**Zadatak 5 (2 boda)**

U programskom jeziku VHDL potrebno je implementirati sljedeću logičku funkciju na razvojon pločici Spartan-6: Preklopnici SW2-SW0 predstavljaju ulaze u logičku funkciju, dok signali LED3-LED1 predstavljaju izlaze logičke funkcije, kako je prikazano u tablici

|  |  |
| --- | --- |
| IZLAZ | FUNKCIJA |
| LED1 | SW0 or SW1 |
| LED2 | SW0 and SW2 |
| LED3 | (SW0 xor SW1) and (SW1 or SW2) |

Rješenje:

entity funkcija is

Port ( SW0: std\_logic; SW1: std\_logic; SW2: std\_logic;

LD1: std\_logic; LD2: std\_logic; LD3: std\_logic;

);

end funkcija;

architecture Beh of funkcija is

begin

LD1 <= SW0 or SW1;

LD2 <= SW0 and SW2;

LD3 <= (SW0 xor SW1) and (SW1 xor SW2);

end Beh;

ovo je bilo samo zagrijavanje, sad slijedi pravi zadatak :)

**Zadatak 6 (15 bodova)**

Na PicoBlaze procesor spojene su dvije ulazne vanjske jedinice (VJ1, adresa: 0x20, VJ2 , adresa: 0x40) i jedna izlazna jedinica (VJ3, adresa: 0x60). Potrebno je rješiti sljedeći zadatak: Na svaki signal s tipkala spojenog na najniži bit na adresi 0x21 treba pročitati podatak sa ulaznih jedinica (podaci su u 8-bitnom 2'k formatu). Na prvi signal čita se podatak sa VJ1, a na drugi signal sa VJ2. Na treći signal potrebno je poslati podatak na VJ3, te postupak ponavljati beskonačno. Ukoliko je podatak primljen s VJ1 paran, onda se podatak s VJ2 šalje nepromijenjen na VJ3. Ako je podatak s VJ1 neparan, onda se na VJ3 šalje negirana vrijednost podatka primljenog s 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 entitet i arhitekturu te nacrtat blok-shemu povezivanja komponenti.

Za procesor i ROM koristite sljedeće deklaracije komponenti:

- - KCPSM2 processor component

component kcpsm3

Port ( address : out std\_logic\_vector(9 downto 0);

instruction : in std\_logic\_vector(17 downto 0);

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

write\_strobe : out std\_logic;

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

read\_strobe : out std\_logic;

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

interrupt : in std\_logic;

interrupt\_ack : out std\_logic;

reset : in std\_logic;

clk : in std\_logic);

end component;

- - ROM component

component prog\_rom

Port ( address : in std\_logic\_vector(9 downto 0);

instruction : out std\_logic\_vector(17 downto 0);

clk : in std\_logic);

end component;

Rješenje:

ADRESS 000

CONSTANT VJ1, 0x20

CONSTANT VJ2, 0x40

CONSTANT VJ3, 0x60

CONSTANT TIPKALO, 0x21

LOAD S 0,00

PETLJA: LOAD, S1, TIPKALO

COMPARE S1, 01

JUMP NZ, PETLJA

ADD S0, 01

SIGNAL1: COMPARE S0,01

JUMP NZ, SIGNAL2

INPUT S2, VJ1

JUMP PETLJA

SIGNAL2: COMPARE S0,02

JUMP NZ, SIGNAL3

INPUT S3, VJ2

JUMP PETLJA

SIGNAL3: SR0 S2 ;pomak bitova udesno, najniži bit „ispada“ u carry zastavicu

JUMP NC, PARAN

XOR S3, FF

PARAN: OUTPUT S3, VJ3

LOAD S0, 00

JUMP PETLJA

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);

tipkalo: in std\_logic;

clk: in std\_logic;

);

end top\_level;

------------------------------------------------------------------------------------------------------------------------------

architecture Beh of top\_level is

component kcpsm3

Port ( address : out std\_logic\_vector(9 downto 0);

instruction : in std\_logic\_vector(17 downto 0);

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

write\_strobe : out std\_logic;

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

read\_strobe : out std\_logic;

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

interrupt : in std\_logic;

interrupt\_ack : out std\_logic;

reset : in std\_logic;

clk : in std\_logic);

end component;

component prog\_rom

Port ( address : in std\_logic\_vector(9 downto 0);

instruction : out std\_logic\_vector(17 downto 0);

clk : in std\_logic);

end component;

------------------------------------------------------------------------------------------------------------------------------

signal address : out std\_logic\_vector(9 downto 0);

signal instruction : in std\_logic\_vector(17 downto 0);

signal port\_id : out std\_logic\_vector(7 downto 0);

signal write\_strobe : out std\_logic;

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

signal read\_strobe : out std\_logic;

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

signal interrupt : in std\_logic;

signal interrupt\_ack : out std\_logic;

signal reset : in std\_logic;

begin

processor: kcpsm3

port map( address => address\_signal,

instruction => instruction\_signal,

port\_id => port\_id\_signal,

write\_strobe => write\_strobe\_signal,

out\_port => out\_port\_signal,

read\_strobe => read\_strobe\_signal,

in\_port => in\_port\_signal,

interrupt => interrupt\_signal,

interrupt\_ack => interrupt\_ack\_signal,

reset => reset\_signal,

clk => clk\_signal);

program: prog\_rom

port map( address => address\_signal,

instruction => instruction\_signal,

clk => clk\_signal);

------------------------------------------------------------------------------------------------------------------------------

ulazne\_vj: process (clk)

begin

if clk' event and clk = '1' then

if port\_id = '0010 0000“ then

in\_port <= data\_vj1;

elsif port\_id = '0100 0000' then

in\_port <= data\_vj2;

elsif port\_id = '0100 0001' then

in\_port <= tipkalo;

end if

end process ulazne\_vj;

izlazna\_vj: process (clk)

begin

if clk' event and clk = '1' then

if write\_strobe = '1' then

if port\_id = '0110 0000' then

data\_vj3 <= out\_port;

end if

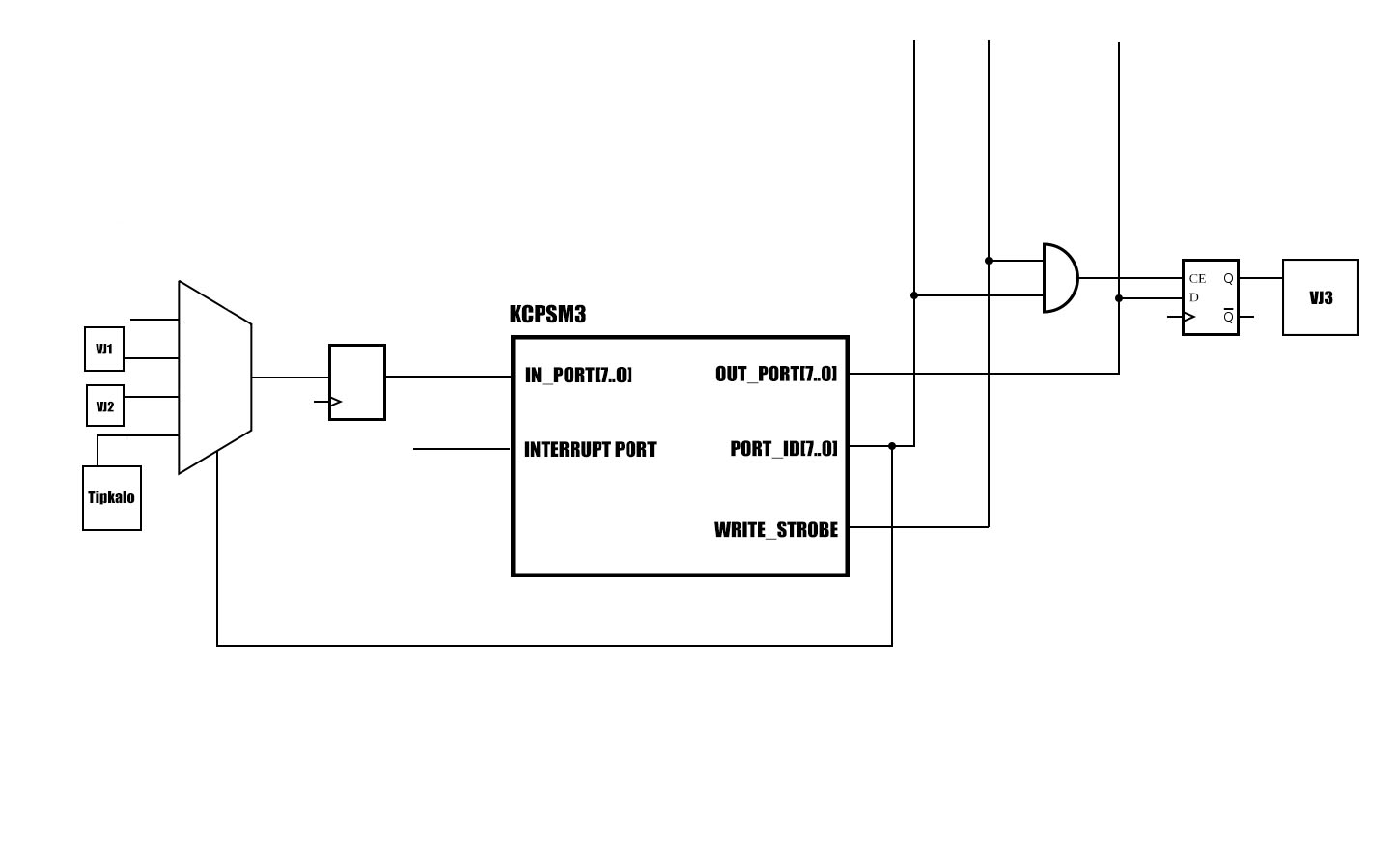
end if

end if

end process izlazna vj;

end Beh;

Blok – shema povezivanja komponenti



Ovo sam pisao po sjećanju na ispit na kojem sam za ove zadatke dobio sve bodove, tako da bi ovo trebalo biti sve točno.

Happy learning :)