

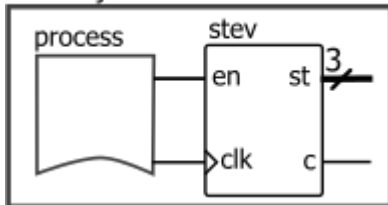


## 5. Vaja: struktarno načrtovanje

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

### Števec in testna struktura

TestVezje



### Naloga

Naredi 3-bitni števec s signalom za omogočanje (**en**) in izhodom **c**, ki naj 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: 20 Period: 10 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);
      c : out std_logic);
end stev;

architecture Behavior 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 števca - 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.

# Pomnilnik ROM

Izhod števca bomo povezali na pomnilnik ROM v katerem bo tabela, ki preslika zaporedne vrednosti števca v kombinacije za izhodne LED. Preslikavo podaja tabela:

adr	d
000	1100
001	0110
010	0011
011	0001
100	0011
101	0110
110	1100
111	1000

## Naloga

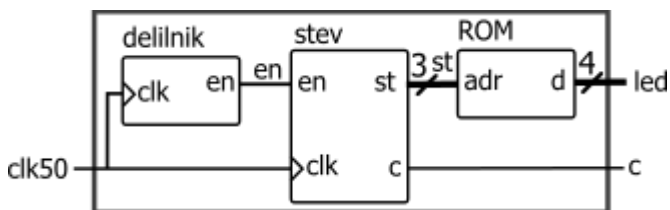
Naredi novo VHDL datoteko za pomnilnik ROM, ki ima 3-bitni nepredznačeni vhod **adr** in 4-bitni izhod **d**. Deklariraj podatkovni tip in notranji signal, ki bo predstavljal vsebino pomnilnika:

```
type memory is array (0 to 7) of unsigned(3 downto 0);  
signal ROM: memory := ( "1100", "0110", ... "1000");
```

V arhitekturnem delu vezja zapiši stavek, ki da na izhod vrednost pomnilnika kot ga določa naslov **adr**:

```
d <= ROM(to_integer(adr));
```

## Struktura



## Naloga

Naredi strukturno vezje, ki povezuje delilnik ure, števec in ROM pomnilnik in preizkusi delovanje na razvojnem sistemu.

Odpri obstoječ projekt: [Struktura.zip](#), ki vsebuje:

- datoteke z opisom projekta, definicij priključkov in uporabniške nastavitve
- Struktura.vhd, glavno datoteko, ki povezuje med seboj komponente vezja
- delilnik.vhd
- stev.vhd
- ROM.vhd