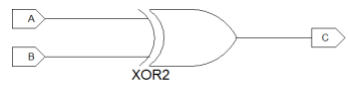
NAČRTOVANJE DIGITALNIH VEZIJ

VAJA 1: REALIZACIJA LOGIČNE FUNKCIJE V XILINX VHDL OKOLJU

Navodilo za delo:
V okolju XILINX ISE ob vsaki nalogi vaje ustvarite nov projekt, ki ga poimenujte VAJAX_Y, kjer je X številka vaje in Y številka naloge. Ob tem se ustvari imenik (VAJAX_Y), v katerega kopirate vse datoteke vaj s spletne strani predmeta. Nekatere naloge so odvisne od prejšnjih zato zberite in kopirajte v imenik vaje *res vse* potrebne datoteke prejšnjih nalog. Potrebne VHDL datoteke za izvedbo neke naloge vedno kopirajte in jih nikar ne povezujte (ang. *link*), ker jih sicer po oddaji na strežnik ne boste mogli povezati skupaj in zato na ustnem izpitu naloge ne bodo delovale.
V razlagi posameznih nalog so uporabljena imena signalov navedeni med besedilom v oklepajih. Isti signali se nahajajo v VHDL predlogi dokumenta. Pri realizaciji uporabljate njihova imena. Ko ste kodo VHDL napisali, naredite morebitno simulacijo (ang. *testbench*) da preverite pravilnost delovanja kode, nato preverite še realizacijo kode na ploščici Basys 2 in šele nato delujoče naloge naložite na strežnik pod kategorijo VAJAX podkategorija Y. Nekatere naloge lahko simulirate doma pred vajami.

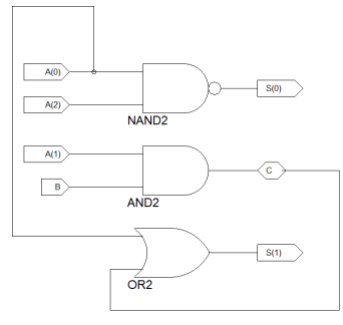
1.1 Ustvarite nov projekt (Vaja1_1) in realizirajte vezje (`xor2vrata.vhd`), katerega arhitektura (`arch`) so dvovhodna vrata XOR. Kopirajte predlogo vaje (xor2vrata.vhd) in ustrezno UCF datoteko (xor2vrata.ucf) v imenik projekta. Delovanje preverite na ploščici Basys 2, tako da spreminjate stanje stikal in opazujete vrednost izhoda na LED diodi. Vhod A se nahaja na stikalu SW0, vhod B pa na stikalu SW1. Izhod C se nahaja na LED diodi LD0.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity xor2vrata is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end xor2vrata;

architecture arch of xor2vrata is
begin

end arch;
```



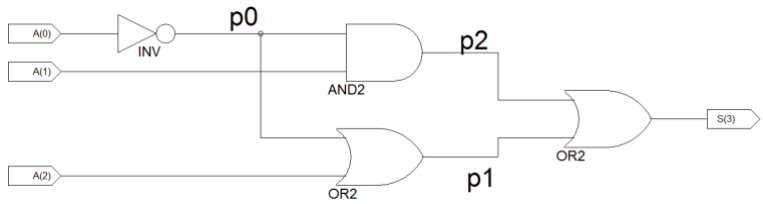
1.2 Ustvarite nov projekt (Vaja1_2) in realizirajte vezje (vaja1_2.vhd), katerega arhitektura (`arch`) realizira spodnje logične operacije. Kopirajte predlogo vaje (vaja1_2.vhd) in ustrezno UCF datoteko (vaja1_2.ucf) v imenik projekta. Delovanje preverite na ploščici Basys 2, tako da spreminjate stanje stikal in opazujete vrednost izhoda na LED diodi. Vektorski vhod A(2:0) se nahaja na stikalih SW0 do SW2 , vhod B pa na stikalu SW3. Izhod S(1:0) se nahaja na LED diodi LD0 in LD1, vhod/izhod C se na LED diodi LD2.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vaja1_2 is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (1 downto 0);
           C : inout STD_LOGIC);
end vaja1_2;

architecture arch of vaja1_2 is
begin
end arch;
```



Zakaj je signal C tipa `inout`? Spremenite kodo tako, da se izognete tipu (`inout`) in postavite C kot izhodni signal tipa (`out`).

1.3 Ustvarite nov projekt (Vaja1_3) in realizirajte vezje (vaja1_3.vhd), katerega arhitektura (arch) realizira spodnje logične operacije. Kopirajte predlogo vaje (vaja1_3.vhd) in ustrezno UCF datoteko (vaja1_3.ucf) v imenik projekta. Delovanje preverite na ploščici Basys 2, tako da spreminjate stanje stikal in opazujete vrednost izhoda na LED diodi. Vektorski vhod A(2:0) se nahaja na stikalih SW0 do SW2. Izhod S(3) je na LED diodi LD3. Vrednosti vmesnega signala p0 prikažite na izhodu S(0), ki je na LED diodi LD0, Vrednosti vmesnega signala p1 prikažite na izhodu S(1), ki je na LED diodi LD1, Vrednosti vmesnega signala p2 prikažite na izhodu S(2), ki je na LED diodi LD2.



a) Oglejte si grafično predstavitev sinteze vezja (Implement→Synthesis→View RTL schematic)

b) Ali zamenjava vrstnega reda ukazov vpliva na rezultat sinteze vezja?

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vaja1_3 is
      Port ( A : in  STD_LOGIC_VECTOR (2 downto 0);
             S : out  STD_LOGIC_VECTOR (3 downto 0)
             );
end vaja1_3;

architecture arch of vaja1_3 is
signal p0, p1, p2: STD_LOGIC;
begin

end arch;
```

1.4 Narišite blok shemo podane funkcije S in jo poenostavite z uporabo pravil Boole-ove algebre. Ustvarite nov projekt (Vaja1_4) in zapišite VHDL kodo vezja (vaja1_4.vhd), katerega arhitektura (arch) realizira opisano funkcijo. Funkcijo realizirajte brez vmesnih signalov. Ta naloga nima svoje UCF datoteke, zato uporabite pripravljeno testno VHDL datoteko (ang. *testbench*), imenovano (vaja1_4_tb.vhd) in preverite pravilnost realizacije v simulatorju.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vaja1_4 is
    Port ( a,b,c : in  STD_LOGIC;
           f : out  STD_LOGIC);
end vaja1_4;

architecture arch of vaja1_4 is
begin

end arch;
```

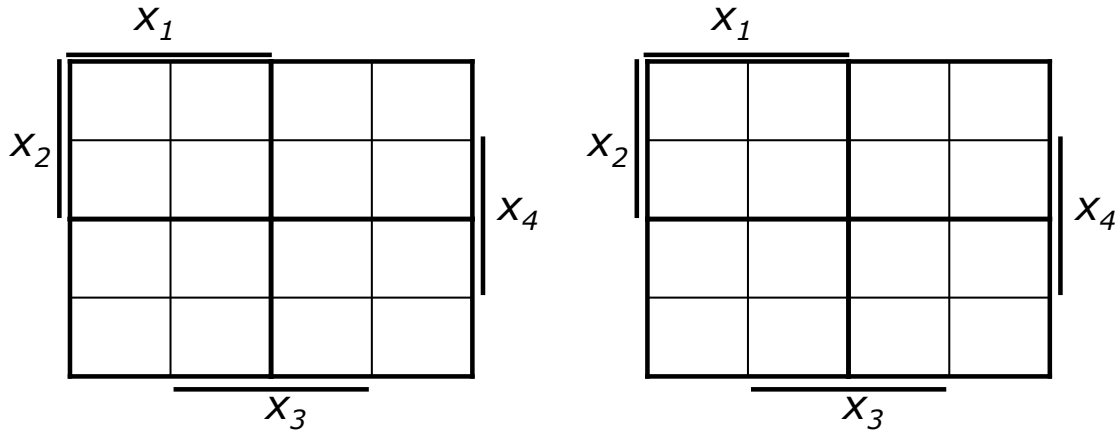
$$f = \left(c \oplus \left(a \uparrow b\right)\right) + \overline{\left(b \downarrow c\right)}$$

1.5 Za spodnjo funkcijo štirih spremenljivk (X1, X2, X3, X4) poiščite oblike MKNO, MDNO, PKNO in MNO. Upoštevajte, da je X1 MSB pri mintermih, X4 LSB. Ustvarite nov projekt (Vaja1_5) in zapišite VHDL kodo vezja (vaja1_5.vhd), katerega arhitektura (`arch`) realizira obliko MNO.

Delovanje preverite na ploščici Basys 2, tako da spreminjate stanje stikal in opazujete vrednost izhoda na LED diodi. Vhodi X1, X2, X3, X4 se nahajajo na stikalih SW3 do SW0. Izhod S se nahaja na LED diodi LD0. Pri realizaciji ne uporabljajte vmesnih signalov.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vaja1_5 is
    Port ( x1, x2, x3, x4 : in  STD_LOGIC;
           f : out  STD_LOGIC);
end vaja1_5;

architecture arch of vaja1_5 is
begin

end arch;
```

$f^4 = V(2,4,5,6,7,10,14)$



f_{PKNO} = _____

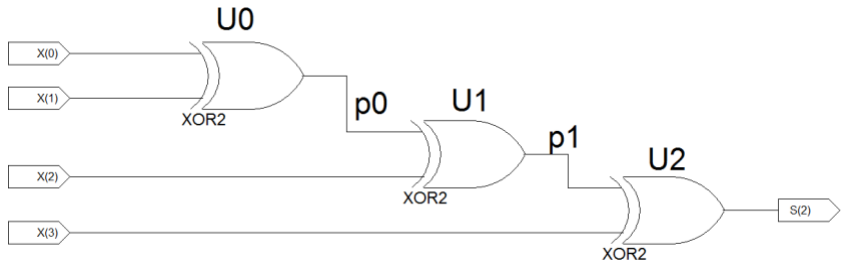
f_{MKNO} = _____

f_{PDMO} =_____

f_{MDNO} = _____

1.6 Ustvarite nov projekt (Vaja1_6) in zapišite VHDL kodo vezja (vaja1_6.vhd), katerega arhitektura (`arch`) realizira spodnje logične operacije z uporabo povezovalnega stavka (`port map`). Povezovalni stavek povezuje tri primere (komponente) vezja (`xor2vrata`) zato kopirajte datoteko (xor2vrata.vhd) v imenik (Vaja1_6).

Delovanje preverite na ploščici Basys 2, tako da spreminjate stanje stikal in opazujete vrednost izhoda na LED diodi. Vektorski vhod X(0:3) se nahaja na stikalih SW3 do SW0. Izhod funkcije S(2) je na LED diodi LD2. Vrednosti vmesnega signala p0 prikažite na izhodu S(0), ki je na LED diodi LD0 in vrednosti vmesnega signala p1 prikažite na izhodu S(1), ki je na LED diodi LD1.




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vaja1_6 is
    Port ( X: in  STD_LOGIC_VECTOR (3 downto 0);
           S : out  STD_LOGIC_VECTOR(2 downto 0)
            );
end vaja1_6;

architecture arch of vaja1_6 is
signal p0, p1: STD_LOGIC;

component xor2vrata
port( A, B: in STD_LOGIC;
      C: out STD_LOGIC);
end component;

begin

end arch;
```

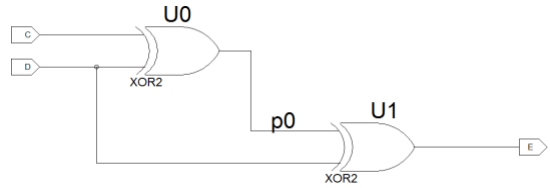
Primer uporabe povezovalnega stavka v novem vezju:

Če želimo ponovno uporabljati vezja (komponente), ki smo jih ustvarili prej, jih moramo v novih projektih deklarirati. To storimo tako, da potrebne datoteke (*.vhd) kopiramo v imenik novega projekta, jih odpremo in kopiramo njihovo deklaracijo entitete. Kopirano vsebino prilepimo v glavno VHDL datoteko novega projekta in zamenjamo rezervirani besedi `entity` s `component`.

```
component ime_komponente
port( A, B: in STD_LOGIC; -- isti vrstni red kot je v deklaraciji enitete
      C: out STD_LOGIC); -- zato je najbolje ce kar skopiramo eniteto in
                         -- spremenimo entity v componenent
end component;
```

Ko so potrebne komponente deklarirane, jih moramo med seboj ustrezno povezati z uporabo `port map` stavka. Ta stavek pomeni opis povezovanja posameznih izhodov in vhodov komponent – torej kateri izhod gre na kateri vhod druge komponente. Pri stavku `port map` je bistveno, da ohranjamo vrstni red zaporedja signalov, kot je določen v deklaraciji komponente. Spodnji primer povezuje realizacijo vezja na desni strani: Izhod iz prvih XOR vrat (ime komponente U1) je vezan na vmesni signal p0, ki se preko `port map` stavka poveže na vhod naslednjih XOR vrat (ime komponente U2). Deklaracija komponente XOR2 izvira iz naloge 1.1.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity primer_port_map is
Port( C, D : in STD_LOGIC;
      E : out STD_LOGIC);
end primer_port_map;

architecture arch of primer_port_map is
signal p0: STD_LOGIC;

component xor2vrata
port( A, B: in STD_LOGIC;
      C: out STD_LOGIC);
end component;

begin
      U0: xor2vrata port map (C, D, p0);
      U1: xor2vrata port map (p0, D, E);
end arch;
```



V primerih, ko je vhodov več in jih težko uredimo po zaporedju signalov komponente, uporabimo imensko povezovanje. Pri tem, vrstni red ni pomemben:

```
begin
      U0: xor2vrata port map (A => C, B => D, C => p0);
      U1: xor2vrata port map (C => E, A => p0, B => D);
end arch;
```

Pri tem imamo možnost, da izhod pustimo ne povezan, in sicer tako da ga deklariramo kot open, npr.: E => open.