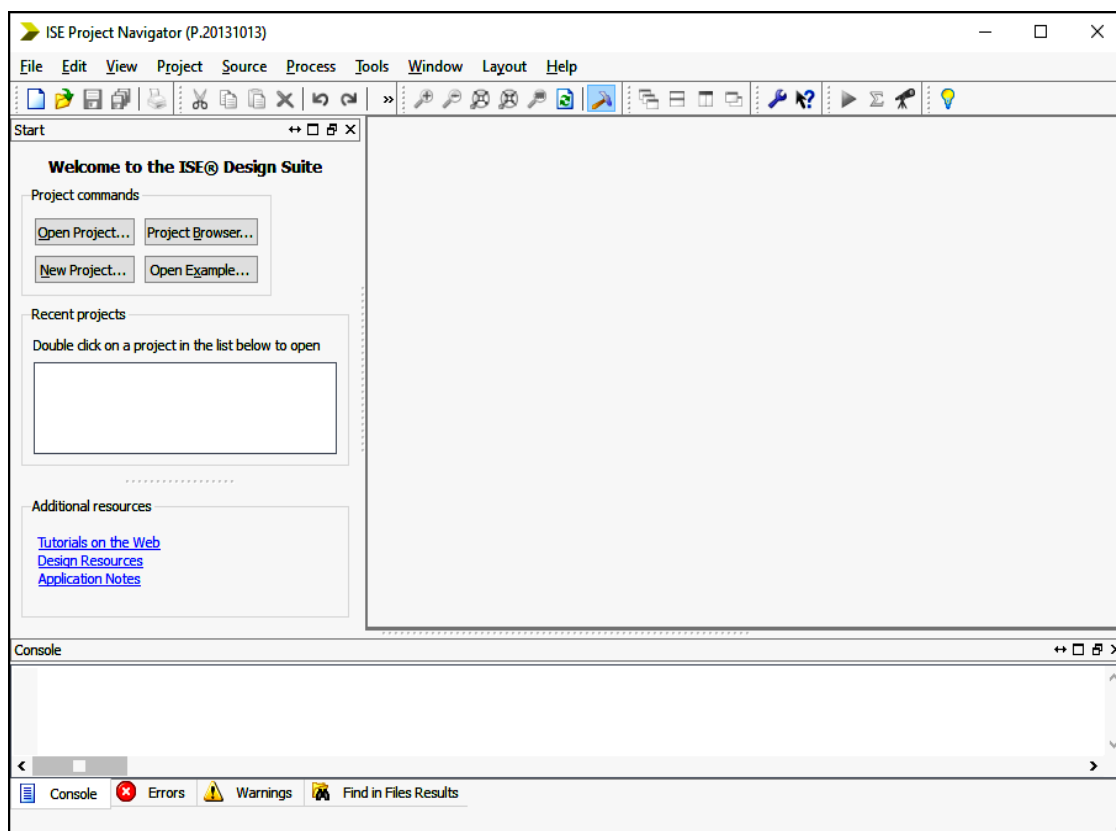


## TUTORIAL – wstępny projekt w języku VHDL

### Uruchomienie systemu projektowego ISE Design Suite 14.7

Na wstępie należy utworzyć katalog dla projektu...

System uruchamia się poprzez dwukrotne kliknięcie lewym klawiszem myszki na ikonie **ISE Design Suite**, lub wybierając z rozwijanego menu: **Start** → **Xilinx Design Tools** → **64-bit Project Navigator**. Po uruchomieniu programu pojawia się nawigator projektu **ISE Project Navigator**.



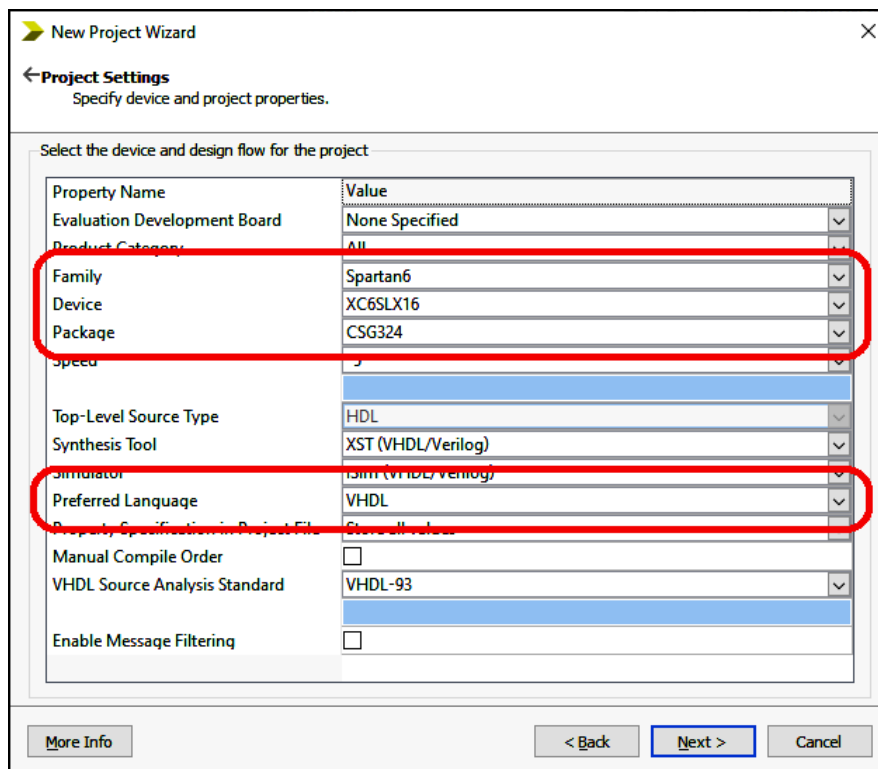
### Plik licencyjny

W czasie pierwszego uruchomienia systemu projektowego należy dołączyć plik licencyjny poprzez:

- **Help** → **Manage License...**,
- w kolejnym oknie **Xilinx License Configuration Manager** wybrać zakładkę **Manage Licenses**,
- nacisnąć **Load License** i wskazać plik licencyjny **Xilinx\_2016.03.lic**.  
( => staff(\oceanic)(P:) => szy..... => Xilinx\_2016.03.lic )

## Utworzenie nowego projektu typu HDL

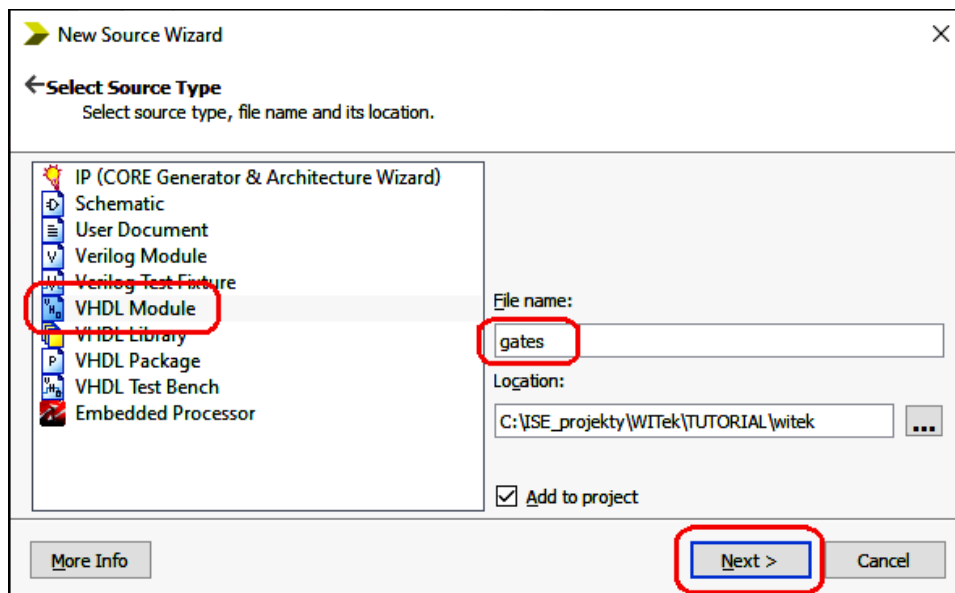
1. Wybrać **File** → **New Project...** uruchamiając aplikację **New Project Wizard**
2. W polu **Location:** ustawić utworzony folder dla projektu
3. W polu **Name:** wpisać nazwę projektu, np. **witek**
4. Sprawdzić czy w polu **Top-level source type:** jest wybrana opcja **HDL**
5. Nacisnąć przycisk **Next**
6. Zgodnie z posiadaną płytką testową wypełnić pola okna **Project Settings**, gdzie najważniejsze ustawienia to:
  - Family** → Spartan6
  - Device** → XC6SLX16
  - Package** → CSG324
  - Preferred Language** → VHDL



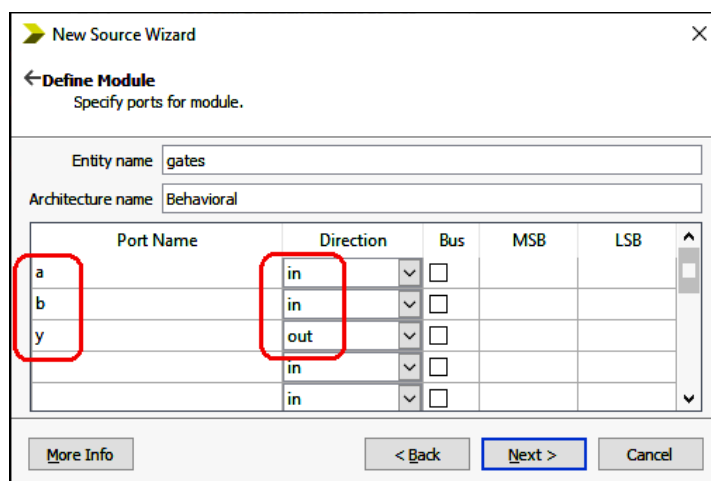
7. Nacisnąć przycisk **Next**
8. W oknie **Project Summary** nacisnąć **Finish**

## Opis działania układu w języku VHDL

1. Wybrać **Project** → **New Source...** uruchamiając **New Source Wizard**
2. W oknie **Select Source Type** zaznaczyć typ **VHDL Module**
3. W polu **File name:** wpisać nazwę **gates**
4. W polu **Location** powinien być wybrany folder projektu



5. Naciśnąć **Next**
6. W oknie **Define Module** wypełnić odpowiednie kolumny nazwami sygnałów (**Port Name**) i ich parametrami (**Direction**), jak poniżej



7. Naciśnąć **Next**
8. W oknie **Summary** nacisnąć **Finish**
9. W utworzonym pliku **gates.vhd** opisać działanie układu jak poniżej (można usunąć „zielony” tekst :), zapisać plik

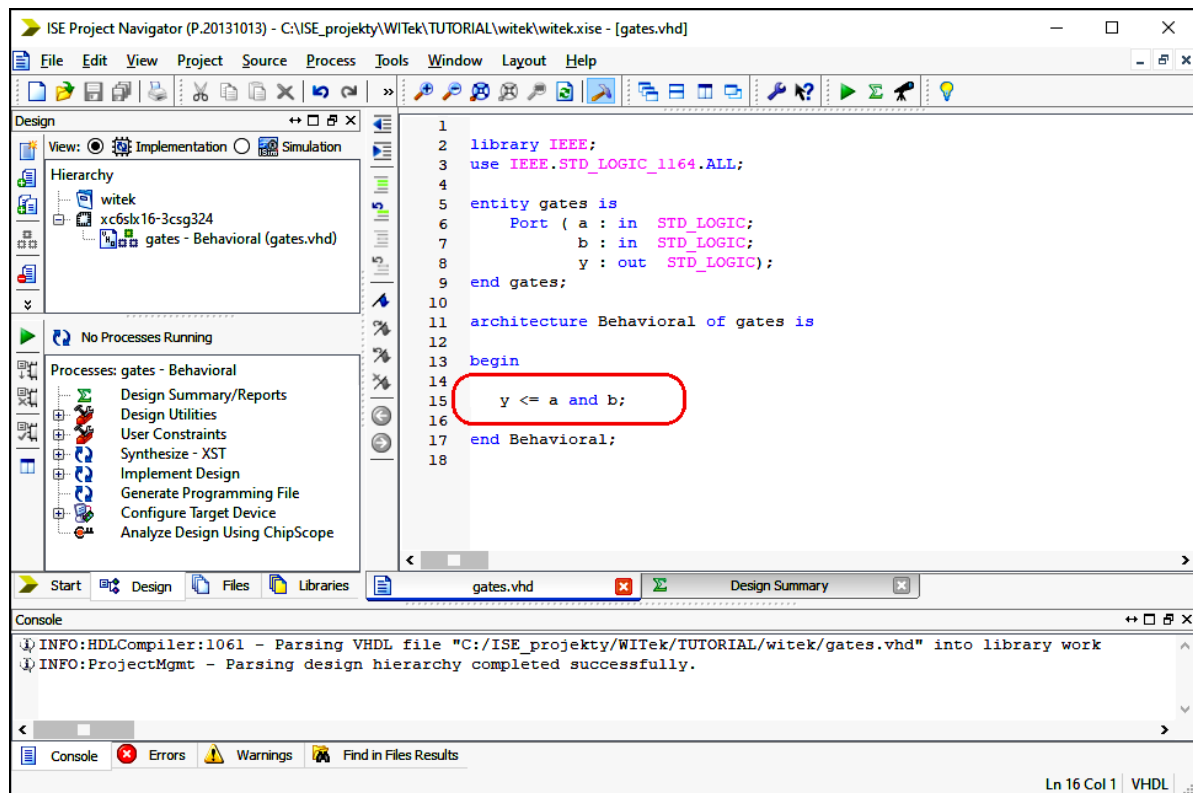
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity gates is
    port( a : in std_logic;
          b : in std_logic;
          y : out std_logic );
end gates;

architecture Behavioral of gates is
begin

    y <= a and b;

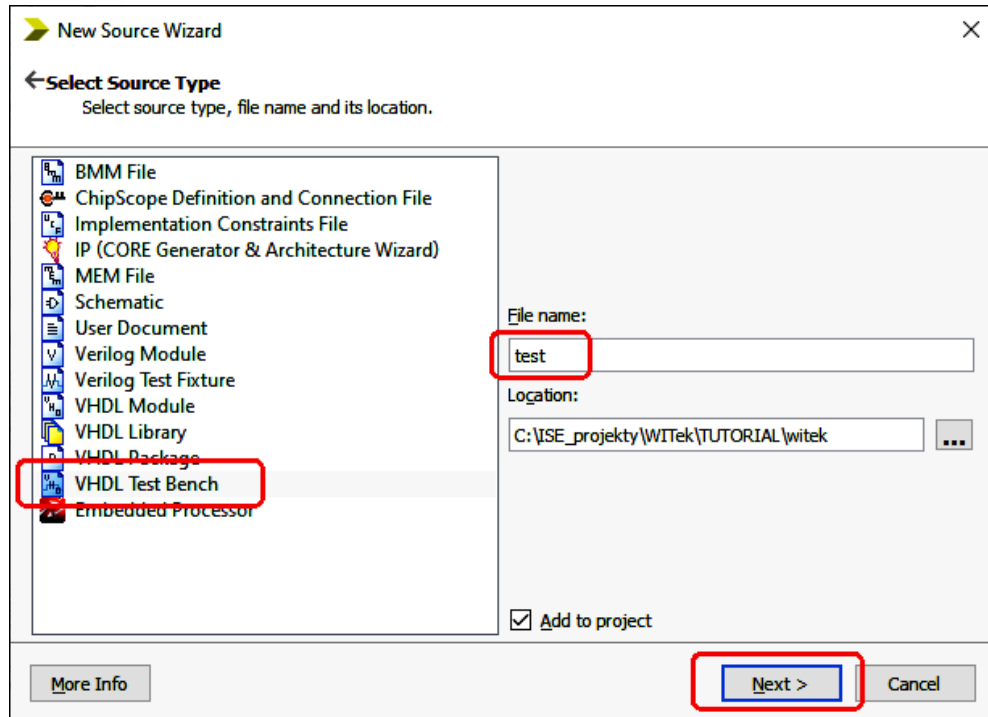
end Behavioral;
```



10. W celu sprawdzenia poprawności kodu w języku VHDL należy w obszarze **Hierarchy** zaznaczyć plik **gates – Behavioral (gates.vhd)**, następnie w polu **Processes** rozwinąć wiersz **Synthesize – XST** i uruchomić **Check Syntax**, poprawić ewentualne błędy...

## Symulacja działania układu

1. Wybrać **Project** → **New Source...** uruchamiając **New Source Wizard**
2. W oknie **Select Source Type** zaznaczyć typ **VHDL Test Bench**



3. W polu **File name**: wpisać nazwę **test**
4. W polu **Location** powinien być wybrany folder projektu...
5. Nacisnąć **Next**
6. W kolejnym oknie **Associate Source** powinien być zaznaczony **gates**, nacisnąć **Next**
7. W oknie **Summary** nacisnąć **Finish**
8. W utworzonym module testowym **test.vhd** można zastosować przykładowy algorytm testowy jak poniżej. Moduł ustala stany na wejściach **a** i **b**, którymi są kolejne kombinacje binarne zmieniane co 50 ns. Ostatnia instrukcja **wait** zatrzymuje symulację.

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test IS
END test;

ARCHITECTURE behavior OF test IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT gates
    PORT( a : IN  std_logic;
          b : IN  std_logic;
          y : OUT std_logic );
    END COMPONENT;

    -- Inputs
    signal a : std_logic := '0';
    signal b : std_logic := '0';
  
```

```
-- Outputs
signal y : std_logic;

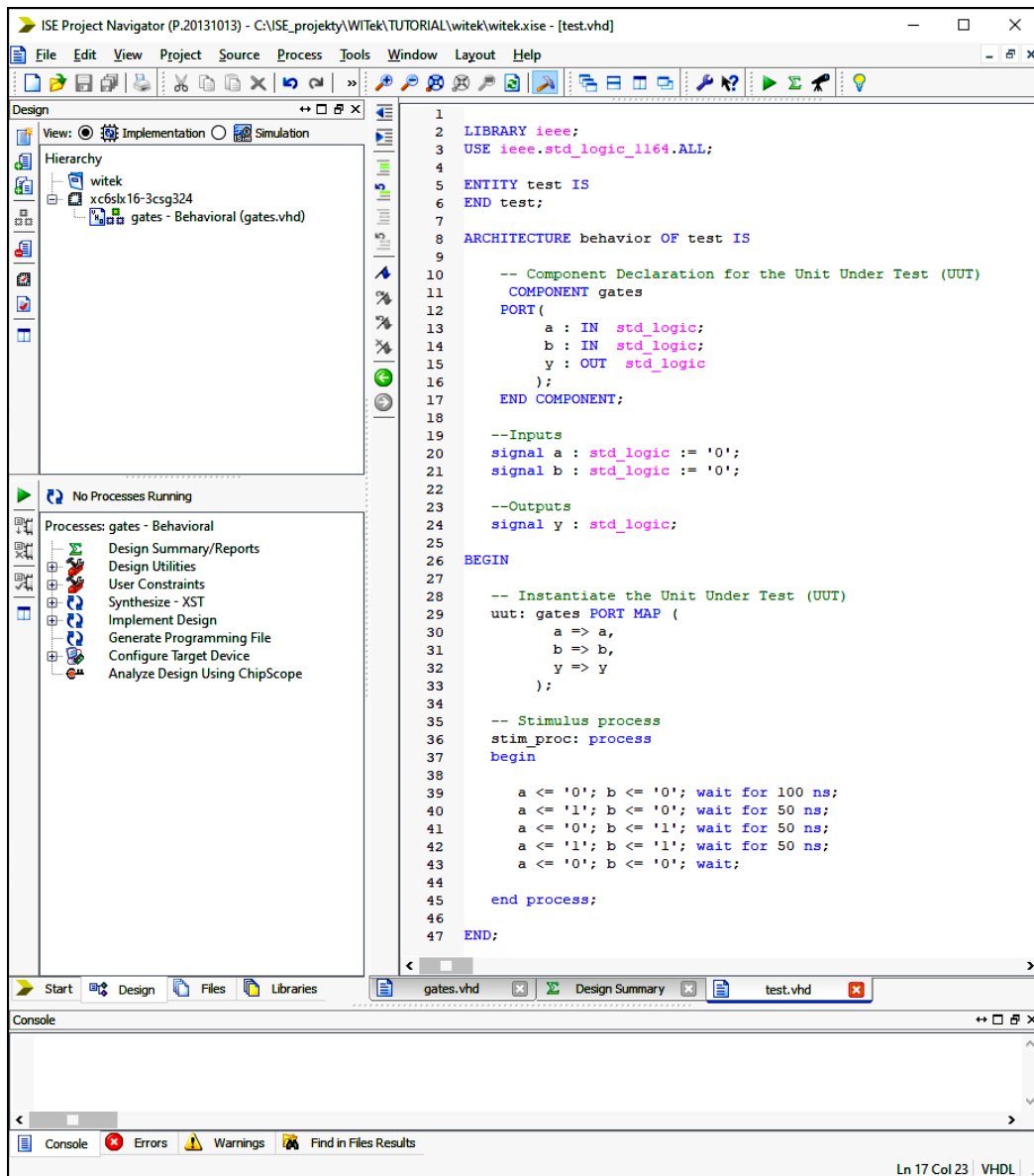
BEGIN
-- Instantiate the Unit Under Test (UUT)
 uut: gates PORT MAP ( a => a, b => b, y => y );

-- Stimulus process
 stim_proc: process
 begin

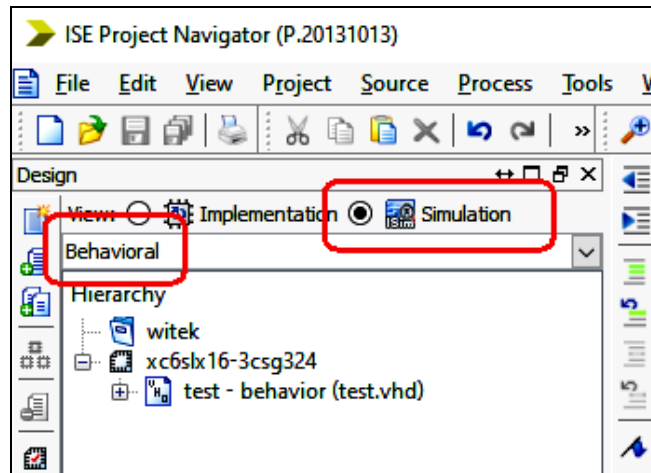
    a<='0'; b<='0'; wait for 100 ns;
    a<='0'; b<='1'; wait for 50 ns;
    a<='1'; b<='0'; wait for 50 ns;
    a<='1'; b<='1'; wait for 50 ns;
    a<='0'; b<='0'; wait;

 end process;

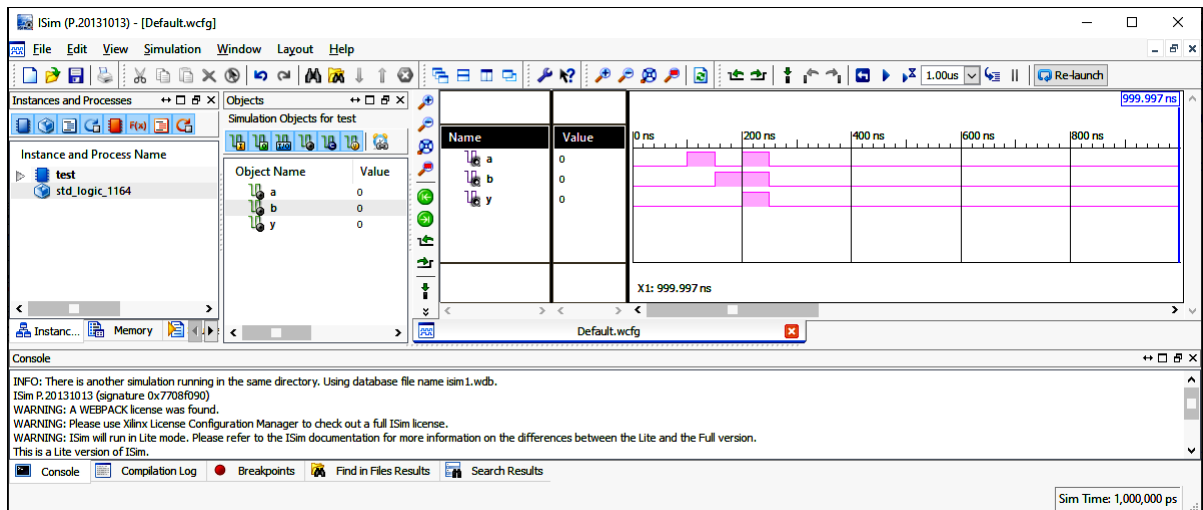
END;
```



9. Następnie w polu **View:** zaznaczyć **Simulation**, a w rozwijanym menu wybrać **Behavioral**

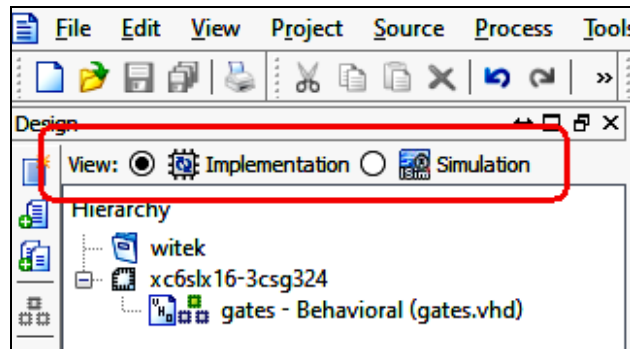


10. W obszarze **Hierarchy** zaznaczyć plik **test – behavior (test.vhd)**  
 11. W polu **Processes** rozwinąć **ISim Simulator** i uruchomić **Behavioral Check Syntax**,  
 poprawić ewentualne błędy...  
 12. Uruchomić symulację poprzez **Simulate Behavioral Model**  
 13. Wynik symulacji zaobserwować w postaci przebiegów sygnałów...

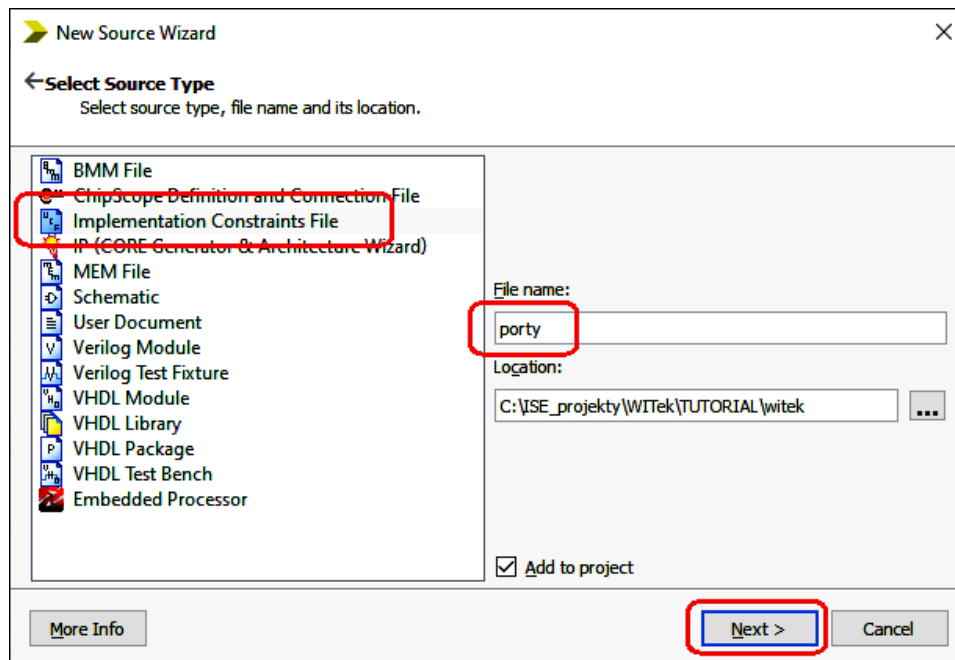


## Implementacja projektu w układzie FPGA

1. W polu **View:** wybrać **Implementation** (wyjście z symulacji, powrót do projektowania...)



2. Wybrać **Project** → **New Source...** uruchamiając **New Source Wizard**
3. W oknie **Select Source Type** zaznaczyć typ **Implementation Constrains File**



4. W polu **File name:** wpisać nazwę **porty**
5. W polu **Location** powinien być wybrany folder projektu
6. Nacisnąć **Next**, a następnie **Finish**
7. Do projektu został dołączony plik tekstowy **porty.ucf** (rozwinąć wiersz **gates – Behavioral (...)** w polu **Hierarchy**)
8. Jako zawartość pliku **porty.ucf** wpisać poniższy zapis, będący przyporządkowaniem sygnałów projektu do końcówek układu scalonego (zgodnie z posiadaną płytką testową), zapisać i zamknąć plik

```
NET "a" LOC="T10";
NET "b" LOC="T9";
NET "y" LOC="U16";
```

9. W obszarze **Hierarchy** zaznaczyć wiersz **gates – Behavioral (...)**
10. Następnie w polu **Processes** uruchomić **Generate Programming File**

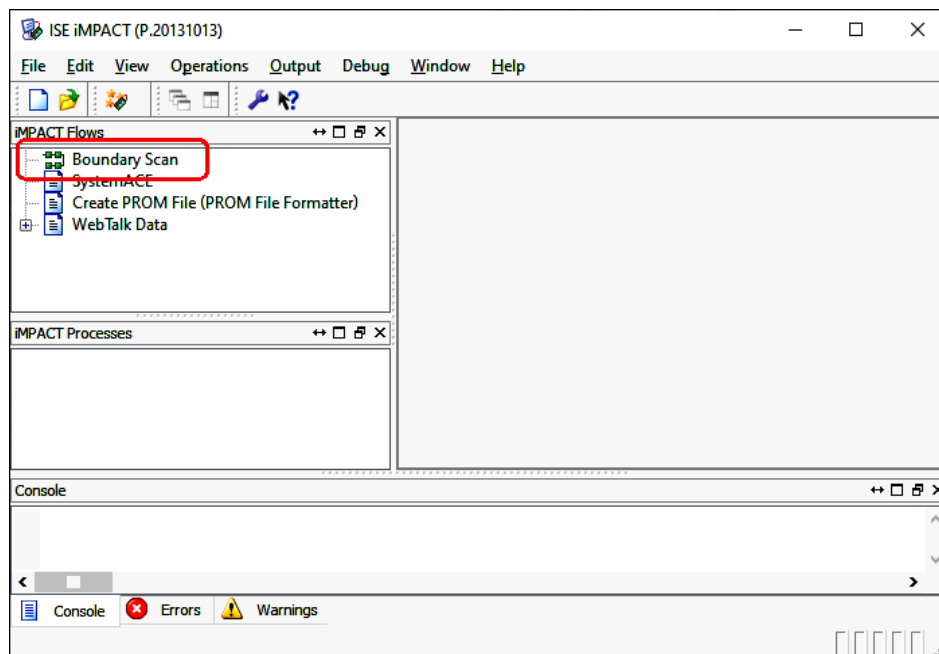


## Uwaga!

Podczas pierwszej implementacji po uruchomieniu systemu projektowego **ISE Design Suite** należy ponownie uruchomić **Generate Programming File**, wybierając prawy przycisk myszki **Rerun All**. Taka operacja wynika z błędnego przypisywania końcówek układu scalonego do sygnałów projektu przy pierwszej implementacji, ale po ponownej re-implementacji problem „znika”...

## Konfiguracja układu FPGA

1. Sprawdzić podłączenie płytki testowej!
2. Następnie w polu **Processes** rozwinąć wiersz **Configure Target Device** i uruchomić **Manage Configuration Project (iMPACT)**, jeśli pojawi się ostrzeżenie – nacisnąć **OK**
3. Pojawia się aplikacja **ISE iMPACT**, w polu **iMPACT Flows** uruchomić **Boundary Scan**



4. Następnie wybrać **File** → **Initialize Chain**
5. Jeśli pojawi się okno **Assign New Configuration Files Query Dialog**, to nacisnąć **Yes**
6. Następnie pojawia się okno **Assign New Configuration File**, w którym należy wskazać katalog projektu i wybrać plik **gates.bit**, nacisnąć **Open**
7. W kolejnym okienku **Attach SPI or BPI PROM** nacisnąć **No**
8. W oknie **Device Programming Properties** -... zatwierdzić proponowane opcje konfigurowania układów FPGA i PROM poprzez **OK**
9. Następnie nacisnąć prawym klawiszem myszki na symbolu układu FPGA i z listy poleceń wybrać **Program**. Konfigurowanie w toku...
10. Sprawdzić praktycznie działanie projektu...

## HARMONOGRAM LABORATORIUM

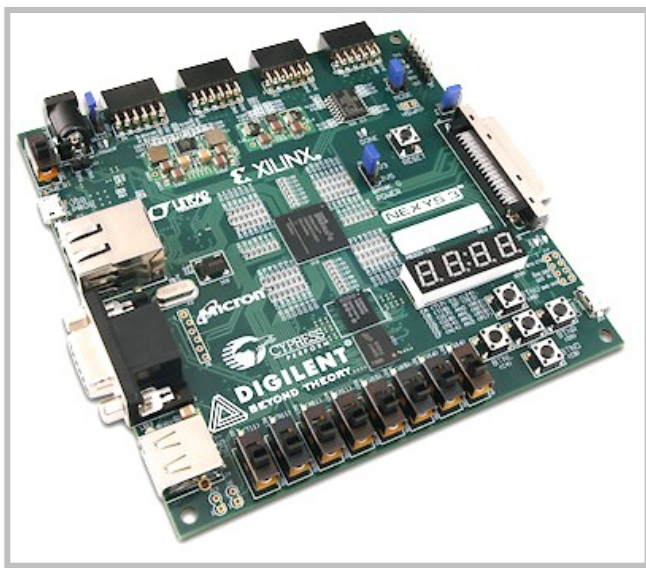
LAB 1	<ol style="list-style-type: none"> <li>1) Projekt wstępny z instrukcji laboratoryjnej.</li> <li>2) Projekt zawierający bramki <math>n</math>-wejściowe typu AND, OR oraz XOR.</li> <li>3) Zaprojektować konwerter 3-bitowego kodu binarnego na kod „1 z 8” ( na wyjściu aktywny stan 0 lub 1 ).</li> <li>4) Zaprojektować sumator dwóch liczb 4-bitowych.</li> </ol>
LAB 2	<ol style="list-style-type: none"> <li>1) Zaprojektować 27-bitowy licznik binarny z kasowaniem asynchronicznym. ( zastosować sygnał zegarowy z płytki testowej, osiem najbardziej znaczących bitów licznika wyprowadzić na diody LED )</li> <li>2) Zaprojektować 8-bitowy licznik Johnsona. ( jako dzielnik częstotliwości sygnału zegarowego zastosować <math>N</math>-bitowy licznik binarny, dobrać <math>N</math> )</li> <li>3) Zaprojektować 8-bitowy licznik pierścieniowy typu „krążąca jedynka” . ( zastosować dzielnik częstotliwości... )</li> <li>4) Zmodyfikować punkt 3) tak, aby za pomocą suwaka wybierać pomiędzy „krążącą jedynką” a „krążącym zerem”</li> </ol>
LAB 3	Indywidualne projekty z użyciem bloków kombinacyjnych i sekwencyjnych...

## Nexys 3

Family: **Spartan 6**

Device: **XC6SLX16**

Package: **CSG324**



Diody LED świecą po podaniu poziomu H na odpowiednią końcówkę układu FPGA:

Dioda LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
Końcówka układu FPGA	T11	R11	N11	M11	V15	U15	V16	U16

Naciśnięcie przycisku monostabilnego ustala poziom H na odpowiedniej końcówce układu FPGA:

Przycisk	BTNL	BTNR	BTNU	BTND	BTNS
Końcówka układu FPGA	C4	D9	A8	C9	B8

Ośiem przełączników suwakowych umożliwia ustalenie poziomu H (pozycja UP) lub poziomu L (pozycja DOWN) na końcówkach układu FPGA:

Przełącznik	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
Końcówka układu FPGA	T5	V8	U8	N8	M8	V9	T9	T10

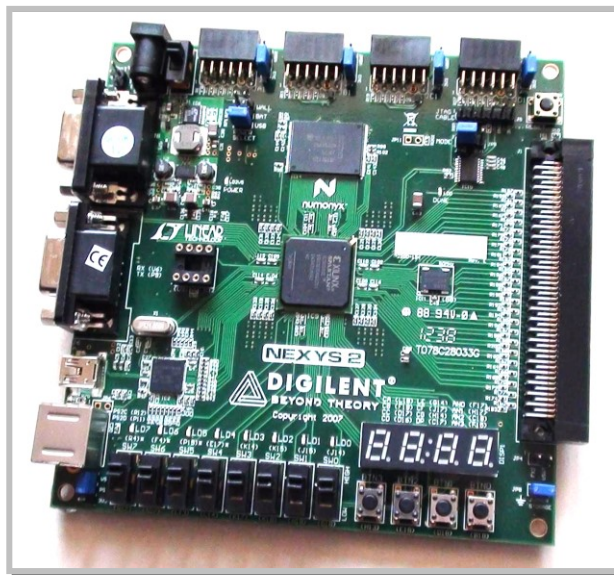
Generator dostarcza sygnał o częstotliwości 100 MHz do układu poprzez końcówkę V10.

## Nexys 2

Family: **Spartan 3E**

Device: **XC3S500E**

Package: **FG320**



Diody LED świecą po podaniu poziomu H na odpowiednią końcówkę układu FPGA:

Dioda LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
Końcówka układu FPGA	R4	F4	P15	E17	K14	K15	J15	J14

Naciśnięcie przycisku monostabilnego ustala poziom H na odpowiedniej końcówce układu FPGA:

Przycisk	BTN3	BTN2	BTN1	BTN0
Końcówka układu FPGA	H13	E18	D18	B18

Osiem przełączników suwakowych umożliwia ustalenie poziomu H (pozycja UP) lub poziomu L (pozycja DOWN) na końcówkach układu FPGA:

Przełącznik	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
Końcówka układu FPGA	R17	N17	L13	L14	K17	K18	H18	G18

Generator kwarcowy dostarcza sygnał o częstotliwości 50 MHz do układu poprzez końcówkę B8.

## Spartan-3 Starter Kit

Family: **Spartan3**

Device: **XC3S200**

Package: **FT256**



Diody LED świecą po podaniu poziomu H na odpowiednią końcówkę układu FPGA:

Dioda LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
Końcówka układu FPGA	P11	P12	N12	P13	N14	L12	P14	K12

Naciśnięcie przycisku monostabilnego ustala poziom H na odpowiedniej końcówce układu FPGA:

Przycisk	BTN3	BTN2	BTN1	BTN0
Końcówka układu FPGA	L14	L13	M14	M13

Osiem przełączników suwakowych umożliwia ustalenie poziomu H (pozycja UP) lub poziomu L (pozycja DOWN) na końcówkach układu FPGA:

Przełącznik	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
Końcówka układu FPGA	K13	K14	J13	J14	H13	H14	G12	F12

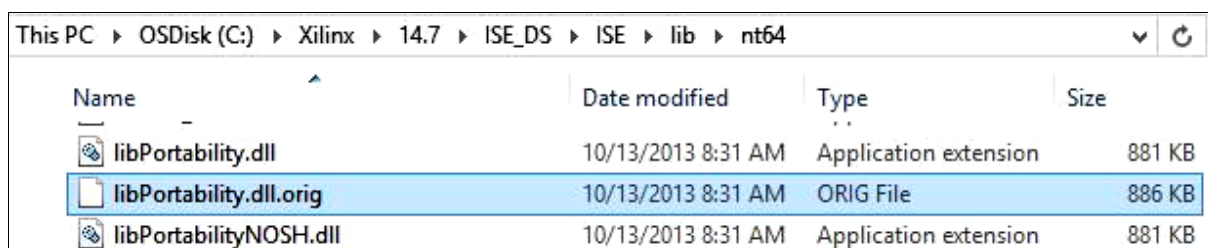
Generator kwarcowy dostarcza sygnał o częstotliwości 50 MHz do układu poprzez końcówkę T9.

## DODATEK – gdyby system projektowy nie działał poprawnie

**Dla systemu operacyjnego Windows 10 należy przed uruchomieniem systemu projektowego wykonać poniższe zmiany:**

### ISE 14.7 64-bit - Turning off SmartHeap:

- 1) Navigate to the following ISE install directory: <install\_path>\Xilinx\14.7\ISE\_DS\ISE\lib\nt64\
- 2) Rename the file "libPortability.dll" to "libPortability.dll.orig".
- 3) Copy the "libPortabilityNOSH.dll" file to the same folder, and rename it to "libPortability.dll".



This PC > OSDisk (C:) > Xilinx > 14.7 > ISE_DS > ISE > lib > nt64			
Name	Date modified	Type	Size
libPortability.dll	10/13/2013 8:31 AM	Application extension	881 KB
libPortability.dll.orig	10/13/2013 8:31 AM	ORIG File	886 KB
libPortabilityNOSH.dll	10/13/2013 8:31 AM	Application extension	881 KB

- 4) Repeat steps 1-3 in the following folder: <install\_path>\Xilinx\14.7\ISE\_DS\common\lib\nt64\

The above steps substitute the original "libPortability.dll" with a "libPortability.dll" file that has SmartHeap disabled, the NOSmartHeap (NOSH) version.

This does not negatively impact the operation of the tools, and should successfully work around the ISE 14.7 crash documented above.