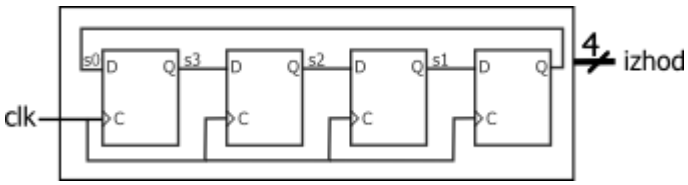
Univerza v Ljubljani
Fakulteta za elektrotehniko

Sekvenčna vezja.

3 Pomikalni register



Naredi vezje, ki je sestavljeno iz štirih D flip-flopov, kot prikazuje shema. Deklariraj notranje signale s0, s1, s2 in s3 in jim določi začetne vrednosti, tako da bo s0 na '1', vsi ostali pa na '0'.

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

entity pomik is
 port (
   clk: in std_logic;
   izhod: out unsigned(3 downto 0)
 );
end pomik;

architecture opis of pomik is
 signal s0: std_logic := '1';
begin
 p: process(clk)
 ...
end opis;
```

a. Za opis flip-flopov uporabi proces s pogojem za fronto ure `rising_edge(clk)` in prireditvenim stavkom za vsak flip-flop.
Preizkusi delovanje pomikalnega registra s simulacijo, kjer nastavi uro:

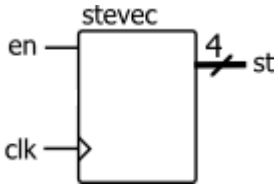
```
force clk 1 0, 0 {50 ps} -r 100
run 2000 ps
```

Ali je vrstni red zapisa prireditvenih stavkov pomemben ?

b. Združi vsa stanja v 4-bitni vektor, ki naj bo izhod vezja. Na katerem mestu je najbolje zapisati ta prireditveni stavek ?

c. Dodaj v opis vezja vhodni signal `en`, ki predstavlja dodaten pogoj za pomikanje podatkov (pomik, kadar je fronta ure in en = '1').
Preveri delovanje s simulacijo.

4 Števec in delilnik



Sinhroni binarni števec je sekvenčno vezje s povratno zanko, v katerem je izhodno stanje vezano nazaj na vhod. Vrednost izhoda binarnega števca je enaka vhodni vrednosti, povečani za 1. Sinhroni števec ima lahko dodaten vhod **en** za omogočanje štetja: stanje števca se povečuje ob fronti ure in pogoju en = '1'.

Naloga

Kopiraj datoteke projekta iz predloge za razvojno ploščo DE0 Nano: DE0_projekt.zip v mapo (npr: c:\proj\iv\ime\stevec). Z dvoklikom na **projekt.qpf** odpri program Quartus in nastavi ime glavne entitete: Assignments > Settings, poišči General in v Top-level entity vnesi: stevec. Nato klikni New, izberi VHDL file in uredi opis števca. Na koncu ne pozabi shraniti datoteke kot: stevec.vhd.

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

entity stevec is
 port (
   clk: in std_logic;
   led: out unsigned(3 downto 0)
 );
end stevec;

architecture opis of stevec is
  signal st: unsigned(3 downto 0);
begin
 ...
end opis;
```

a. Naredi opis 4-bitnega števca in simulacijo v programu Modelsim, ki ga poženeš z Run Simulation > RTL Simulation.

   Namig: uporabi notranji signal **st** za stanje števca, ki naj bo v začetku na "0000", stanje pa na koncu prepiši na izhod **led**. Kaj se zgodi z izhodom, ko pride do najvišje vrednosti ("1111") ?

b. Dodaj v opis števca signal **en,** ki predstavlja dodatni pogoj za omogočanje štetja in preveri delovanje na simulaciji.

c. Dodaj v vezje še en števec (nov proces), ki povečuje vrednost novega 4-bitnega notranjega signala (npr. st2). Števec naj tudi postavlja signal **en**: ko pride števec do najvišje vrednosti, naj dobi **en** vrednost '1', sicer pa '0'.

   Nov števec deluje sedaj kot delilnik ure za prvega - šele ko nov števec prišteje do konca, se bo prvotni števec povečal za 1. Preveri delovanje delilnika na simulaciji.

d. Spremeni delilni števec tako, da bo se bo postavil na 0, ko prišteje do 9 in preizkusi delovanje na simulaciji. Takšen števec imenujemo števec po modulu - števec, ki od 0 do 9 ima modul štetja 10.

   Če želimo videti delovanje števca na razvojni plošči, moramo delilnik ure prilagoditi frekvenci ure clk, ki je v našem primeru 50 MHz. Povečaj velikost vektorja tega števca na 21 bitov in modul štetja na 2 000 000. Prevedi vezje in preizkusi delovanje na plošči DE0 Nano.