Cvičenie 6
Čítače vo VHDL, Hierarchický dizajn, Úvod do skriptov pre ModelSIM

Zadanie:
Realizujte parametrizovateľný čítač na doske CPLD_KIT, tak by výstup čítača bol zobrazený na 7 segmentovom displeji. Parametre majú byť: frekvencia pričítavania, horný limit v rozsahu 1 až 15 a krok pričítavania v rozsahu 1 až 5. Popis celého čítača realizujte hierarchicky, teda zvlášť popis: parametrizovateľnej deličky, samotného čítača a dekóderu. Signály medzi uvedenými popismi poprepájajte v rámci popisu „Top Level" entity. Projekt odsimulujte v ModelSIMe tak, aby boli zobrazené nielen signály z Test benchu, ale aj interné signály z „Top Level" entity. Projekt nahrajte do adresára BASIC_CNT, a nazvite ho BASIC_CNT !!!

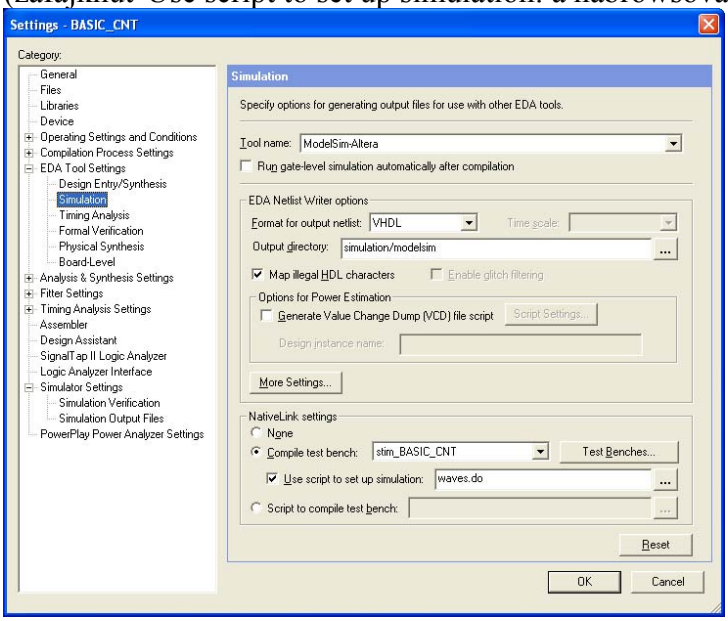
Riešenie:
Výklad k celému projektu bude uskutočnený na cvičení a základná filozofia hierarchického dizajnu je naznačená na obrázku na 2. strane. V rámci tohto dokumentu budú uvedené zdrojové kódy a postup ako do dizajnu zakomponovať jednoduchý konfiguračný skript pre ModelSIM, ktorý do výstupného grafického okna vloží požadované priebehy signálov a nastaví sa v ňom konečný čas simulácie.

Výpis konfiguračného skriptu pre ModelSIM: waves.do
(Pre celkové nastavenie simulácie tento skript nestačí. Viac sa o týchto skriptoch dozviete na nasledujúcich cvičeniach. V skutočnosti QUARTUS II generuje ešte jeden .do súbor, *(napr: BASIC_CNT_run_msim_rtl_vhdl.do pre funkčnú simuláciu),* v ktorom sú zvyšné nastavenia simulácie)

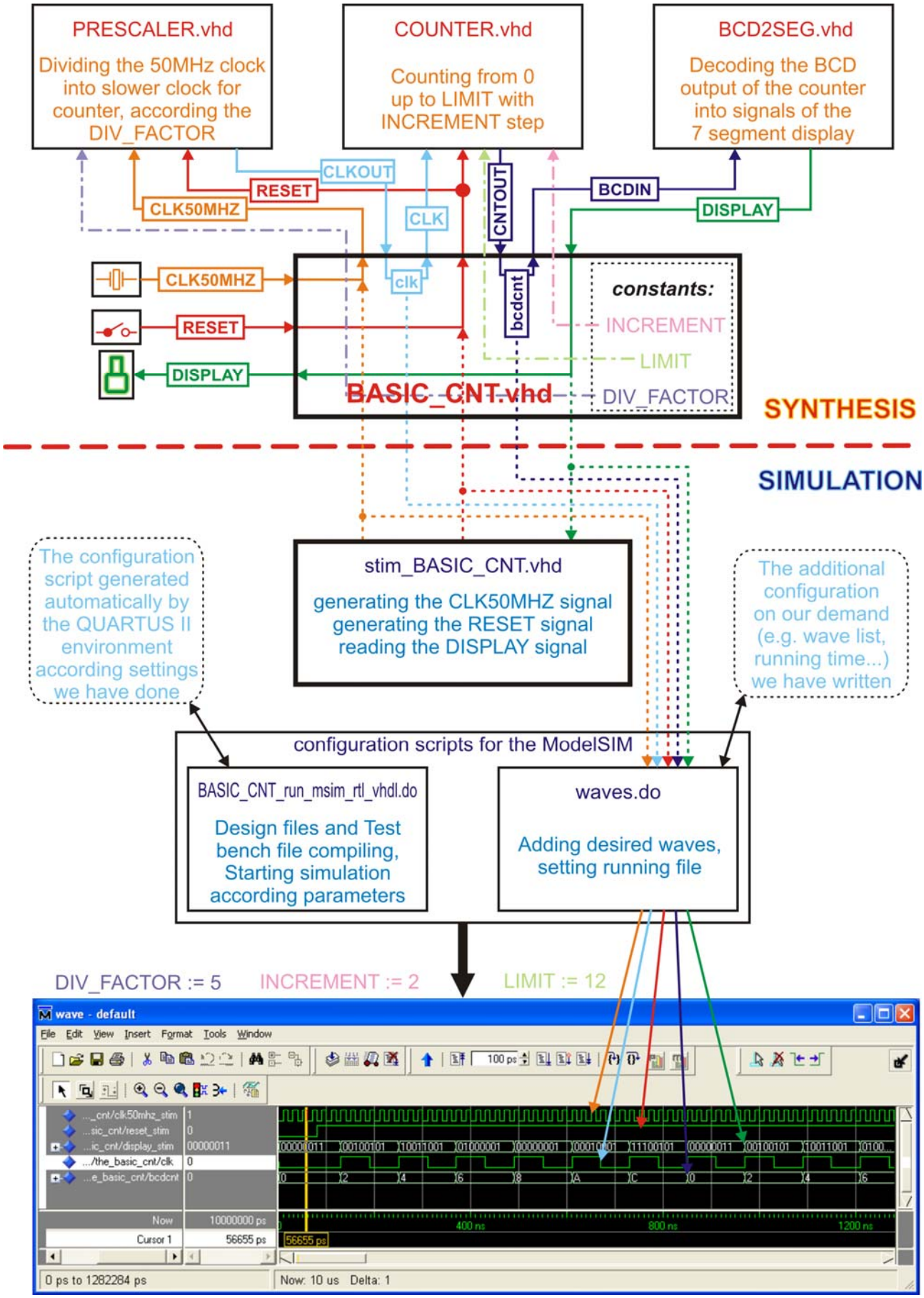
waves.do:
```
#adding waves
add wave sim:/stim_basic_cnt/clk50mhz_stim
add wave sim:/stim_basic_cnt/reset_stim
add wave sim:/stim_basic_cnt/display_stim
add wave sim:/stim_basic_cnt/the_basic_cnt/clk
add wave sim:/stim_basic_cnt/the_basic_cnt/bcdcnt

#setting running time
run 10us
```

Nastavenie v QUARTUS II: horné menu>Assigments>EDA Tool Settings>Simulation (zafajknúť Use script to set up simulation: a nabrowsovať waves.do)



Znázornenie usporiadania a logického poprepájania súborov projektu z hľadiska syntézy a simulácie.



BASIC_CNT.vhd:

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

entity BASIC_CNT is
port(
    CLK50MHZ    : in std_logic;
    RESET       : in std_logic;
    DISPLAY     : out std_logic_vector(7 downto 0)
    );
end BASIC_CNT;

architecture arch of BASIC_CNT is

  constant DIV_FACTOR   : natural range 1 to 25000000 := 5;
  constant LIMIT        : natural range 1 to 15       := 12;
  constant INCREMENT    : natural range 1 to 5        := 2;

  signal clk            : std_logic;
  signal bcdcnt         : std_logic_vector(3 downto 0);

  component PRESCALER is
    generic(
      DIV_FACTOR  : natural range 1 to 25000000 :=  10
    );
    port(
      CLK50MHZ    : in std_logic;
      RESET       : in std_logic;
      CLKOUT      : out std_logic
    );
  end component;

  component COUNTER is
    generic(
      LIMIT       : natural range 1 to 15 := 10;
      INCREMENT   : natural range 1 to 5  := 1
    );
    port(
      CLK         : in std_logic;
      RESET       : in std_logic;
      CNTOUT      : out std_logic_vector(3 downto 0)
    );
  end component;

  component BCD2SEG is
    port(
      BCDIN       : in std_logic_vector  (3 downto 0);
      DISPLAY     : out std_logic_vector (7 downto 0)
    );
  end component;

begin

  the_PRESCALER: PRESCALER
    generic map(
      DIV_FACTOR => DIV_FACTOR
    )
    port map(
      CLK50MHZ   => CLK50MHZ,
      RESET      => RESET,
      CLKOUT     => clk
    );

  the_COUNTER: COUNTER
    generic map(
      LIMIT      => LIMIT,
```

```
      INCREMENT  => INCREMENT
    )
    port map(
      CLK        => clk,
      RESET      => RESET,
      CNTOUT     => bcdcnt
    );

  the_BCD2SEG: BCD2SEG
    port map(
      BCDIN       => bcdcnt,
      DISPLAY     => DISPLAY
    );

end arch;
```

PRESCALER.vhd:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all; -- using comparison ""<""

entity PRESCALER is
  generic(
    DIV_FACTOR  : natural range 1 to 25000000 :=  10
  );
  port(
    CLK50MHZ    : in std_logic;
    RESET       : in std_logic;
    CLKOUT      : out std_logic
  );
end PRESCALER;

architecture arch of PRESCALER is

    signal cnt : std_logic_vector (24 downto 0);
    signal clk : std_logic;

begin

  CLKOUT <= clk;

  process(RESET,CLK50MHZ)
    begin
      if (RESET = '0') then
         cnt <= (others => '0');
         clk <= '0';
      elsif (rising_edge(CLK50MHZ)) then
         if (cnt < DIV_FACTOR/2) then
           cnt <= cnt + 1;
         else
           cnt <= (others => '0');
           clk <= not clk;
         end if;
      end if;
    end process;
end arch;
```

COUNTER.vhd:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all; -- using comparison ""<""

entity COUNTER is
  generic(
```

```
    LIMIT       : natural range 1 to 15 := 10;
    INCREMENT   : natural range 1 to 5  := 1
  );
  port(
    CLK         : in std_logic;
    RESET       : in std_logic;
    CNTOUT      : out std_logic_vector(3 downto 0)
  );
end COUNTER;

architecture arch of COUNTER is

  signal cnt : std_logic_vector (3 downto 0);

begin

  CNTOUT <= cnt;

  process(RESET,CLK)
    begin
      if (RESET = '0') then
         cnt <= (others => '0');
      elsif (rising_edge(CLK)) then
         if (cnt < LIMIT) then
           cnt <= cnt + INCREMENT;
         else
           cnt <= (others => '0');
         end if;
      end if;
    end process;
end arch;
```

BCD2SEG.vhd:

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

entity BCD2SEG is
  port(
    BCDIN   : in std_logic_vector  (3 downto 0);
    DISPLAY : out std_logic_vector (7 downto 0)
  );
end BCD2SEG;

architecture arch of BCD2SEG is
begin
  with BCDIN select
    DISPLAY <=
      "00000011" when "0000", -- 0
      "10011111" when "0001", -- 1
      "00100101" when "0010", -- 2
      "00001101" when "0011", -- 3
      "10011001" when "0100", -- 4
      "01001001" when "0101", -- 5
      "01000001" when "0110", -- 6
      "00011111" when "0111", -- 7
      "00000001" when "1000", -- 8
      "00001001" when "1001", -- 9
      "00010001" when "1010", -- A
      "11000001" when "1011", -- b
      "11100101" when "1100", -- c
      "10000101" when "1101", -- d
      "01100001" when "1110", -- E
      "01110001" when "1111", -- F
      "11111111" when others; -- pre ine moznosti
end arch;
```

stim_BASIC_CNT.vhd
(nezabudnite nastaviť vlastnosti tohto súboru ako „VHDL Test Bench File“)

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

entity stim_BASIC_CNT is
end stim_BASIC_CNT;

architecture stim of stim_BASIC_CNT is

  component BASIC_CNT is
    port(
      CLK50MHZ    : in std_logic;
      RESET       : in std_logic;
      DISPLAY     : out std_logic_vector(7 downto 0)
    );
  end component;

  constant TIME_PERIOD : time := 20ns;

  signal CLK50MHZ_stim, RESET_stim : std_logic := '0';
  signal DISPLAY_stim              : std_logic_vector(7 downto 0);

begin

  the_BASIC_CNT: BASIC_CNT
    port map (
      CLK50MHZ => CLK50MHZ_stim,
      RESET    => RESET_stim,
      DISPLAY  => DISPLAY_stim
    );

  -- generating the 50MHz Clock signal
  CLK50MHZ_stim <= not CLK50MHZ_stim after TIME_PERIOD/2;

  -- hold reset for 4 clock cycles_
  stimulus: process
    begin
      RESET_stim <= '0';
      wait for TIME_PERIOD*4;
      RESET_stim <= '1';
      wait;
    end process stimulus;

end stim;
```