
     Selector : in std_logic_vector(2 downto 0);
     T : out std_logic_vector
);
End entity bhv;
architecture bhv of select__bhv is
begin
     with selector select
          T <= In_data(1) when "001",
               In_data(2) when “010”,
               …
               In_data(0) when others;
end bhv;
```

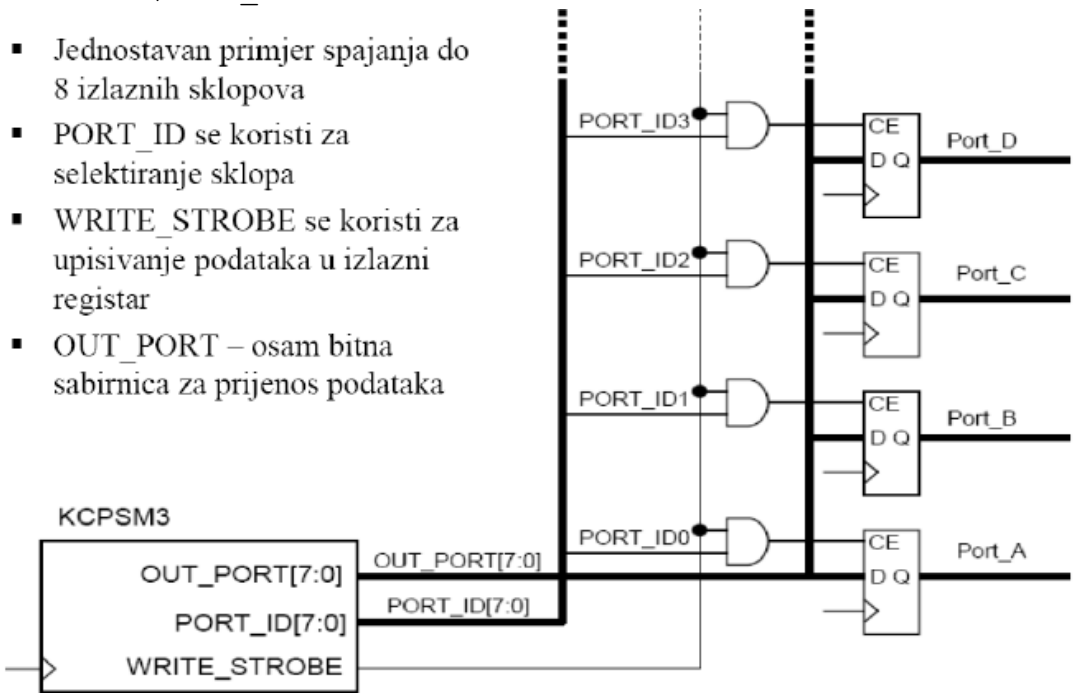
- **Napišite dio VHDL koda (potrebno je napisati samo proces) koji služi za spajanje izlaznih vanjskih jedinica na PicoBlaze? Objasnite kako radi? Napisati primjer korištenja vanjske jedinice u assembleru. Nacrtati shematski spajanje na KCPSM3 procesor.**

```
output_ports: process (clk)
      begin
      if clk'event and clk='1' then
         if write_strobe='1' then
            -- Write to LEDs at address 80 hex.
            if port_id(7)='1' then
               led <= out_port;
            end if;
               end if;
      end if;
end process output ports;
```

Primjer korištenja vanjske jedinice u assembleru:

```
CONSTANT LED_PORT, 40
…
LOAD s0,08
OUTPUT s0, LED_PORT
```

- Jednostavan primjer spajanja do 8 izlaznih sklopova
- PORT_ID se koristi za selektiranje sklopa
- WRITE_STROBE se koristi za upisivanje podataka u izlazni registar
- OUT_PORT – osam bitna sabirnica za prijenos podataka



- **Objasnite koje je razlika između Diskretnih, Hard i Soft Core procesora? Koje su prednosti i nedostatci svakih od njih?**

Diskretni procesori

- diskretni ili „off the shelf“
- tradicionalan pristup
- veliki izbor različitih proizvođača
- veliki izbor različite funkcionalnosti
- izvedeni kao ASIC sklopovi koji osim samoga procesora uglavnom sadrže i periferije
- ispravan odabir odgovarajućeg procesora može biti vrlo zahtjevan zadatak

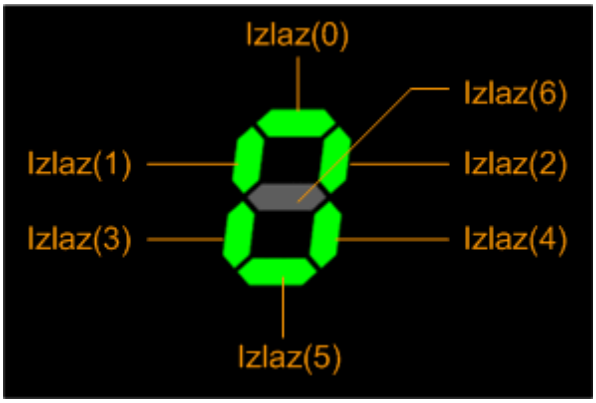
Hard Core procesori:

- posebni namjenski sklopovi unutar komponente (uglavnom FPGA)
- s obzirom na izvedbu mogu raditi na velikim brzinama
- prednost im je što postoje u okruženju gdje se njegova okolina može jednostavno prilagoditi potrebama dizajna
- ne omogućavaju prilagodbu procesora dizajnu
- samo pojedini članovi porodica FPGA sklopova imaju takvu mogućnost
- vrlo ograničen broj proizvođača i FPGA sklopova

Soft Core procesori:

- u potpunosti izveden u logici
- zbog toga ne može raditi na maksimalnim brzinama samog sklopa
- uglavnom višestruko sporiji od diskretnih i Hard Core procesora
- uglavnom pogodni za sustave gdje brzina izvođenja algoritma nije presudna
- kontrola GPIO
- iako nije ograničena na samo takvu primjenu

- **Isprojektirati u VHDLu komponentu (entitet i arihitekturu) za 7-segmentni pokaznik korištenjem kombinacijske procesne logike. Ulaz je signal tipa int koji prima vrijednosti od 0 do 9 a izlaz je std_logic_vector veličine 7 koji predstavlja kod za 7-segmentni pokaznik kako je prikazano na slici:**




```
Entity SevenSegDecoder is
Port(in_data : in integer;
     Out_data: out std_logic_vector(6 downto 0)
     );
End entity SevenSegDecoder;

architecture Beh of SevenSegDecoder is
begin
     process(in_data)
     begin
          case in_data is
               when 1 =>
                    out_data <= "0010100";
               when 2 =>
                    out_data <= "1011011";
               when 3 =>
                    out_data <= "1010111";
               when 4 =>
                    out_data <= "0100101";
               when 5 =>
                    out_data <= "1100111";
               when 6 =>
                    out_data <= "1101111";
               when 7 =>
                    out_data <= "1010100";
               when others =>
                    out_data <= "0000000";
          end case;
     end process;
end Beh;
```

- **Za PicoBlaze procesor napisati sljedeću prekidni potprogram: Na svaki zahtjev za prekid potrebno je očitati stanje vanjske jedinice koja je spojena na IN_PORT. Na dobivenom podatku ispituje se parnost: Ukoliko je on paran, potrebno je povećati brojač parnih podataka u ScratchPad memoriji na adresi hex(10), a ukoliko je neparan, povećati brojač neparnih podataka koji je u ScratchPad memoriji na adresi hex(11).**

```
            ADDRESS 000
            CONSTANT IN_PORT, 00
            ENABLE INTERRUPT
            ; brojac parnih i neparnih
            LOAD s1, 00
            LOAD s2, 00
CEKAJ       JUMP CEKAJ
PREKID
            ; ucitaj podatak
            INPUT s0, IN_PORT
            ;ispitaj parnost
            AND s0, 01
            COMPARE s0, 00
            JUMP Z, PARAN
NEPARAN
            ADD s1, 01
            STORE s1, (10)
            RETURNI
PARAN
            ADD s2, 01
            STORE s2, (11)
            RETURNI


            ; prekidna adresa
            ADDRESS 3FF
            JUMP PREKID
```

- **U VHDL-u projektirat 16 bitni brojac (COUNTER) s asinkronim resetom. Brojac omogucuje brojanje naprijed i natrag koji se kontrolira pomocu prikljucka UP_DOWN ('1' - inkrementiranje, '0' - dekrementiranje)**

```
entity counter is port (
      CLK: in STD_LOGIC;
      RESET: in STD_LOGIC;
      UP_DOWN: in STD_LOGIC;
      DO: out STD_LOGIC_VECTOR(15 downto 0)
      );
end counter;

architecture broji of counter is
      begin
            process(CLK, RESET)
                  variable COUNT: STD_LOGIC_VECTOR(15 downto 0);
      begin
            if RESET = ‘1’ then
                  COUNT := “0000000000000000”;
            elsif CLK = ‘1’ and CLK’event then
                  if UP_DOWN = ‘1’ then
                        COUNT := COUNT + ''0000000000000001'';
                  else
                        COUNT := COUNT – ''0000000000000001'';
                  end if;
            end if;
            DO<=COUNT;
end process;
end broji;
```

- **Zadana je sljedeca tablica (*) ulazno izlaznih vr. za enkoderski sklop. Potrebno ga je implementirat (entitet i arh.) u VHDL-u i to pomocu kombinacijske logike**

  *** ULAZ IZLAZ**

  "000" "00000001"

  … …

  "111" "10000000"

```
entity dek3na8 is port (
     D_IN: in STD_LOGIC_VECTOR (2 downto 0);
     D_OUT: out STD_LOGIC_VECTOR (7 downto 0)
);
end dek3na8;

architecture dekoder of dek3na8 is
     begin
                with D_IN select
                D_OUT <="00000001" when „000“,
                      <="00000010" when „001“,
                …
                      <=NULL when others;

end dekoder;
```

- **Na PicoBlaze procesor spojena je 1 ulazna vanjska (port id 0x80) jedinica i 2 izl. jedinice s port idovima 0x60 i 0x40, te vremenski sklop koji je spojen na prekidnu liniju PicoBlaze procesora. NA svaki prekid dobiven od prekidne jedinice treba procitat podatak s ulazne jedinice i ispitat njegov paritet (br. binarnih '1'). Ukoliko je paran -> pod. se šalje na vanj. jedinicu sa idom 0x60, u suprotnom – na 0x40. Potrebno je napisat program u VHDL-u koji ce obradivat vanj. i prekidnu jedinicu, te prog. za procesor. Ovaj program odvija se beskonacno.**

```
            ADDRESS 000
            CONSTANT PORT_ID1, 80 ;ulazna vj
            CONSTANT PORT_ID2, 60
            CONSTANT PORT_ID3, 40

            ENABLE INTERRUPT
CEKAJ       JUMP CEKAJ
PREKID

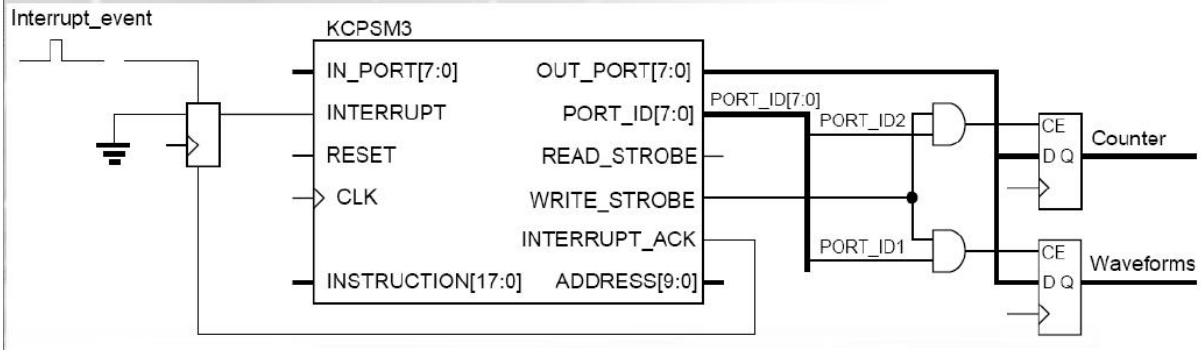
            INPUT s0, PORT_ID1
            TEST s0, FF
            JUMP C, NEPARAN
PARAN

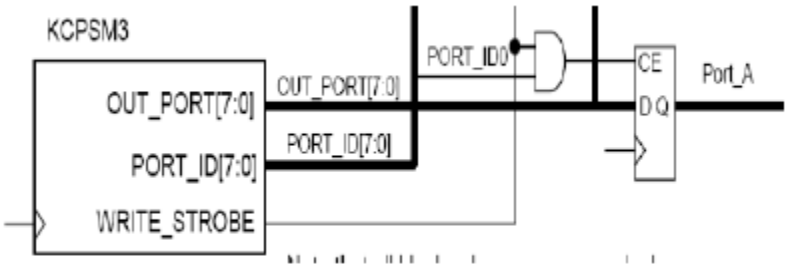
            OUTPUT s0, PORT_ID2
            RETURNI
NEPARAN
            OUTPUT s0, PORT_ID3
            RETURNI

; prekidna adresa
            ADDRESS 3FF
            JUMP PREKID
```

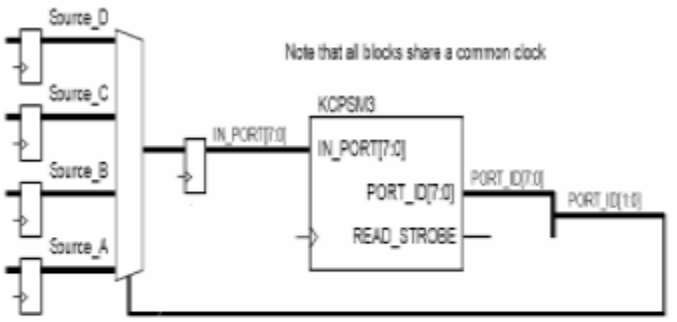
Prekidni sustav

```
interrupt_control: proces(clk)
begin
   if clk'event and clk = '1' then
        if interrupt_ack = '1' then
               interrupt <= '0';
        elsif interrupt_event_1 = '1' then
               interrupt <= '1';
        else
               interrupt <=interrupt;
        end if;
End process interrupt_control;
```






```
izlazne_vj: process(clk)
begin
      if clk'event and clk='1' then
            if write_strobe='1' then
                  if port_id="01100000" then
                        out_port <= in_port;
                  elsif port_id="01000000"
                        out_port <= in_port;
            end if;
      end if;
end izlazne_vj;
```




```
ulazna_vj: process(clk)
begin
        if clk'event and clk='1' then
                if port_id =”10000000” then
                        -- read data at address 80 hex
                        in_port <= ”10010110”;
                else
                        in_port <= "XXXXXXXX";

                end if;
        end if;
end ulazna_vj;
```

- **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 vodenja a izlazi (njih 3) se koriste za paljenje signalizacijskih svjetala (crvena, Zuta, zelena). Stroj mjenja stanja tako da se dobije sljedeii redosljed svjetala: 1. Crvena, 2. Crvena i Luta,3. Zelena,4.Luta, te tako iznova.Isto tako, sklop ima asinkroni reset, koji postavlja stanje u 1. Crvena.**

```
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) -- "100" ->
svijetli crveno, "110" -> svijetli crveno i zuto, "001" ->
svijetli zeleno, "010" -> svijetli zuto
);
end SEMAFOR;

architecture ARHITEKTURA of SEMAFOR is

type STATE_TYPE is (S_CRVENO, S_CRVENO_ZUTO, S_ZELENO, S_ZUTO);
signal CurS : STATE_TYPE := S_CRVENO;
signal NextS : STATE_TYPE := S_CRVENO;

begin


process(CLK, CurS)
begin
if (CLK'event and CLK='1') then
   case CurS is
         when S_CRVENO =>
               SVJETLO <= "100";
               NextS <= S_CRVENO_ZUTO;
         when S_CRVENO_ZUTO =>
               SVJETLO <= "110";
               NextS <= S_ZELENO;
         when S_ZELENO =>
               SVJETLO <= "001";
               NextS <= S_ZUTO;
         when S_ZUTO =>
               SVJETLO <= "010";
               NextS <= S_CRVENO;
   end case;
end if;
end process;
-----------------------------------------------------------------
-
```

```
   process(CLK, RESET)
   begin
      if (RESET = '1') then
            CurS <= S_CRVENO;
      elsif (CLK'event and CLK = '0') then
            CurS <= NextS;
      end if;
   end process;
   ------------------------------------------------------------------
   end ARHITEKTURA;
```

- **U VHDL-u je potrebno projektirati komponentu koja ispituje jednakost dva ulazna signala bit po bit. Signali su tipa std_logic velidine koja je definirana pomoiu generic kljudne rijedi 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 jednakost is generic (n : integer := 8);
port (
     A,B : in std_logic_vector (n-1 downto 0);
     Y : out std_logic
);
end jednakost;

architecture bhv of jednakost is
begin
process(A,B)
     variable temp: bit
     temp='0';

begin
     for i in n'range loop
          if(A(i)=B(i)) then
               temp:='1';
          else
               temp:='0';
               exit;
          end if;
     end loop;
Y<=temp;
end process;
end bhv;
```

- **U VHDLu projektirati komponentu koja računa broj vodećih nula u ulaznom signalu tipa std_logic veličine 32bita.Izlazni signal je tipa std_logic veličine 5 bitova.**

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all
entity vod_nul is port (
      IN : in std_logic_vector (31 downto 0);
      OUT : out std_logic_vector(4 downot 0);
);
end vod_nul;

architecture bhv of vod_nul is
begin
proces(IN)
      variable brojn: std_logic_vector (4 downto 0);
      brojn:="00000";

      begin
            for i in 0 to 31 loop
                  if (IN[31-i]='0') then
                        brojn:=brojn+1;
                  elseif(IN[31-i]) then
                        exit loop;
                  end if;
            end loop;
OUT<= brojn;
end process;
end bhv;
```

Zadatak 13 (4 boda)

Na PicoBlaze procesor spojena je ulazna vanjska jedinica VJ1 (PORT ID=0x80) i izlazna vanjska jedinica VJ2 (PORT ID=0x40), te LCD jedinica sa predavanja (PORT ID = 0x60). Potrebno je isprojektirati sustav (PicoBlaze asemblerski kod, procesi u VHDL-u i blok shema spajanja KCPSM3 procesora sa jedinicama) koji će rješiti sljedeći problem: Sa vanjske jedinice VJ1 se čitaju 8 bitni podatci u formatu 2'k. Ukoliko je primljeni 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. Za ispis na LCD trebat će vam sljedeće konstante:

```
CONSTANT chr_P, 50 ;znak P
CONSTANT chr_O, 4F ;znak O
CONSTANT chr_Z, 5A ;znak Z
CONSTANT chr_N, 4E ;znak N
CONSTANT chr_E, 45 ;znak E
CONSTANT chr_G, 47 ;znak G
CONSTANT clrdisp, 01 ;ciscenje LCDa
```

```
          ADDRESS 000
          CONSTANT VJ1, 80
          CONSTANT VJ2, 40
          CONSTANT LCD, 60
          CONSTANT chr_P, 50 ;znak P
          CONSTANT chr_O, 4F ; znak O
          CONSTANT clnr_Z, 5A ;znak Z
          CONSTANT chr_N, 4E ; znak N
          CONSTANT chr_E, 45 ; znak E
          CONSTANT chr_G, 47 ;znak G
          CONSTANT clrdisp, 01

          INTERRUPT ENABLE

PREKID:
          INPUT S0, VJ1
          LOAD S1, 80
          AND S1, S0 --ako je 10000000 rezultat znaci da je prvi
bit 1 odnosno neg vrijednost, a ak je sve 0 znaci poz
          JUMP Z, POZ

NEG:
          LOAD S2, clrdisp --brisi sto je prije pisalo
          OUTPUT S2, LCD
          LOAD S2, chr_N --pisi nove stvari
          OUTPUT S2, LCD
          LOAD S2, chr_E
          OUTPUT S2, LCD
          LOAD S2, chr_G
          OUTPUT S2, LCD
```

```
          SUB S0, 1 --pretvori u poz vrijednost 2'k ... oduzme se 1
i invertira sve bitove s xor
          XOR S0, FF
          OUTPUT S0, VJ2 --upisi na vj2 tu aps vrijednost
          RETURNI

POZ:
          LOAD S2, clrdisp
          OUTPUT S2, LCD
          LOAD S2, chr_P
          OUTPUT S2, LCD
          LOAD S2, chr_O
          OUTPUT S2, LCD
          LOAD S2, chr_Z
          OUTPUT S2, LCD

          OUTPUT S0, VJ2 -- za poz vrijednost nije potrebno
invertirat ni pretvarat
          RETURNI

ADDRESS 3FF
JUMP PREKID
```