Primjer 1

U VHDL-u realizovati kolo koje obavlja funkciju univerzalnog binarnog brojača, tabela 1. Osim ulaznih signala datih u tabeli 1, kolo ima i asinhroni **reset**. Potrebno je uočiti razliku između **syn_clr** i **reset** signala. Dok se asinhroni **reset** koristi u procesu inicijalizacije sistema, **syn_clr** se odabira na uzlaznoj ivici takt impulsa i koristi se kod sinhronizovanog dizajna.

Tabela 1 - Univerzalni binarni brojač - funkcionalni opis

syn_clr	load	en	up	q	Operacija
1	-	-	-	00 … 00	Sinhroni clear
0	1	-	-	d	Paralelno učitavanje
0	0	1	1	q+1	Broji naprijed
0	0	1	0	q-1	Broji nazad
0	0	0	-	q	pause

Rješenje

VHDL kod moguće realizacije univerzalnog binarnog brojača dat je u listingu 4.

Listing 4 – Univerzalni binarni brojač

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter is
generic(N: integer := 4);
	port
	(
		clk, reset: in std_logic;
		syn_clr, load, en, up: in std_logic;
		d: in std_logic_vector(N-1 downto 0);
		max_tick, min_tick: out std_logic;
		q: out std_logic_vector(N-1 downto 0)
	);
	end counter;
architecture arch of counter is
	signal r_reg: unsigned(N-1 downto 0);
	signal r_next: unsigned(N-1 downto 0);
begin
	process(clk, reset)
	begin
		if (reset = '1') then
			r_reg <= (others=>'0');
		elsif (clk' event and clk = '1') then
			r_reg <= r_next;
		end if;
	end process;
	r_next <= (others=>'0') when syn_clr = '1' else
		  unsigned(d) when load = '1' else
		  r_reg + 1 when en = '1' and up = '1' else
		  r_reg - 1 when en = '1' and up = '0' else
		  r_reg;
	q <= std_logic_vector(r_reg);

	max_tick <= '1' when r_reg = (2**N-1) else '0';
	min_tick <= '1' when r_reg = 0 else '0';
```

```
end arch;
```

U radnom direktorijumu kreirati fajl **counter_test.txt**, čiji sadržaj je dat u listingu 5.

VHDL **testbench** kod dat je u listingu 6.

Listing 5 - counter_test.txt

```
--- vector file for counter
-- time clk reset sync_clr load en up d
10 010011 0000
20 110011 0000
30 000111 1000
40 100111 1000
50 000011 0000
60 100011 0000
70 000011 0000
80 100011 0000
90 000111 1111
100 100111 1111
110 000011 0000
120 100011 0000
130 000011 0000
140 100011 0000
150 000011 0000
160 100011 0000
```

Listing 6 – Testbench fajl za univerzalni binarni brojač

```
entity counter_testbench is end;

library IEEE;
use ieee.std_logic_1164.all;
use std.textio.all;
use ieee.std_logic_textio.all;

use ieee.numeric_std.all;

architecture stimonly of counter_testbench is

constant N : integer := 4;
component counter
	port
	(
		clk, reset: in std_logic;
		syn_clr, load, en, up: in std_logic;
		d: in std_logic_vector(N-1 downto 0);
		max_tick, min_tick: out std_logic;
		q: out std_logic_vector(N-1 downto 0)
	);
end component;

signal clk, reset, syn_clr, load, en, up : std_logic;
signal max, min: std_logic;
signal q, d : std_logic_vector(N-1 downto 0);

begin
uut: counter
	port map(clk => clk, reset => reset, syn_clr => syn_clr,
				load => load, en => en, up => up, d => d,
				max_tick => max, min_tick => min, q => q);

test:
```

```
process
	variable tmpclk, tmprst, tmpclr, tmpld, tmpup, tmpen : std_logic;
	variable tmpdin, tmpqout : std_logic_vector(N-1 downto 0);
	file vector_file : text is in "counter_test.txt";
	variable l : line;
	variable vector_time : time;
	variable r : real;
	variable good_number, good_val : boolean;
	variable space : character;
begin
while not endfile(vector_file) loop
	readline(vector_file, l);
	read(l, r, good => good_number);
	next when not good_number;
		vector_time := r * 10 ns;
			if (now < vector_time) then
				wait for vector_time - now;
			end if;
	read(l, space);
	read(l, tmpclk, good_val);
	assert good_val report "bad clk value";

	read(l, tmprst, good_val);
	assert good_val report "bad rst value";

	read(l, tmpclr, good_val);
	assert good_val report "bad sync_clr value";

	read(l,tmpld, good_val);
	assert good_val report "bad load value";

	read(l,tmpen, good_val);
	assert good_val report "bad en value";

	read(l,tmpup, good_val);
	assert good_val report "bad up value";

	read(l, space);
	read(l, tmpdin, good_val);
	assert good_val report "bad din value";
	clk <= tmpclk;
	reset <= tmprst;
	syn_clr <= tmpclr;
	load <= tmpld;
	up <= tmpup;
	en <= tmpen;
	d <= tmpdin;

end loop;
	assert false report "Test complete";
wait;
end process;
end;
```

Ponoviti simulaciju koristeći VHDL **testbench** dat u listingu 7.

Listing 7 – Testbench fajl za univerzalni binarni brojač

```
library ieee;
use ieee.std_logic_1164.all;

entity counter_testbench_1 is
```

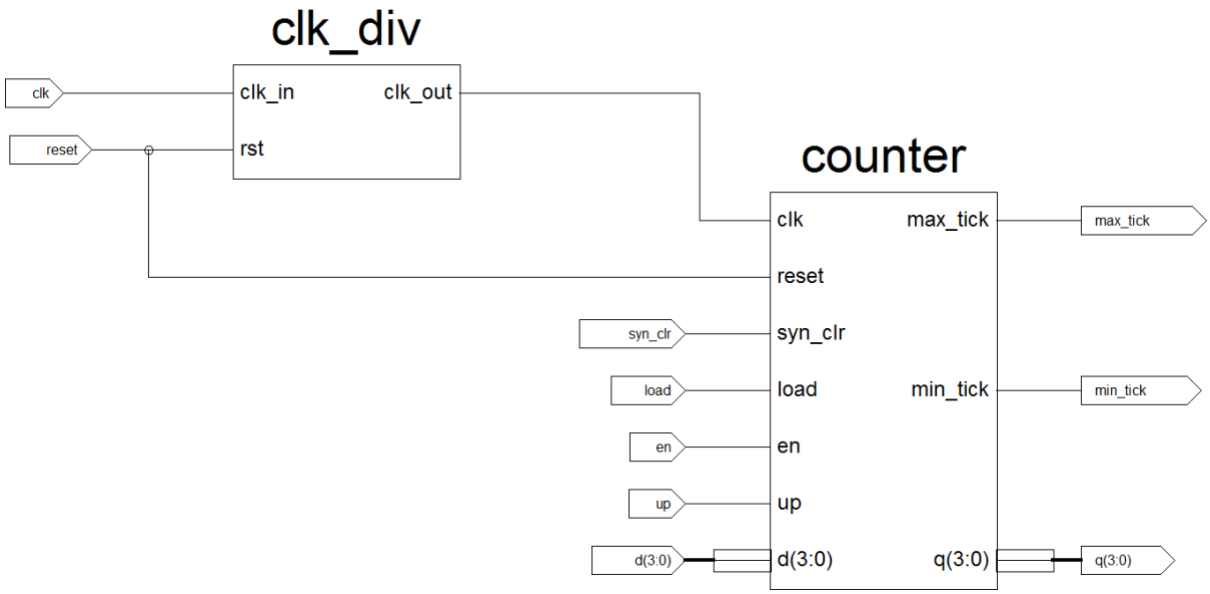
```
end counter_testbench_1;

architecture arc_counter_testbench_1 of counter_testbench_1 is

    -- Component Declaration for the Unit Under Test (UUT)

    component counter
    port(
         clk : in  std_logic;
         reset : in  std_logic;
         syn_clr : in  std_logic;
         load : in  std_logic;
         en : in  std_logic;
         up : in  std_logic;
         d : in  std_logic_vector(3 downto 0);
         max_tick : out  std_logic;
         min_tick : out  std_logic;
         q : out  std_logic_vector(3 downto 0)
        );
    end component;

   --Inputs
   signal clk : std_logic := '0';
   signal reset : std_logic := '0';
   signal syn_clr : std_logic := '0';
   signal load : std_logic := '0';
   signal en : std_logic := '0';
   signal up : std_logic := '0';
   signal d : std_logic_vector(3 downto 0) := (others => '0');

   --Outputs
   signal max_tick : std_logic;
   signal min_tick : std_logic;
   signal q : std_logic_vector(3 downto 0);

   -- Clock period definitions
   constant clk_period : time := 10 ns;

begin

       -- Instantiate the Unit Under Test (UUT)
   uut: counter port map (
          clk => clk,
          reset => reset,
          syn_clr => syn_clr,
          load => load,
          en => en,
          up => up,
          d => d,
          max_tick => max_tick,
          min_tick => min_tick,
          q => q
        );

   -- Clock process definitions

   clk_process :process
   begin
             clk <= '0';
             wait for clk_period/2;
             clk <= '1';
             wait for clk_period/2;
   end process;

   --- reset asserted for T/2
```

```
    reset <= '1', '0' after clk_period/2;

--- other stimulus

    process
    begin

    -- initial input

    syn_clr <= '0';
    load <= '0';
    en <= '0';
    up <= '1';
    d <= (others => '0');
    wait until falling_edge(clk);
    wait until falling_edge(clk);

    -- test load

    load <= '1';
    d <= "0011";
    wait until falling_edge(clk);

    load <= '0';

    -- pause

    wait until falling_edge(clk);
    wait until falling_edge(clk);

    -- test sync clear

    syn_clr <= '1';
    wait until falling_edge(clk);
    syn_clr <= '0';

    -- test up counter

    up <= '1';
    en <= '1';
    for i in 0 to 10 loop
          wait until falling_edge(clk);
    end loop;
    en <= '0';
    wait until falling_edge(clk);
    wait until falling_edge(clk);
    en <= '1';
    wait until falling_edge(clk);
    wait until falling_edge(clk);

    -- test down counter

    up <= '0';
    for i in 0 to 10 loop
          wait until falling_edge(clk);
    end loop;

    -- other wait conditions

    up <= '1';
    wait on min_tick;
    up <= '0';
    wait for 4*clk_period;
    en <= '0';
```

```
        wait for 4*clk_period;

        -- terminate simulation

        assert false
               report "Simulation completed"
        severity failure;

end process;

end arc_counter_testbench_1;
```

Prikazati rezultate simulacije.

U cilju verifikacije rada sistema, razvijeno je testno kolo prikazano na slici 2.



Slika 2. Counter - testno kolo

VHDL kod za modul **clk_div** je dat u listingu 8.

Listing 8 – VHDL kod za modul clk_div

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;

entity clk_div is
    port (
           clk_in : in  std_logic;
           rst : in  std_logic;
           clk_out : out  std_logic
         );
end clk_div;

architecture clk_div_arc of clk_div is
signal count : unsigned(22 downto 0) := (others => '0');
```

```
signal count_next : unsigned(22 downto 0) := (others => '0');
begin
	process(rst, clk_in)
	begin
		if rst = '1' then
			count <= (others => '0');
		elsif (clk_in'event and clk_in = '1') then
			count <= count_next;
		end if;
	end process;
	count_next <= (others => '0') when count = 5000000 else
			count + 1;
	clk_out <= '0' when count < 2500000 else
		   '1';
end clk_div_arc;
```

Izvršiti procese **synthesize**, **translate**, **map** i **place** & **route**.

Kreirati **constraints** fajl, listing 9.

Listing 9 – Constraints fajl za univerzalni binarni brojač

```
NET "d<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "d<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;
NET "d<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
NET "d<3>" LOC = "N17" | IOSTANDARD = LVTTL | PULLUP ;

NET "clk" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "clk" PERIOD = 20.0ns HIGH 40%;


NET "q<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "q<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "q<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "q<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

NET "max_tick" LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "min_tick" LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

NET "reset" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "en" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "up" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "load" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
```

Izvršiti implementaciju kola uz pomoć **Spartan-3E Starter Kit** razvojne platforme (pogledati uputstvo u okviru vježbi 2) i verifikovati rad kola.