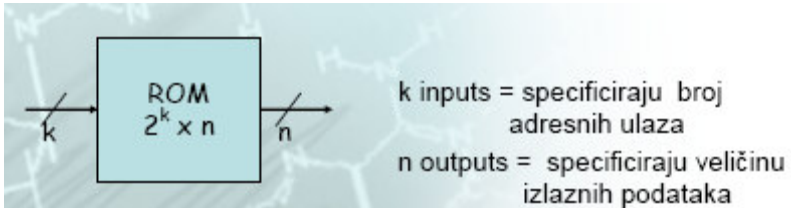
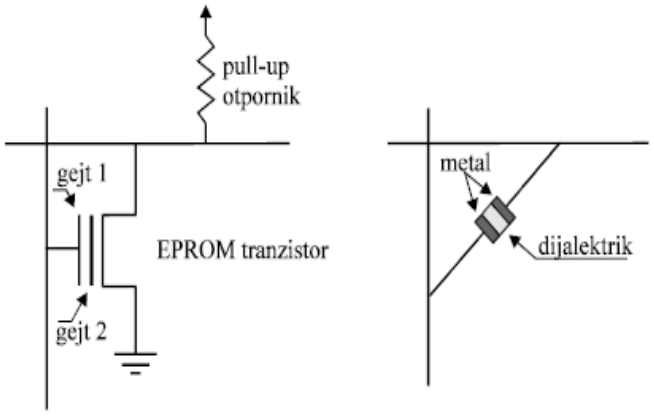
1. PROM - koje oblike logičke funkcije je moguće (s koliko ulaza i izlaza) implementirati i zašto?

Korištenjem 2k x n PROM-a, mozemo implementirati BILO KOJU logičku 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.



2. Kako se izvode programibilni prekidači u PLD sklopovima?
Objasni osnovne karakteristike!

Izvode se kao osigurači (fuse) EPROM, EEPROM, SRAM, Antifuse.
Kod EPROMa / EEPROMa podatci se upisuju tako da se na upravljačku elektrodu dovede impuls napona od 20V i duga trajanja (nekoliko ms) koji uzrokuje nabijanje lebdeće elektorde. Lebdeća elektroda je okružena izolatorom pa naboj traje i mnogo godina nakon prestanka djelovanja impulsa napona. Antifuse prekidači nastali su kao odgovor na drugačiji način korištenja prekidača od standardnih osigurača (fuse). Dakle, umjesto topljenja metalne veze prolaskom struje koristimo taj antifuse prekidač koji predstavlja jednostavnu modifikaciju CMOS tehnologije.



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

Budući da zadane logičke funkcije u svojim sumama minterma obuhvaćaju minterme sve do 7-og; zaključujemo kako će nam trebati dekoder koji će imati tri ulaza tako da za svaku kombinaciju tih ulaza mi dobivamo aktivan jedan od 8

mintermova koji predstavljaju izlaze iz dekodera. Te mintermove jednostavno "pozdrajamo" OR vratima tako da ostvaruju zadane funkcije.

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

CLB blokovi – sadrže RAM zasnovane funkcijske tablice (LUT) za izvedbu logičkih i memorijskih elemenata.
Ulazno izlazni blokovi - kontroliraju protok podataka između ulazno izlaznih pinova i unutarnje logike. Svaki IOB podržava dvosmjerni protok podataka i stanje visoke impedancije. Dostupne su 24 različite naponske razine uključujući 6 diferencijalnih standarda. Digitalno kontrolirana impedancija omogućava automatsko terminiranje signala čime uvelike pojednostavnjuje izradu sklopa.
Memorijski blokovi – omogućuju pohranu podataka u 18Kb memorijama
Blokovi za množenje imaju 18 bitne ulaze i računaju 36 bitni produkt.
Blokovi za digitalnu kontrolu perioda signala vremenskog vođenja sinkroniziraju signal vremenskog vođenja po cijelom sklopu, omogućavaju množenje, dijeljenje i fazni pomak perioda signala vremenskog vođenja.

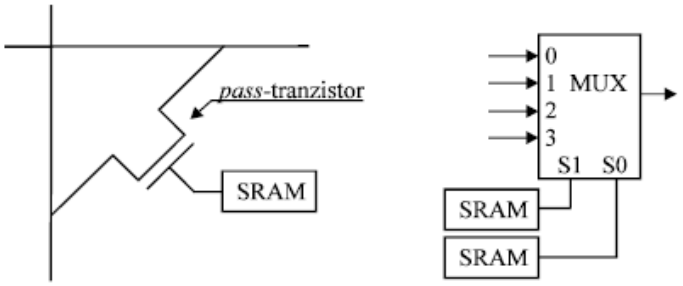
Prsten IOB okružuje pravilno polje CLB-ova. Ovisno o modelu Spartan-3 sklopovi mogu imati jedan stupac, dva stupca ili četiri stupca RAM memorije, svaki stupac sačinjen je od nekoliko 18Kb RAM blokova, svaki blok memorije povezan je sa blokom za množenje, a blokovi za digitalnu kontrolu perioda signala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

5. Što su LUT blokovi, kako se koriste i čemu služe?

Look-Up Tables su generatori funkcijske logike. Koriste se unutar CLB-ova gdje kombinacijom LUTova korištenjem muxa dobivamo mogućnost ostvarenja složenih logičih funkcija. Kod Spartana 3 svaki LUT ima 4 logička ulaza i 1 izlaz. To nam omogućava izvođenje logičkih funkcija sa 4 varijable.

6. KOji tip programibilnih prekidača se koristi u SPARTAN 3 sklopovima?

SRAM tehnologija - Kod ove tehnologije, konfiguriranje uređaja se ostvaruje pomoću *pass* tanzistora i multipleksora koji se upravljaju SRAM ćelijama. Par SRAM ćelija - *pass* tranzistor" se koristi kao programabilna veza između dva žičana segmenta. Kada je u SRAM ćeliji memorirana "jedinica", *pass* tranzistor se ponaša kao zatvoren prekidač malog serijskog otpora. U suprotnom, kada je stanje SRAM ćelije "nula", *pass* tranzistor je otvoren prekidač veoma velikog serijskog otpora.
Za upravljanje multipleksorom, SRAM ćelije su vezane za selektivne ulaze multipleksora. Stanje SRAM ćelija određuje koji je od ulaza multipleksera povezan sa izlazom.



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

Globalni reset služi da uređaj uđe u određeno inicijalno stanje nakon paljenja uređaja. Početna vrijednost se postavlja inicijalno za komponentu i neovisna je o lokalnom resetu. lokalni reset se koristi tako da napišemo proces koji će napraviti inicijalizaciju sklopa.

```
begin
signal arb_onebit : std_logic :='1';  --globalni reset

Process (ckl,rst)
begin

  If rst='1' then
        Arb_onebit <='0';
   End if;

End process;
```

8. ...

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

Kolika je vrijednost signala b nakon izvođenja gornjeg procesa pri čemu se dogodila promjena (event) na signalu a s vrijednosti '0' na '1'?

Njaprije je vrijednost a = 0 što znači da je c = 0 i da je b = 0. Kada se promjeni vrijednost signala a s vrijednosti 0 na 1 znaci da se aktivira proces gdje

druga naredba poništava djelovanje ove prve tako da je ostala jednaka vrijenost signala c pa dakle i signala b = 0.

9. Razlika između 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 moguću ih je koristiti bilo gdje u arhitekturi. Varijable se definiraju unutar procesa ili podprograma i mogu se koristiti samo unutar navedenog procesa ili potprograma.

10. Objasni mem. sustav PicoBlaze procesora!
S obzirom na organizaciju mem. u koju arhitekturu pripada?

Organizacija memorije je Harvardska što znači da se ista memorija dijeli za podatkovni i programski prostor. Programska memorija ima kapacitet za 1024 instrukcija. Radna memorija se sastoji od 16 registara (8b) opće namjene. Scratch Pad memorija se sastoji od 64 B memorije opće namjene. sadržaj bilo kojeg registra se može pohraniti i učitati korištenjem STORE i FETCH.

11. U VHDL-u projektirat 16 bitni brojač (COUNTER) s asinkronim resetom. Brojač omogućuje brojanje naprijed i natrag koji se kontrolira pomoću priključka UP_DOWN ('1' - inkrementiranje, '0' - dekrementiranje)

```
entity counter is port (

        CLK: in STD_LOGIC;
       RESET: in STD_LOGIC;
       UP_DOWN: in STD_LOGIC;
       COUNT: inout STD_LOGIC_VECTOR(3 downto 0)
);
end counter;

architecture broji of counter is
begin
       if  RESET = '1'  then
            COUNT <=  "0000000000000000";
       elsif  CLK = '1'  and  CLK'event then
              if  UP_DOWN = '1'  then
                     COUNT <=  COUNT  +  1;
              else
                     COUNT <=  COUNT   -  1;
              end if;
       end if;
   end process;
end broji;
```

12. Zadana je sljedeća tablica (*) ulazno izlaznih vr. za enkoderski sklop.
Potrebno ga je implementirat (entitet i arh.) u VHDL-u i to pomoću 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 od dek3na8 is
begin
        case D_IN is
                when "000" => D_OUT <= "00000001";
                when "001" => D_OUT <= "00000010";
                when "010" => D_OUT <= "00000100";
                when "011" => D_OUT <= "00001000";
                when "100" => D_OUT <= "00010000";
                when "101" => D_OUT <= "00100000";
                when "110" => D_OUT <= "01000000";
                when "111" => D_OUT <= "10000000";
                when others => NULL;
        end case;

end dekoder;
```

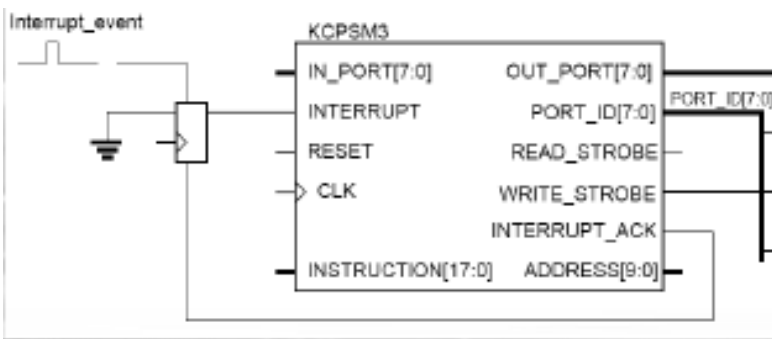
13. Na PicoBlaze procesor spojena je 1 ulazna vanjska (port id 0x80) jedinica i 2 izl. jedinice s port idovima 0x60 i 0x40, te vremenski sklop koji je spojen na prekidnu liniju PicoBlaze procesora.
NA svaki prekid dobiven od prekidne jedinice treba pročitat podatak s ulazne jedinice i ispitat njegov 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 će obrađivat vanj. i prekidnu jedinicu, te

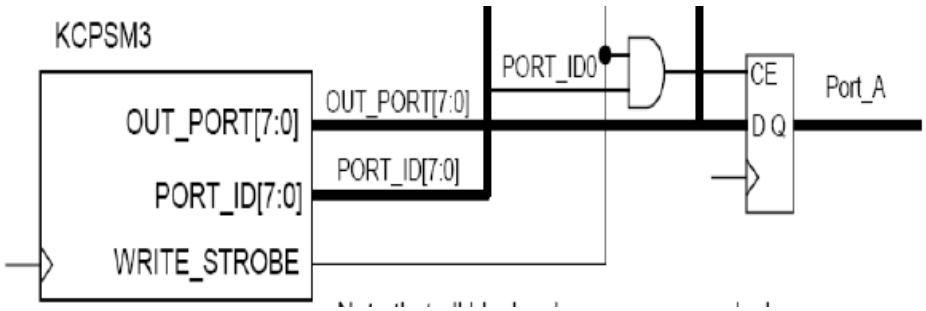
prog. za procesor.

Ovaj program odvija se beskonačno.

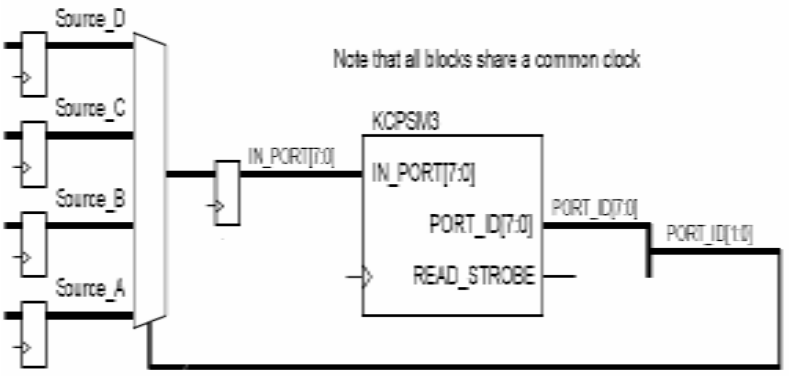
```
        ADDRESS     000
        CONSTANT  PORT_ID1, 80   ;ulazna vj
        CONSTANT  PORT_ID2, 60
        CONSTANT  PORT_ID3, 40

        ENABLE  INTERRUPT

CEKAJ   JUMP   CEKAJ

PREKID

        INPUT  s0,  PORT_ID1
        TEST  s0, 11

        JUMP  C, NEPARAN

PARAN

        OUTPUT  s0, PORT_ID2
        RETURNI

NEPARAN

        OUTPUT  s0, PORT_ID3
        RETURNI

        ; prekidna adresa
        ADDRESS     3FF
        JUMP  PREKID
```



```
prekidna: process(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 prekidna;
```




```
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;
```