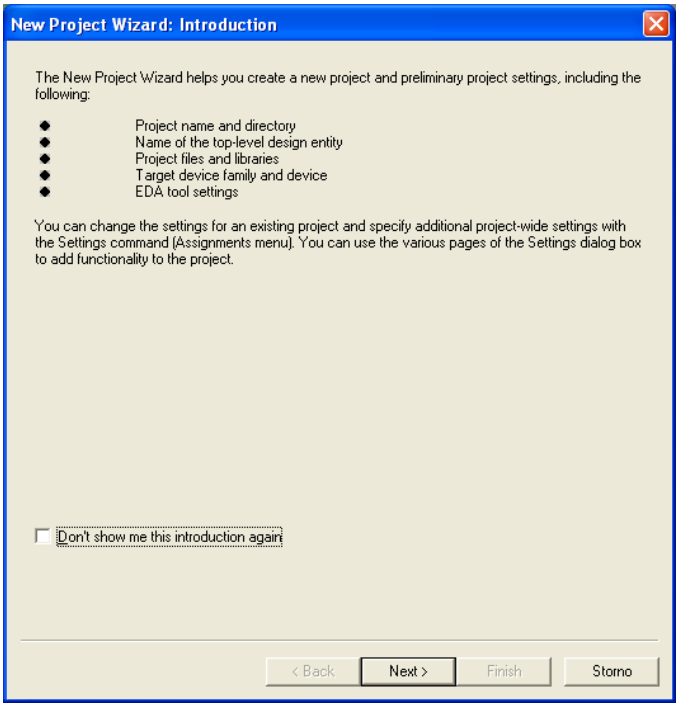
Cvičenie 3
Funkčná a časová simulácia kombinačnej logiky v ModelSime.

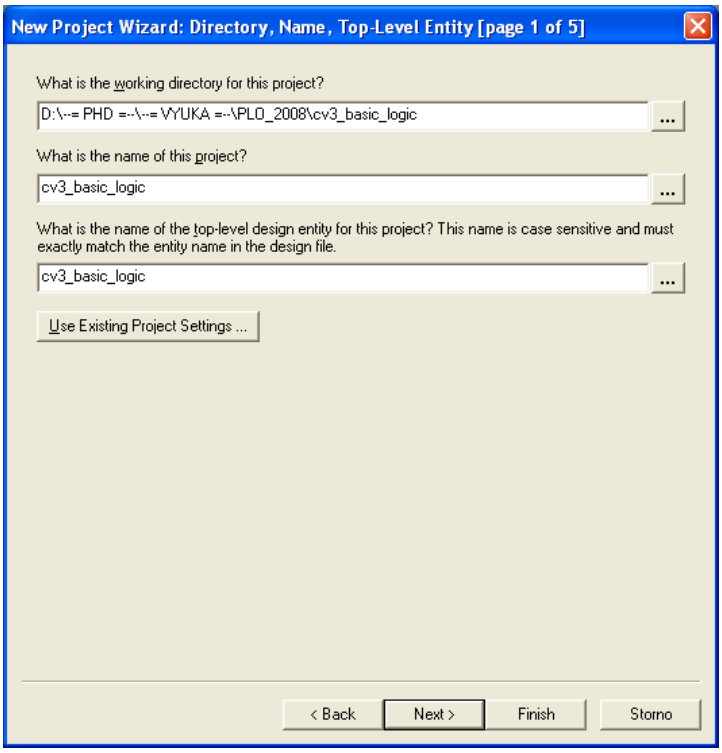
Otvoriť Quartus II

Horné menu: File: New Project Wizard



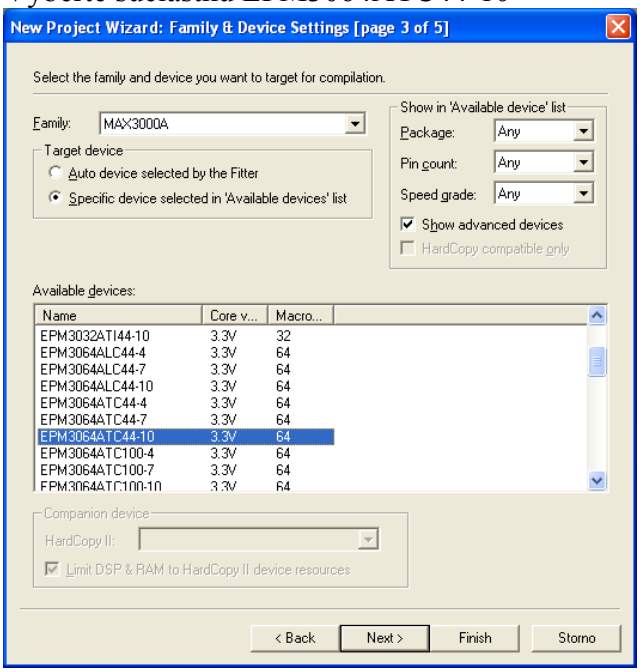
Next...

Zadajte cestu k projektu, názov projektu a meno „top level design entity“ (odporúčam používať rovnaké názvy)



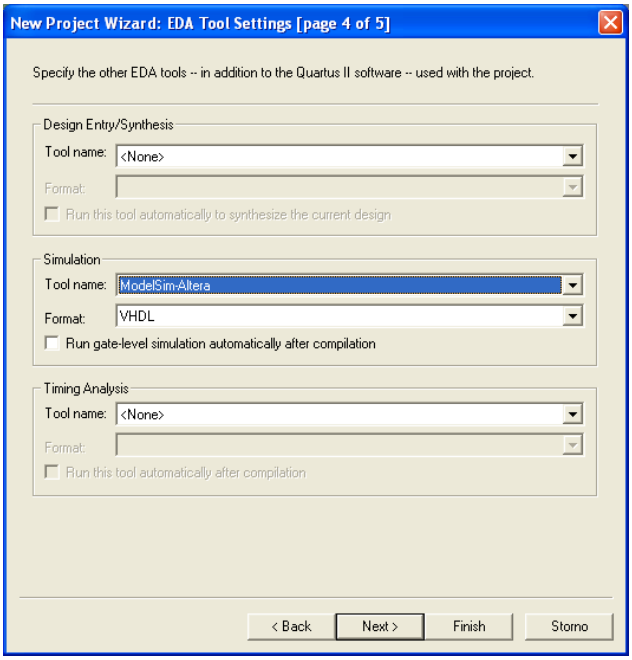
Next...
Next...

Vyberte súčiastku EPM3064ATC44-10

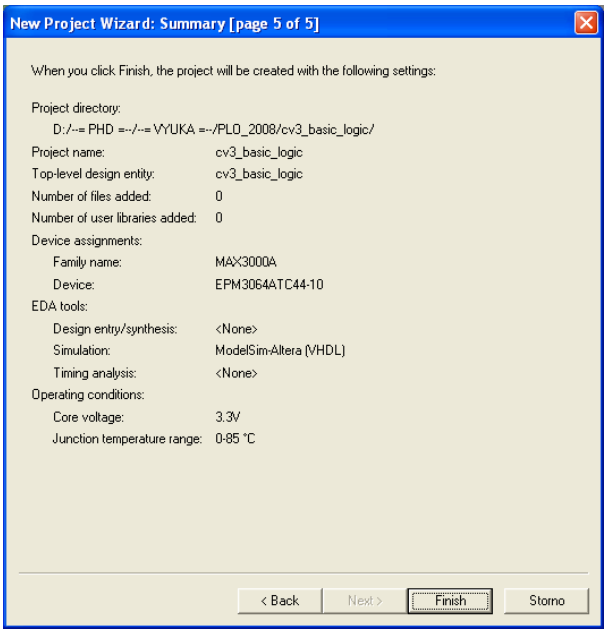


Next...

Výber EDA (Electronic Design Automation) nástrojov. Pre simuláciu budeme používať ModelSim-Altera (domáca inštalácia) alebo ModelSim (školská inštalácia) :

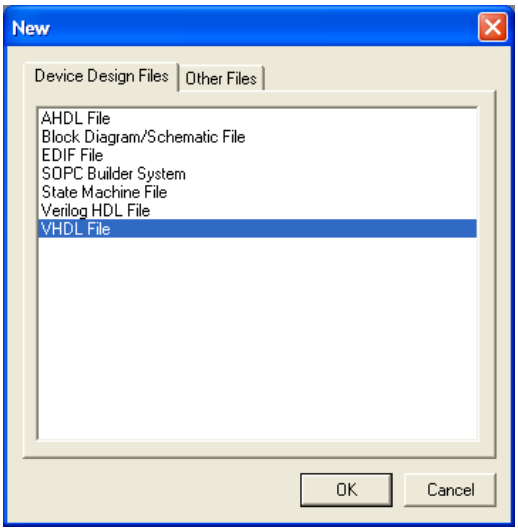


(Pozor!!! Toto dialogove okno je z verzie Quartusu 7.2, v školskej inštalácii je Quartus verzie 6.1 a vyzerá inak, avšak, filozofia výberu ModelSimu sa nemení.)
Next..



Finish.

Teraz vytvoríme popis jednoduchej logiky (Hradlo AND) v jazyku VHDL.
Horné menu: File: New: VHDL File



OK.

Na začiatku kódu je nutné uviesť knižnice, v ktorých sú definované napr. typy signálov, ktoré nie sú štandardne zahrnuté vo VHDL:

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

Budeme používať knižnicu IEEE (Institute of Electrical and Electronics Engineers). Z nej použijeme definíciu štandardnej logiky „ieee.std_logic_1164“ (viac na prednáške). Túto knižnicu budeme používať vždy.

Popis Entity. Entitu si môžeme predstaviť ako elektronickú logickú súčiastku, ktorá má svoje meno, vstupy a výstupy. Tento popis musí obsahovať každý VHDL kód, ktorým popisujeme hardvér, ktorý chceme syntetizovať v CPLD alebo FPGA.

```
entity cv3_basic_logic is
  port(
    my_input  :  in std_logic_vector (1 downto 0);
    my_output : out std_logic
  );
end cv3_basic_logic;
```

entity, is, port, in, out, downto, end sú klúčové slová a v editore sú zvýraznené modrou farbou. „cv3_basic_logic“ je názov entity. „my_input“ je vstup, v tomto prípade dvojbitový vektor. „my_output“ je výstup, v tomto prípade jeden bit.

Ďalej pokračujeme popisom architektúry. Vo vnútri architektúry bude popis logiky, ktorá bude syntetizovaná. Aj keď VHDL umožňuje použitie viacerých architektúr, v rámci jedného VHDL súboru, vždy budeme používať len jednu.

```
architecture arch of cv3_basic_logic is
begin

end arch;
```

architecture, of, is, begin, end sú kľúčové slová. „arch“ je názov architektúry. Riadok „end arch“ je posledným riadkom v súbore a zvyšný kód budeme písať medzi „begin“ a „end arch“.

Pokračujeme popisom samotnej logiky. Keďže chceme realizovať hradlo AND priradíme výstupu AND kombináciu vstupného vektora:

```
my_output <= my_input(1) and my_input(0);
```

symbol <= sa používa na priradenie jedného signálu k druhému signálu. Okrem AND môžeme použiť OR pre logický súčet, alebo NOT pre logickú negáciu.

Súbor uložíme ako cv3_basic_logic.vhdl.

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

entity cv3_basic_logic is
  port(
    my_input  : in std_logic_vector (1 downto 0);
    my_output : out std_logic
  );
end cv3_basic_logic;

architecture arch of cv3_basic_logic is
begin

  my_output <= my_input(1) and my_input(0);

end arch;
```

Skompilovať a priradiť piny podľa postupu z predošlého cvičenia.
Súbor je možné odsimulovať v Quartuse podľa postupu uvedenom v predošlom cvičení.

Simulácia v ModelSime.
Filozofia: ModelSim je externý EDA program, ktorý buď treba Quartusom zavolať (spolu s konfiguráciou simulácie, ktorú Quartus za nás urobí), alebo spustiť samostatne (tu bude treba konfiguráciu popísať skriptom). Na tomto cvičení bude uvedený postup, ako spustiť simuláciu v ModelSime prostredníctvom Quartusu. Pre simuláciu bude treba simulovanej logike vnútiť vstupy, ktoré nezadáme graficky, ale VHDL kódom. Takýto vstup je pomerne jednoduché vytvoriť a editovať, zvlášť pre simulácie rozsiahlych dizajnov.

Vytvoríme simulačný VHDL skript (tzv. test bench)
Horné menu: File: New: VHDL File:

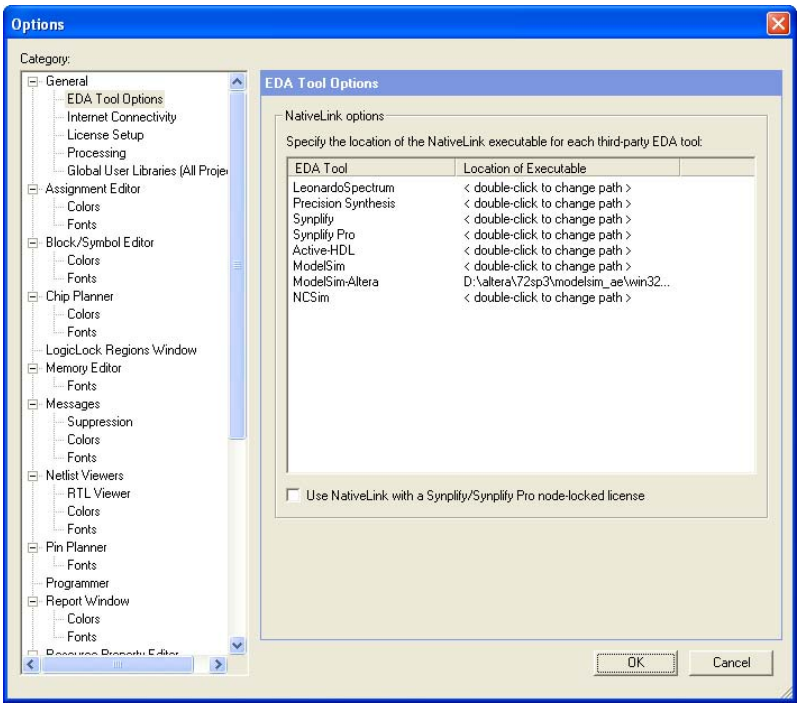
```
library ieee;
use ieee.std_logic_1164.all;

entity stim_cv3_basic_logic is
end stim_cv3_basic_logic;
```

```
architecture stim of stim_cv3_basic_logic is

  component cv3_basic_logic is
  port(
    my_input  : in std_logic_vector (1 downto 0);
    my_output : out std_logic
  );
  end component;

  signal input_data   : std_logic_vector (1 downto 0);
  signal output_data  : std_logic;

begin
  the_cv3_basic_logic: cv3_basic_logic
    port map (
      my_input => input_data,
      my_output => output_data
    );

  stimulus: process
    begin
      input_data <= "00";
      wait for 100ns;
      input_data <= "01";
      wait for 100ns;
      input_data <= "10";
      wait for 100ns;
      input_data <= "11";
      wait for 100ns;
      input_data <= "00";
      wait;
    end process stimulus;

end stim;
```

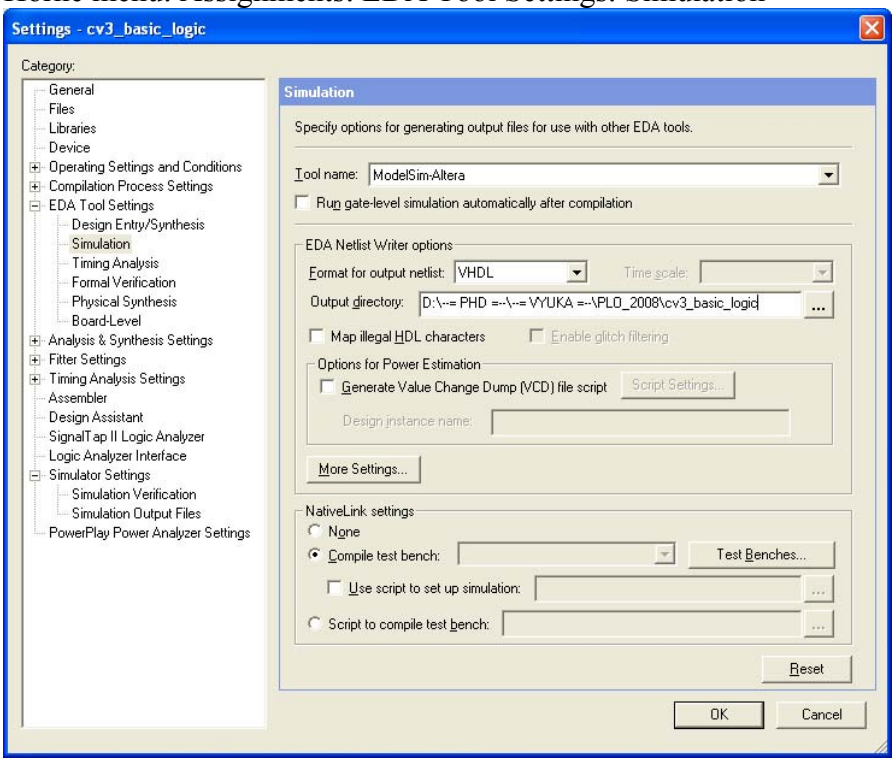
Súbor uložíme ako stim_cv3_basic_logic.vhdl

Nakonfigurujeme Quartus II pre priame spúšťanie simulácie v ModelSime:

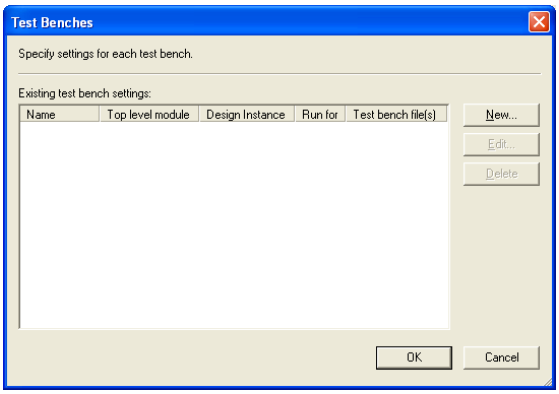
Zadáme cestu k modelsim.exe:
Horné menu: Tool: Options: General: EDA Tool Options
Dvojklikom na ModelSim-Altera (pre domácu inštaláciu) alebo na ModelSim (pre školskú inštaláciu) a nasledným kliknutím na „...“ zadáme cestu k modelsim.exe



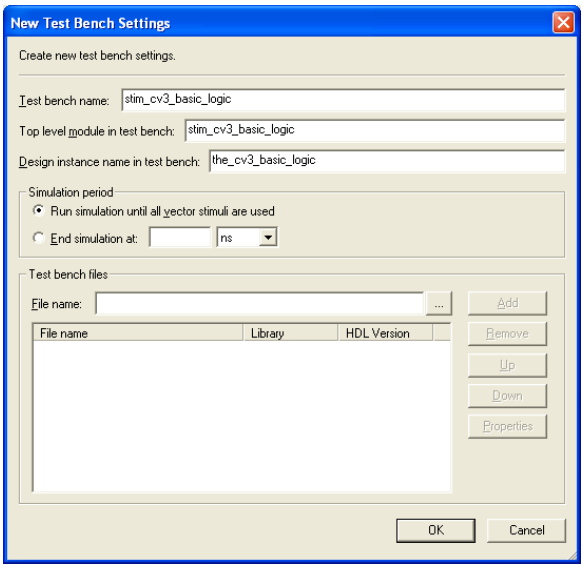
Nakonfigurujeme simuláciu:
Horné menu: Assignments: EDA Tool Settings: Simulation



V Tool name bude uvedený ModelSim-Altera (domáca inštalácia) alebo ModelSim (školská inštalácia)
V EDA Netlist Writer Options: Output Directory zadajte cestu k projektovému adresáru.
V NativeLink settings zaškrtnite Compile test bench a kliknite na tlačidlo Test Benches...

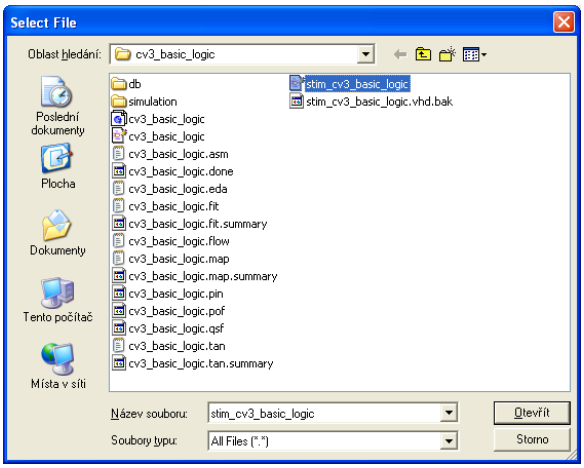


New...

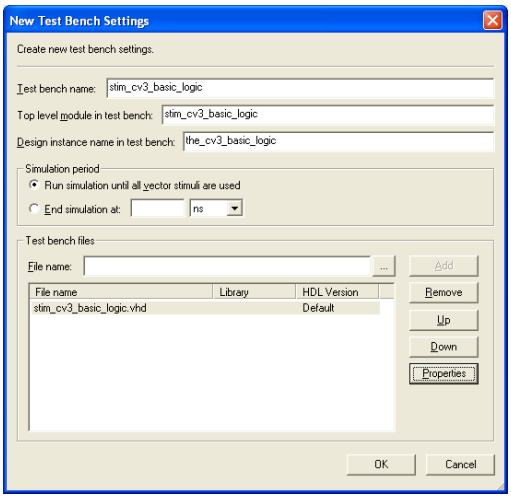


Test bench name: stim_cv3_basic_logic
Top level module in test bench: stim_cv3_basic_logic
Design instance name in test bench: the_cv3_basic_logic

Vložime test bench súbor kliknutím na ...

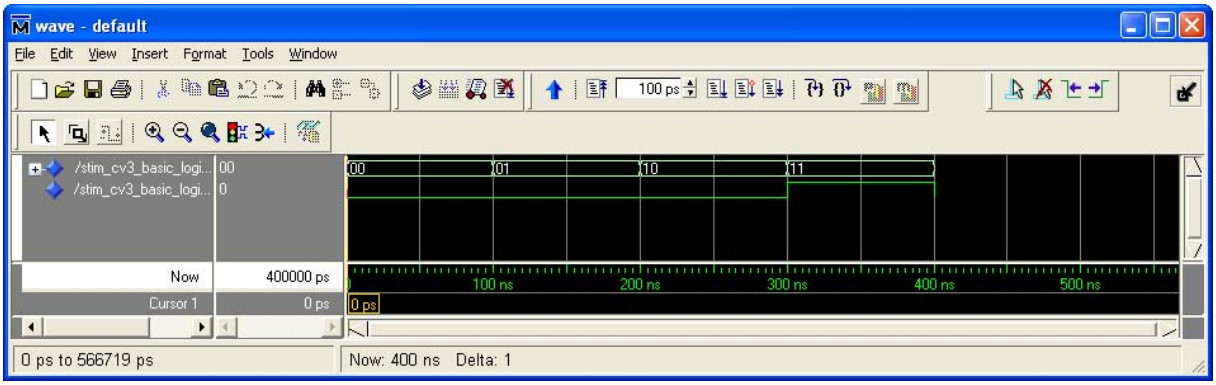


Kliknete na Add

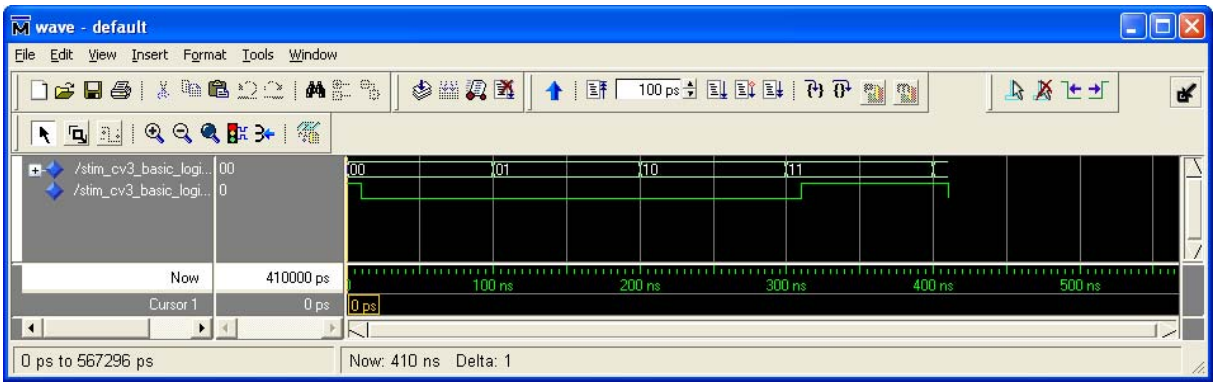


OK.
OK.
OK.

Prekompilujeme celý dizajn.
Spustíme funkčnú simuláciu: Horné menu: Tools: EDA Simulation Tool: Run EDA RTL Simulation



Prekompilujeme celý dizajn.
Spustíme časovú simuláciu: Horné menu: Tools: EDA Simulation Tool: Run EDA Gate Level Simulation



Dizajn nahrajte do dosky do dosky CPLD_KIT.

Námet na samostatnú prácu:

Realizujte funkčnú a časovú simuláciu v ModelSime prevodníka BCD na 7 segmentový displej. Projekt nahrajte do dosky.

Prevodník:

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

entity comb_unit is
port(
    data : in std_logic_vector (3 downto 0);
    seg : out std_logic_vector (6 downto 0)
    );
end comb_unit;

architecture a of comb_unit is
begin
  with data select
    seg <=
      "0000001" when "1111", -- 0
      "1001111" when "1110", -- 1
      "0010010" when "1101", -- 2
      "0000110" when "1100", -- 3
      "1001100" when "1011", -- 4
      "0100100" when "1010", -- 5
      "0100000" when "1001", -- 6
      "0001111" when "1000", -- 7
      "0000000" when "0111", -- 8
      "0000100" when "0110", -- 9
      "0001000" when "0101", -- A
      "1100000" when "0100", -- b
      "1110010" when "0011", -- c
      "1000010" when "0010", -- d
      "0110000" when "0001", -- e
      "0111000" when "0000", -- f
      "1111111" when others; -- else
end a;
```

Test Bench:

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

entity stim_comb_unit is
end stim_comb_unit;

architecture stim of stim_comb_unit is

  component comb_unit is
  port(
      data : in std_logic_vector (3 downto 0);
      seg : out std_logic_vector (6 downto 0)
      );
```

```
  end component;

  signal stim_data : std_logic_vector (3 downto 0);
  signal out_data  : std_logic_vector (6 downto 0);

begin
  the_comb_unit: comb_unit
    port map (
      data => stim_data,
      seg => out_data
    );

  stimulus: process
    begin
      stim_data <= "1111"; wait for 100ns;
      stim_data <= "1110"; wait for 100ns;
      stim_data <= "1101"; wait for 100ns;
      stim_data <= "1100"; wait for 100ns;
      stim_data <= "1011"; wait for 100ns;
      stim_data <= "1010"; wait for 100ns;
      stim_data <= "1001"; wait for 100ns;
      stim_data <= "1000"; wait for 100ns;
      stim_data <= "0111"; wait for 100ns;
      stim_data <= "0110"; wait for 100ns;
      stim_data <= "0101"; wait for 100ns;
      stim_data <= "0100"; wait for 100ns;
      stim_data <= "0011"; wait for 100ns;
      stim_data <= "0010"; wait for 100ns;
      stim_data <= "0001"; wait for 100ns;
      stim_data <= "0000";
      wait;
    end process stimulus;

end stim;
```