



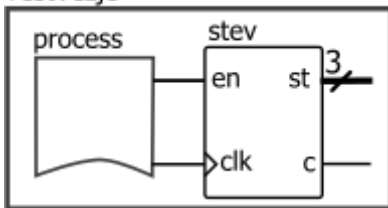
### 3. Vaja: sekvenčna vezja

Opis strukture digitalnih vezij: [vhdl\\_str](#)

## Binarni števec in testna struktura

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. Delovanje števca opišemo s procesom, ki se proži na fronto ure: `rising_edge(clk)`

TestVezje



### Naloga

Naredi 3-bitni števec s signalom za omogočanje (**en**), ki predstavlja pogoj za štetje. Izhod se poveča za 1 ob naraščajoči fronti ure in pogoju `en='1'`. Števec naj ima dodatni izhod **c**, ki se postavi na '1' takoj ko pride števec na najvišjo vrednost "111".

- Naredi opis števca in simulacijo s testno strukturo v jeziku VHDL. V testni strukturi definiramo spreminjanje vhodov vezja, ki ga testiramo: uro (`clk`) in signal za omogočanje (`en`).

Uporabi grafični pripomoček za izdelavo predloge vezja in testne strukture, ki je na voljo na spletni strani [grafTB](#).

### Grafični Test Bench

Circuit type:  Sequential circuit

Name	In/Out	Type	MSB	LSB
en	in	std_logic		0
st	out	unsigned	2	0
c	out	std_logic		0

#### Orodje za izdelavo testnega vezja

- v tabeli določi zunanje priključke vezja (gumb Add Port), ime vezja pa v oknu Entity name
- Izriši graf signalov (Draw Signals)
- VHDL predlogo naredi z Generate Entity
- s klikanjem signalov na grafu nastavi vhode in naredi VHDL Test Bench

Entity name:

TestBench name:

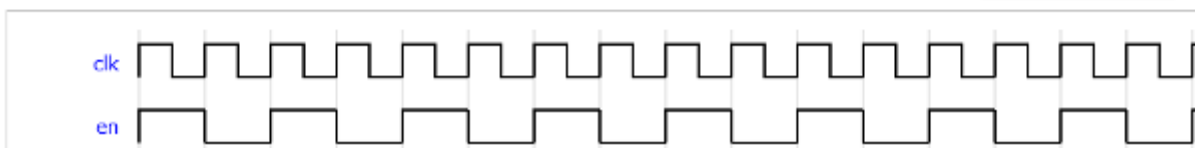
OnClick Bus value input:

Clock cycles:  Period:  ns

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

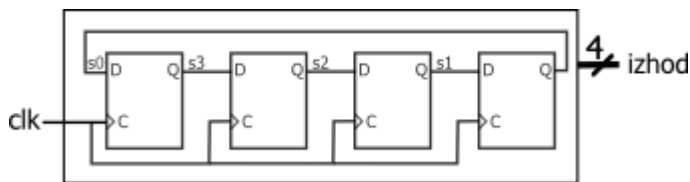
```
entity stev is
port (clk : in std_logic;
      en : in std_logic;
      st : out unsigned(2 downto 0);
      c : out std_logic);
end stev;
```

```
architecture Behavioral of stev
begin
end Behavioral;
```



- V tabeli definiraj vse vhode in izhode testiranega vezja (razen ure) in pri tem pazi na podatkovni tip signalov (npr. **st** naj bo tipa unsigned). Določi št. urnih ciklov (npr. 20) in zapiši ime vezja (**stev**) v okence **Entity name**. Klikni na gumb **Draw Signals** in nato **Generate Entity**, tako da dobiš predlogo VHDL opisa vezja. Dokončaj opis števec - za štetje uporabi notranji signal, ki mu definiraj začetno stanje "000".
- V Grafičnem orodju nastavi s klikanjem vrednosti vhoda **en**, tako da se bo postavil na '1' vsak drugi urni cikel. Nato klikni na gumb **Generate Test Bench** shrani kodo iz okna v datoteko in jo uporabi za izvedbo simulacije.
- Poglej na simulaciji kako deluje izhod **c**, če je pogoj za preliv zapisan v procesu ali izven procesa ?

## Pomikalni register



### Naloga

Naredi vezje, ki je sestavljeno iz štirih D flip-flopov, kot prikazuje shema. Deklariraj notranje signale  $s_0$ ,  $s_1$ ,  $s_2$  in  $s_3$  in jim določi začetne vrednosti, tako da bo  $s_0$  na '1', vsi ostali pa na '0'. Preizkusi delovanje vezja na simulatorju !

- Ali je vrstni red zapisa prireditvenih stavkov pomemben ?
- Združi vsa stanja v 4-bitni vektor, ki naj bo izhod vezja. Na katerem mestu je najboljše zapisati ta prireditveni stavek ?
- Dodaj v opis vezja vhodni signal reset, ki naj postavi signale v začetno stanje. Signal reset naj deluje sinhrono z uro.