

Loginių įrenginių sintezė ir modeliavimas

(Xilinx Vivado 2021.1 aplinka)

Turinys

1. Darbo tikslas.....	1
2. Darbo priemonės	1
3. Projekto sukūrimas Vivado aplinkoje	1
3.1. Projekto sukūrimas.....	1
3.2. Loginio įrenginio aprašo VHDL kalba įtraukimas	5
3.3. Loginių įrenginių aprašų architektūros	10
3.4. Virtualaus bandymų stendo skirtą įrenginio modeliavimui sukūrimas	11
3.5. Įrenginio modeliavimas.....	17
3.5.1. Modeliavimo paleidimas	17
3.5.2. Modeliavimo išjungimas	18
3.6. Stimulų generavimas simulatoriaus aplinkoje	19
3.7. Stimulų aprašų generavimas bandymų stende, panaudojant Vivado šablonus.....	23
4. Užduotys.....	26
5. Ataskaita ir gynimas.....	26
6. Papildomi šaltiniai.....	27

1. Darbo tikslas

1. Įgyti gebėjimus projektuoti ir modeliuoti loginius įrenginius (angl. *logic devices*) Vivado paketu,
2. Tobulinti loginių įrenginių aprašymo VHDL kalba įgūdžius,
3. Išmokti skurti virtualiuosius bandymo stendus VHDL kalba.

2. Darbo priemonės

Aprašyme naudojama programinė įranga Vivado 2021.1

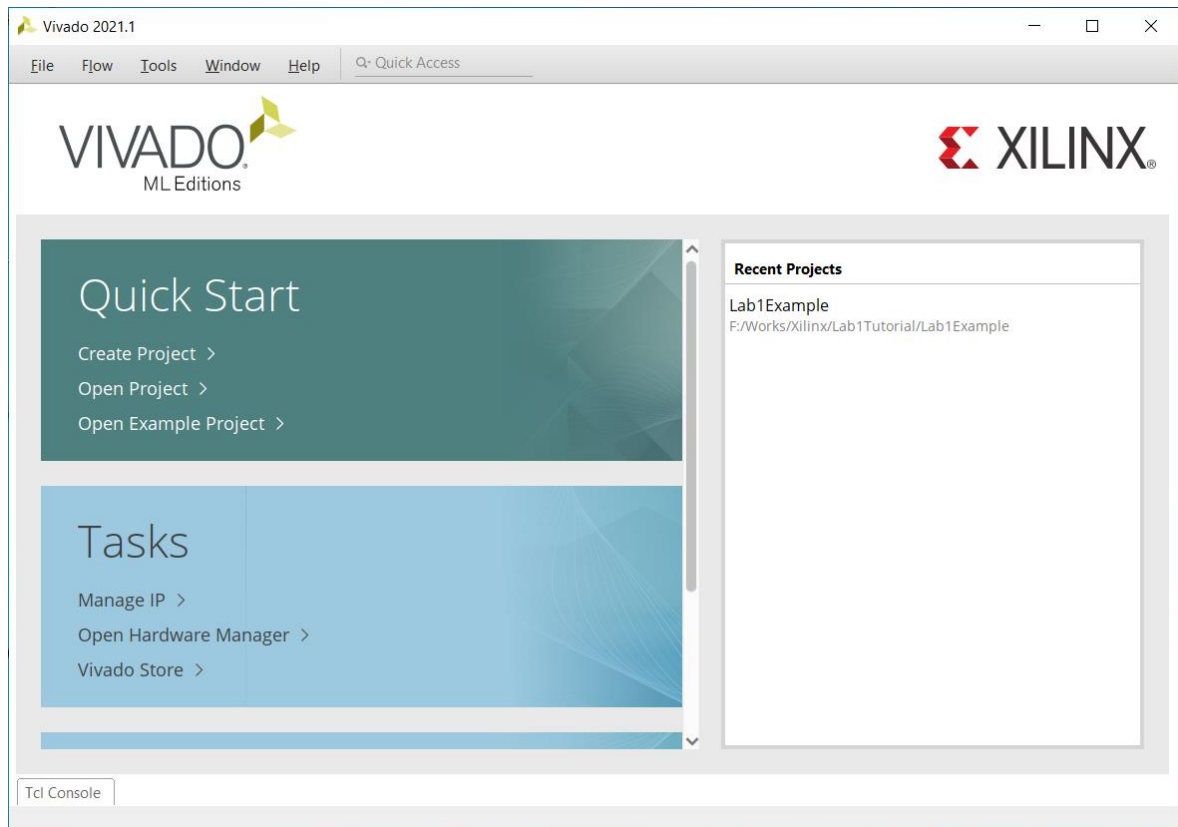


3. Projekto sukūrimas Vivado aplinkoje

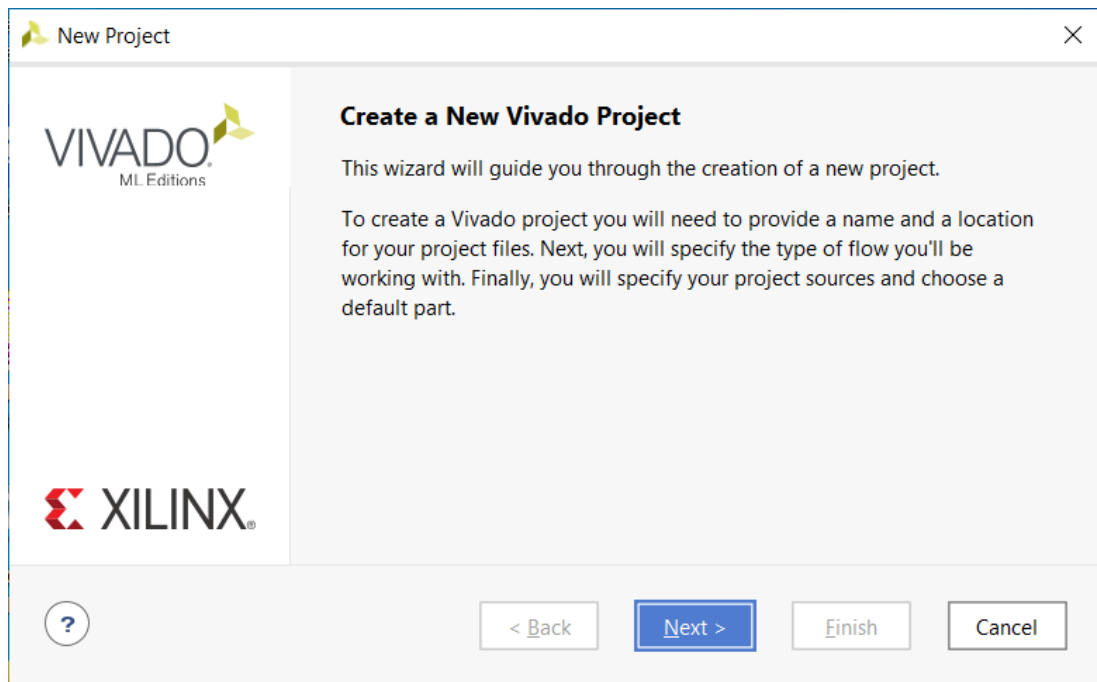
Tiek visą įrenginį, tiek atskirus jo komponentus galima aprašyti naudojant grafinį aprašą (*schematic*) arba HDL kalbą (*VHDL* ar *Verilog*). HDL kalba taip pat naudojama, kuriant virtualius bandymo stendus (*TestBench*), skirtus projektuojamo įrenginio modeliavimui..

3.1. Projekto sukūrimas

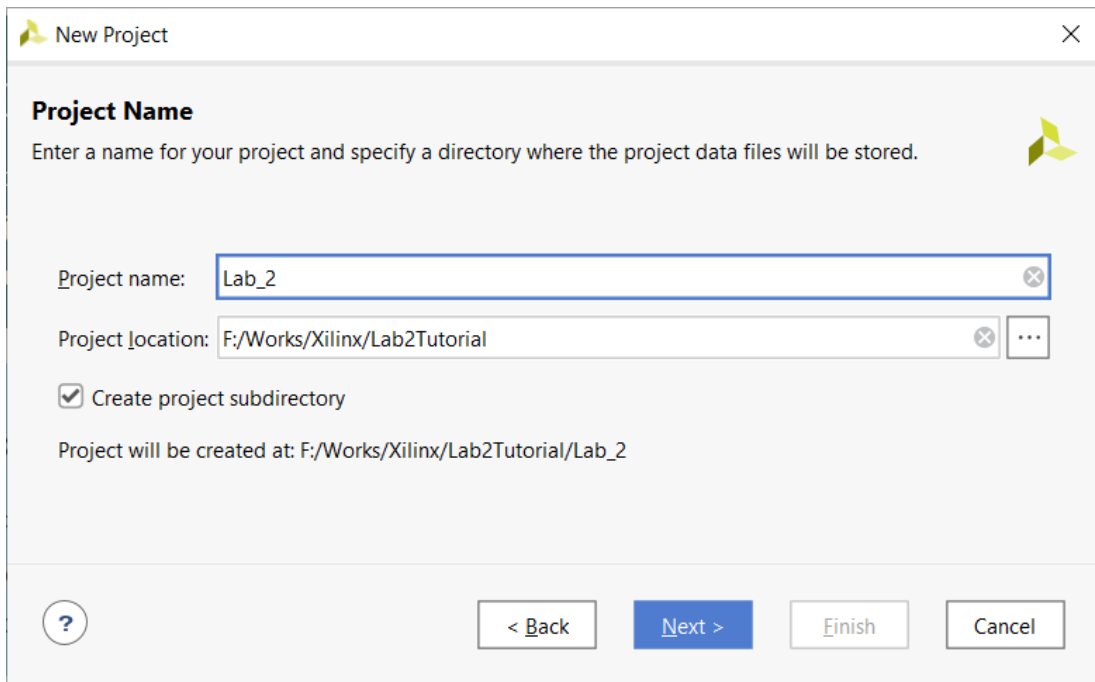
Paleidus *Vivado* paketą, pasirinkite meniu punktą **File/Project/New...** arba iš **Quick Start** meniu pasirinkite **Create Project** > punktą. Tinkamam projekto sukūrimui tęsiame nuosekliai atlikdami pasirinkimus žemiau parodytuose dialogo languose.



1 pav. Pradinis Vivado langas



2 pav. Naujo projekto kūrimas



New Project

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

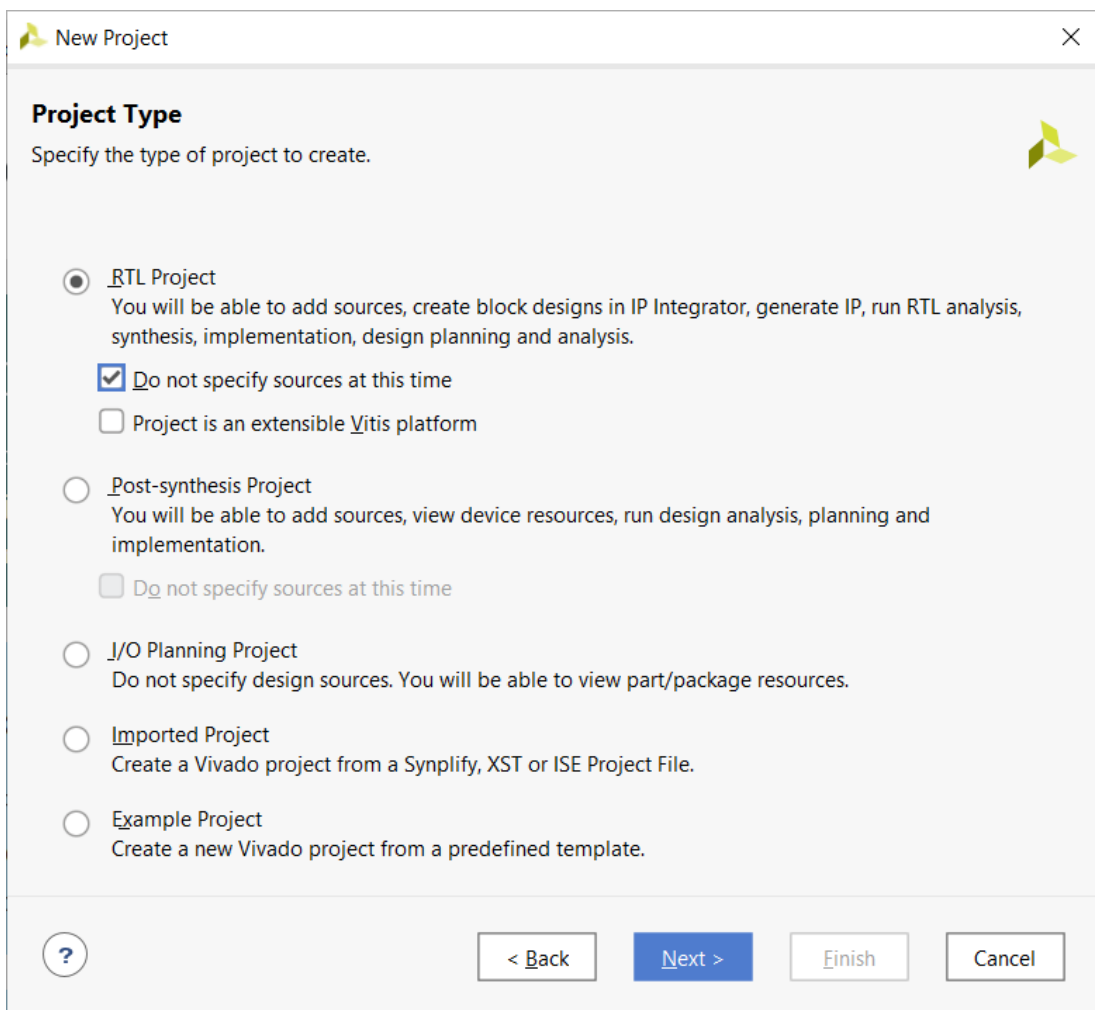
Project name:

Project location:

☒ Create project subdirectory

Project will be created at: F:/Works/Xilinx/Lab2Tutorial/Lab_2

3 pav. Kuriamo projekto išsaugojimas



New Project

Project Type
Specify the type of project to create.

☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☒ Do not specify sources at this time
☐ Project is an extensible Vitis platform

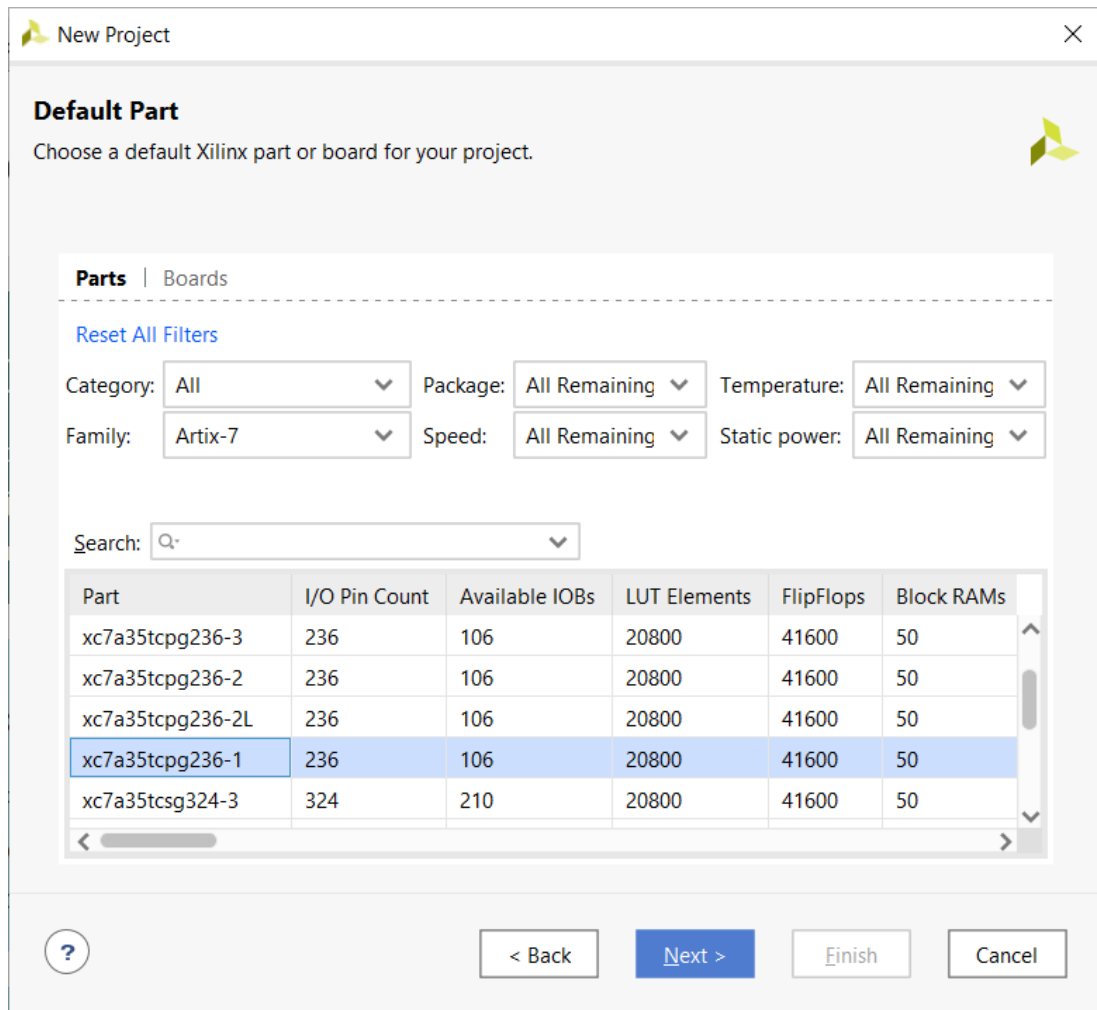
☐ **Post-synthesis Project**
You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.

4 pav. Projekto tipo pasirinkimas



New Project

Default Part
Choose a default Xilinx part or board for your project.

Parts | Boards

[Reset All Filters](#)

Category: All Package: All Remaining Temperature: All Remaining
Family: Artix-7 Speed: All Remaining Static power: All Remaining

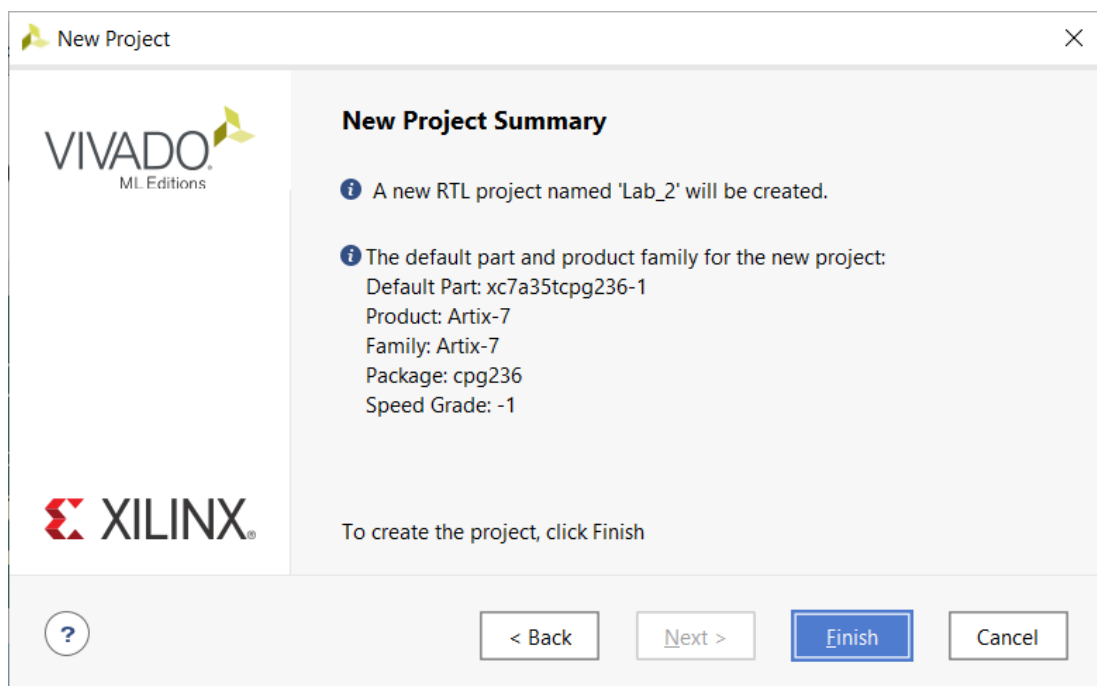
Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs
xc7a35tcpg236-3	236	106	20800	41600	50
xc7a35tcpg236-2	236	106	20800	41600	50
xc7a35tcpg236-2L	236	106	20800	41600	50
xc7a35tcpg236-1	236	106	20800	41600	50
xc7a35tcsg324-3	324	210	20800	41600	50

Navigation: ? < Back Next > Finish Cancel

5 pav. Planuojamo naudoti Artix-7 šeimos komponento pasirinkimas

Svarbu nurodyti tinkamus duomenis, kurie pateikti ant *Basys-3* maketo esančio *ARTIX-7 FPGA* elemento.



New Project

VIVADO
ML Editions

XILINX

New Project Summary

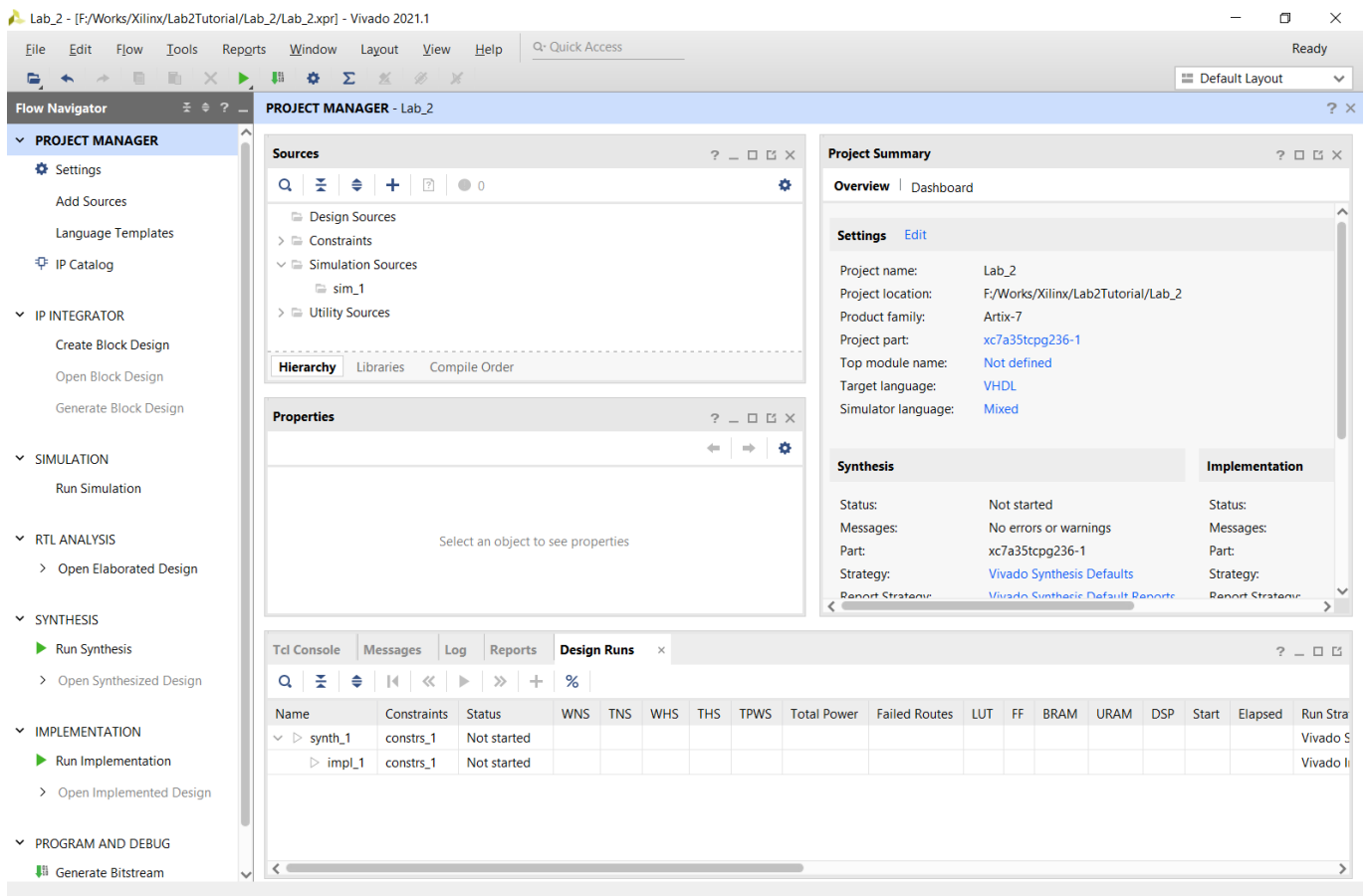
- A new RTL project named 'Lab_2' will be created.
- The default part and product family for the new project:
Default Part: xc7a35tcpg236-1
Product: Artix-7
Family: Artix-7
Package: cpg236
Speed Grade: -1

To create the project, click Finish

Navigation: ? < Back Next > Finish Cancel

6 pav. Projekto kūrimo pabaiga

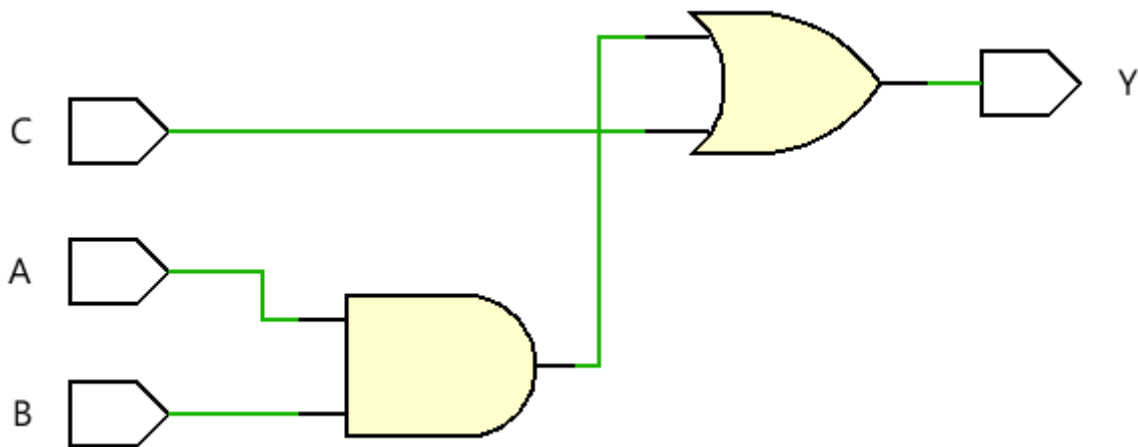
Projekto sukūrimas užbaigiamas aukščiau parodytame lange, paspaudus *Finish*.



7 pav. Sukurtas projekto vaizdas Vivado aplinkoje

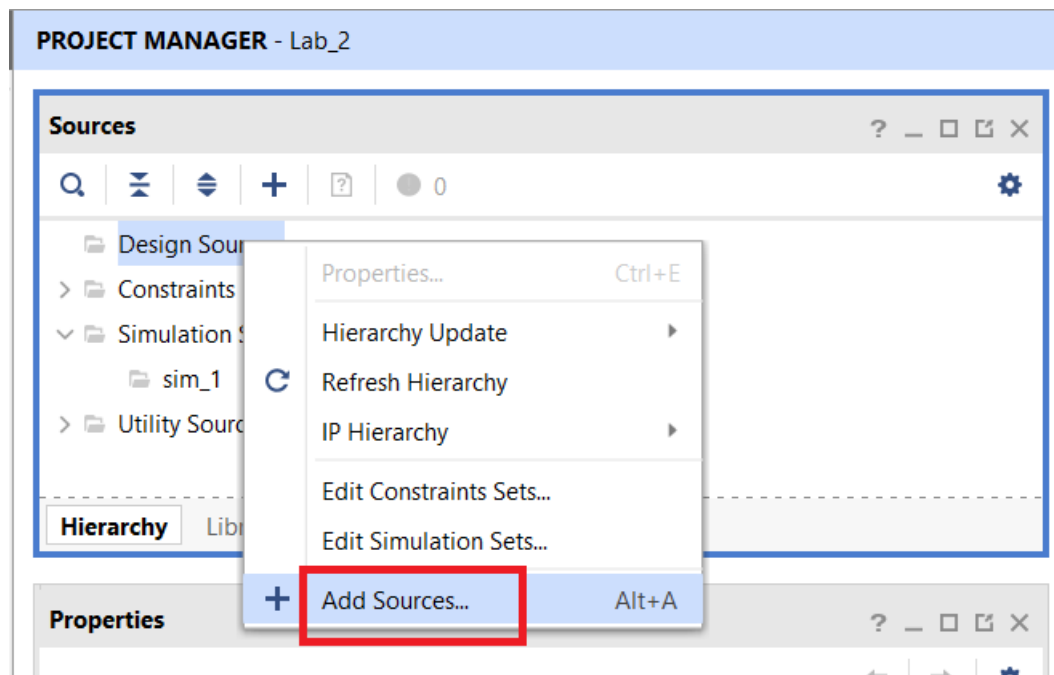
3.2. Loginio įrenginio aprašo VHDL kalba įtraukimas

Panaudodami VHDL kalbos aprašą, sukursime paprastą loginį įrenginį, kurio schema pavaizduota žemiau.

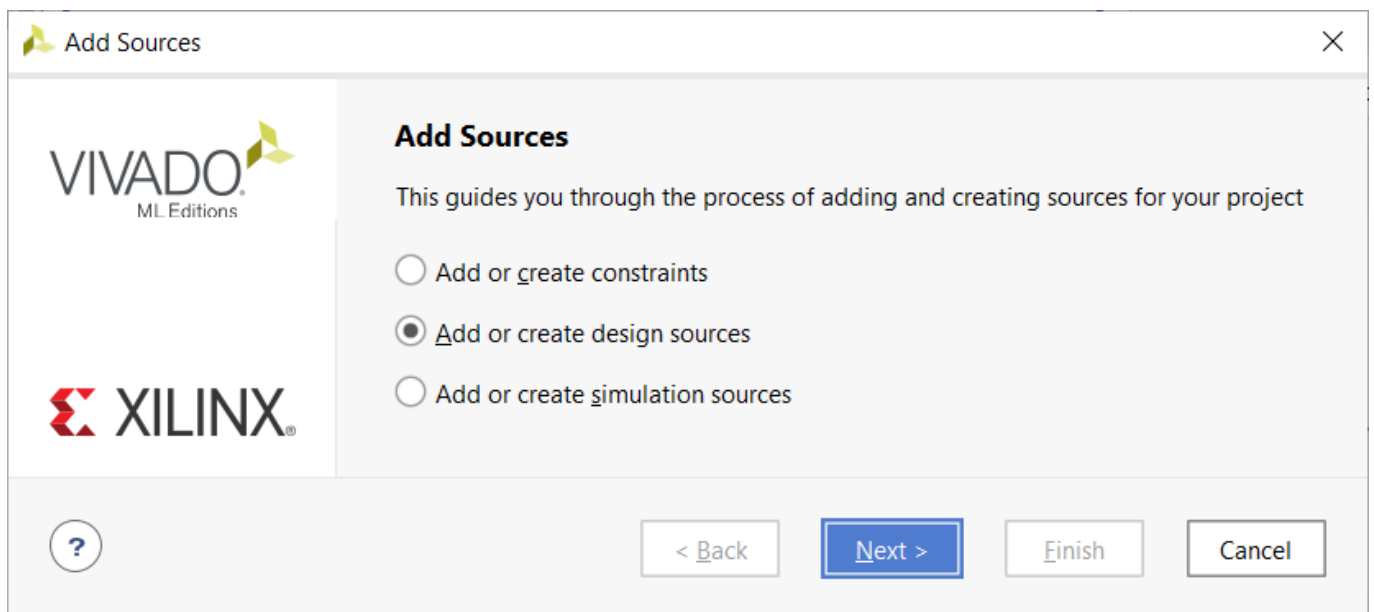


8 pav. Kuriamo loginio įrenginio schema

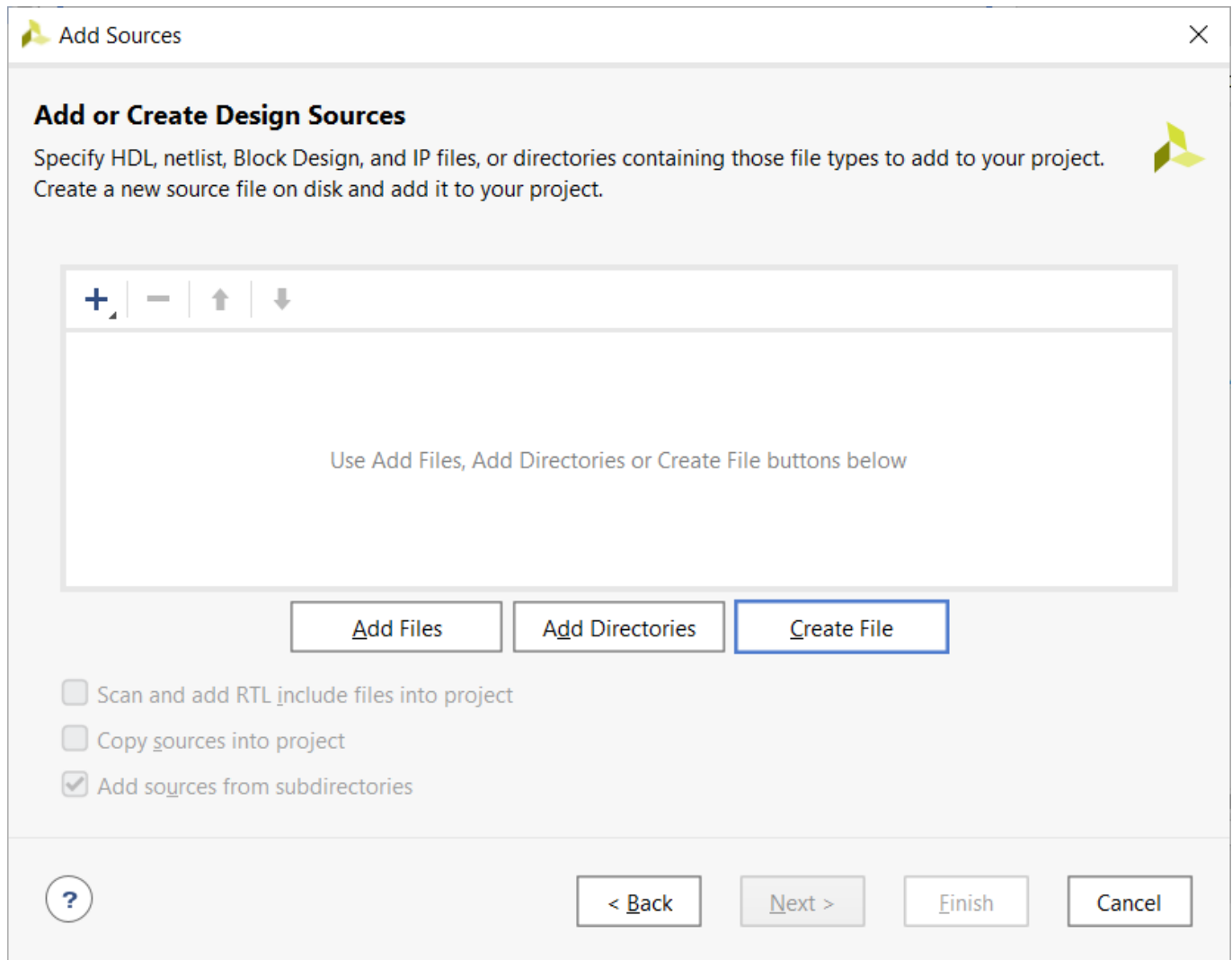
Naujo VHDL objekto aprašo failas projekte sukuriamas pasirinkus **Design Sources** iškrentančiame meniu pasirinkus punktą **Add Sources**.



9 pav. VHDL modulio įtraukimas į projektą

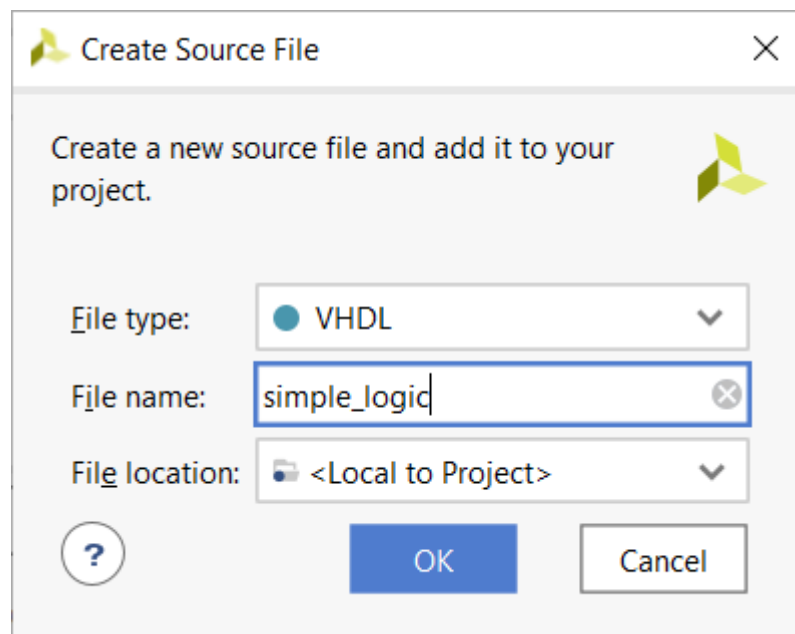


10 pav. VHDL modulio įtraukimas į projektą

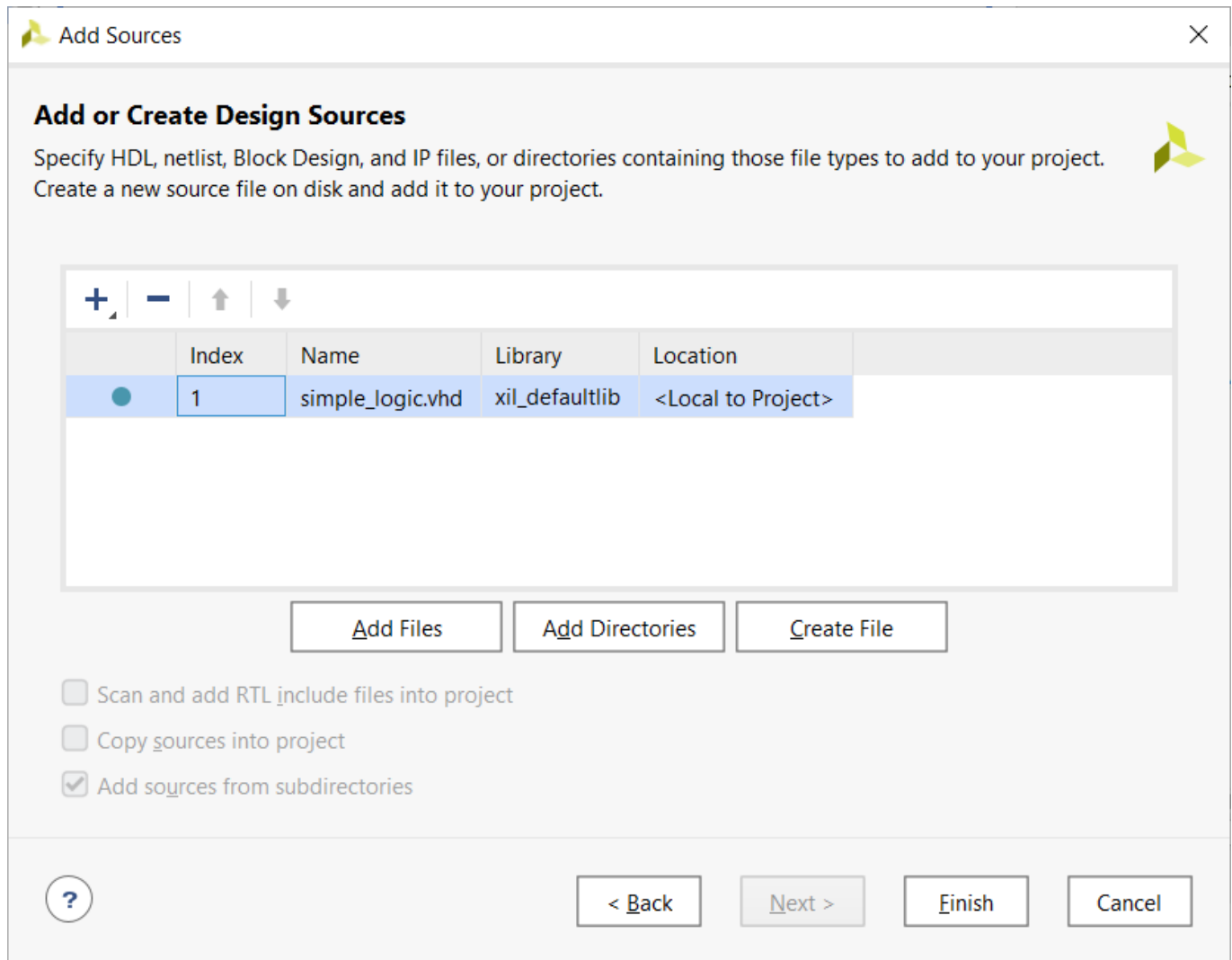


11 pav. VHDL modulio įtraukimo dialogo langas

Atsidariusiame lange pažymime **VHDL Module** ir kuriamo įrenginio pavadinimą, pvz., *simple_logic*. Spaudžiame **Next** ir atsidariusiame objekto aprašo lange pažymime įėjimus bei išėjimus.



12 pav. VHDL modulio pavadinimo įvedimas



13 pav. VHDL modulio įtraukimas dialogo langas

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Entity name:

Architecture name:

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
A	in	<input type="checkbox"/>	0	0
B	in	<input type="checkbox"/>	0	0
C	in	<input type="checkbox"/>	0	0
Y	out	<input type="checkbox"/>	0	0
	in	<input type="checkbox"/>	0	0

OK Cancel

14 pav. Įterpiamo objekto įėjimų/išėjimų aprašas

Naujo objekto šablono kūrimas baigiamas objekto santraukos (Summary) lange paspaudus mygtuką **Finish**.

Sources

- Design Sources (1)
 - simple_logic(Behavioral) (simple_logic.vhd)
- Constraints
- Simulation Sources (1)
 - sim_1 (1)
- Utility Sources

Source File Properties

simple_logic.vhd

Enabled

Location: F:/Works/Xilinx/Lab2Tutorial/Lab_2/Lab_2.srscs/sources_1/new/simple_logic.vhd

Type: VHDL

Library: xil_defaultlib

Project Summary

simple_logic.vhd

```

22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity simple_logic is
35     Port ( A : in STD_LOGIC;
36           B : in STD_LOGIC;
37           C : in STD_LOGIC;
38           Y : out STD_LOGIC);
39 end simple_logic;
40
41 architecture Behavioral of simple_logic is
42
43 begin
44
45
46 end Behavioral;
    
```

15 pav. Įrenginio sąsajos aprašas VHDL kalba Vivado aplinkos projekte

Loginio įrenginio veikimą aprašome įterpdami geltonai pažymėta eilutę.

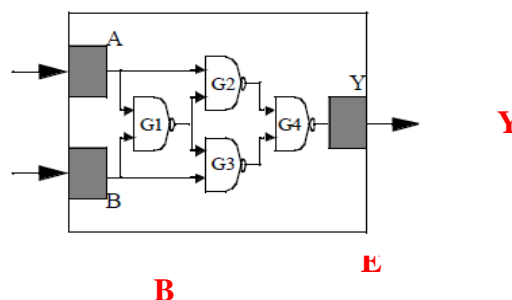
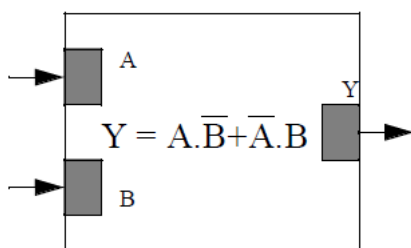
```
entity simple_logic is
  Port ( A : in STD_LOGIC;
        B : in STD_LOGIC;
        C : in STD_LOGIC;
        Y : out STD_LOGIC);
end simple_logic;

architecture Behavioral of simple_logic is

begin
  Y<=(A and B) or C;
end Behavioral;
```

3.3. Loginių įrenginių aprašų architektūros

Sudėtingų komponentų (įrenginių) aprašymui HDL kalba gali būti naudojamas arba elgsenos (*behavioral*) arba struktūrinis (*structural*) architektūros aprašas. Elgsenos apraše pateikiamas objekto veikimo algoritmas pagrįstas matematiniais ir logikos dėsniais. Struktūriniame objekto apraše aprašomi jį sudarantys atskiri diskretiniai elementai ir jų tarpusavio jungtys. Galimas ir abiejų šių aprašų derinys.



```
75 entity XOR is
76   port ( A,B : in bit; Y : out bit);
77 end XOR;
78
79 -- =====
80
81 architecture BEHAVIOR of XOR is
82   begin
83     Y <= '1' when A/=B else '0';
84   end BEHAVIOR
85
86 -- =====
87
88 architecture BEHAVIOR of XOR is
89   begin
90     Y <= (A and not B) or (not A and B);
91   end BEHAVIOR
92
93 -- =====
94
```

```
93 -- =====
94
95 architecture STRUCTURE of XOR is
96
97   component NAND
98     port ( A, B : in bit; Y : out bit);
99   end component;
100
101   signal C, D, E : bit;
102
103   begin
104     G1 : NAND port map (A, B, C);
105     G2 : NAND port map (A => A, B => C, Y => D);
106     G3 : NAND port map (C, B => B, Y => E);
107     G4 : NAND port map (D, E, Y);
108   end STRUCTURE;
109
110 -- =====
111
```

```
110 -- =====
111
112 architecture MIXED of XOR is
113   component NAND
114     port ( A, B : in bit; Y : out bit);
115   end component;
116
117   signal C, D, E : bit;
118
119   begin
120     D <= A nand C;
121     E <= C nand B;
122     G1 : NAND port map (A, B, C);
123     G4 : NAND port map (D, E, Y);
124   end MIXED;
125
126 -- =====
```

16 pav. Įrenginių architektūros elgsenos (*behavioral*) ir struktūrinis (*structural*) ir mišrus aprašai

3.4. Virtualaus bandymų stendo skirto įrenginio modeliavimui sukūrimas

Sukurto loginio įrenginio veikimo patikrinimui atliekamas funkcinis modeliavimas. Tam sukuriamas bandymo stendas (*TestBench*), tiriamam įrenginiui būdingi įėjimo signalai (stimulai) ir pagal simulatoriaus pateiktas laikines diagramas sprendžiama apie įrenginio veikimo teisingumą.

Kuriant bandymo stendą galima pasinaudoti internete prieinamais bandymo stendų generatoriais, pvz. <https://vhdl.lapinoo.net/testbench/>. Jame reikia nukopijuoti kuriamo loginio įrenginio VHDL aprašą. Kuriamo pavyzdžio atveju tikslinga pasirinkti **No Clock generation** bei **No reset generation** opcijas.

This tool automatically generates a template file for a testbench for the simulation of a VHDL entity.

A testbench is a VHDL code that simulates the environment around your DUT (design under test).

The testbench generates stimuli to the inputs of the DUT and allows to check its functionality and outputs within a simulator.

The declarative part of the testbench is quite boring to write, hence the existence of this automatic generator.

The generator is in constant development, but for now it seems to work pretty well for most standard source files, but it has not yet been extensively tested with a lot of different sources with different writing styles.

Feel free to test it with your files and to report problems to help to improve it.

This is for VHDL only, don't try to paste Verilog code, it will not work ;-)

Simply copy and paste your VHDL code below, then press the Generate button.

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity simple_logic is
    Port ( A : in STD_LOGIC;
          B : in STD_LOGIC;
          C : in STD_LOGIC;
          Y : out STD_LOGIC);
end simple_logic;

architecture Behavioral of simple_logic is
begin
    Y<=(A and B) or C;
end Behavioral;
```

☐ Generate clock and try to automatically guess signal name

☐ Generate clock without guessing signal name

☒ No clock generation

☐ Generate reset (positive polarity)

☐ Generate reset (negative polarity)

☒ No reset generation

Generate

17 pav. Internetinis VHDL bandymų stendų generatorius

Paspaudus mygtuką **Generate** sukuriamas bandymo stendo šablono kodas.

```

-- Testbench automatically generated online
-- at https://vhdl.lapino.net
-- Generation date : 23.9.2021 13:55:51 UTC

library ieee;
use ieee.std_logic_1164.all;

entity tb_simple_logic is      -- Svarbu, kad Vivado aplinkoje bandymų stendo failas
end tb_simple_logic; -- vadintųsi šiuo vardu tb_simple_logic.vhd

architecture tb of tb_simple_logic is

    component simple_logic
        port (A : in std_logic;
              B : in std_logic;
              C : in std_logic;
              Y : out std_logic);
    end component;

    signal A : std_logic;
    signal B : std_logic;
    signal C : std_logic;
    signal Y : std_logic;

begin

    dut : simple_logic
    port map (A => A,
              B => B,
              C => C,
              Y => Y);

    stimuli : process
    begin
        -- EDIT Adapt initialization as needed
        A <= '0';
        B <= '0';
        C <= '0';

        -- EDIT Add stimuli here

        wait;
    end process;

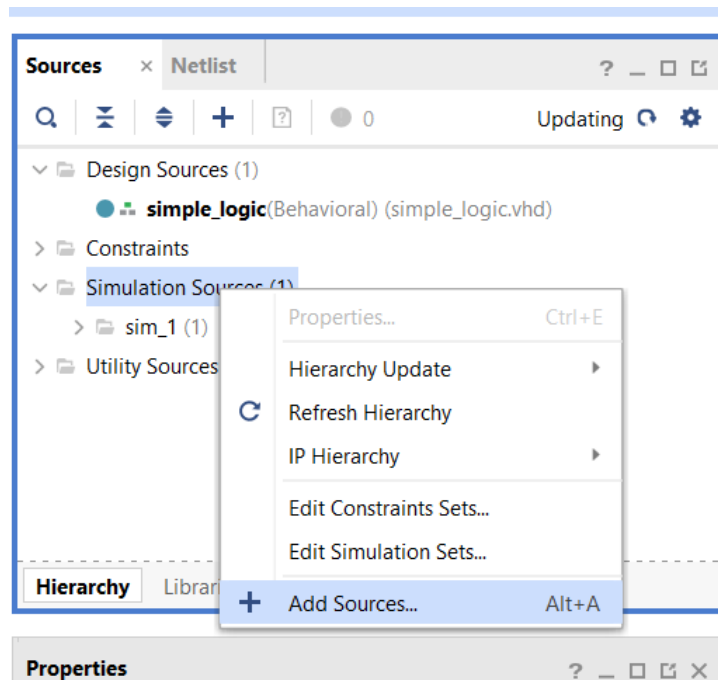
end tb;

-- Configuration block below is required by some simulators. Usually no need to edit.

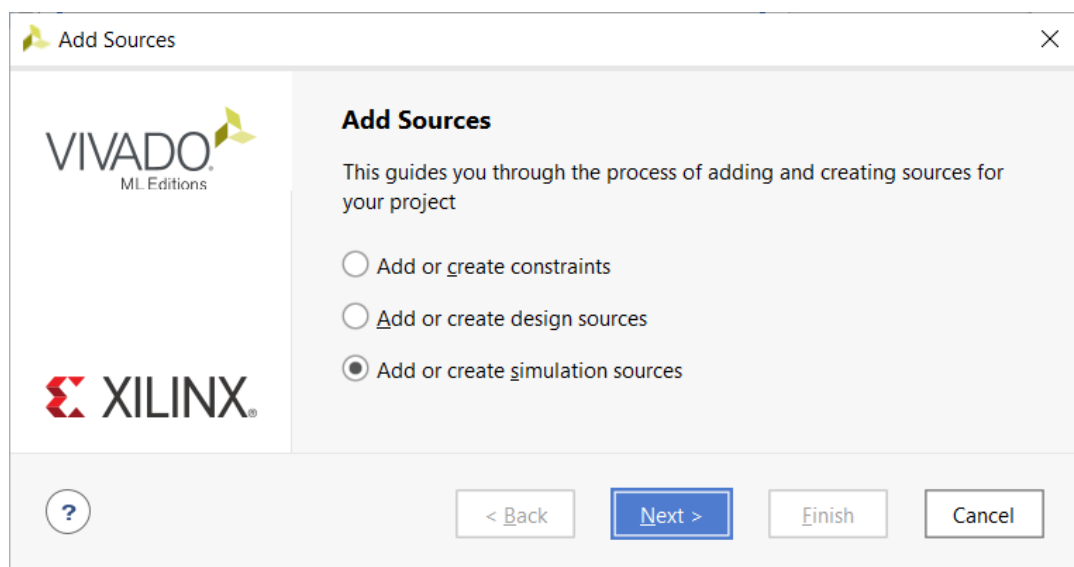
configuration cfg_tb_simple_logic of tb_simple_logic is
    for tb
    end for;
end cfg_tb_simple_logic;

```

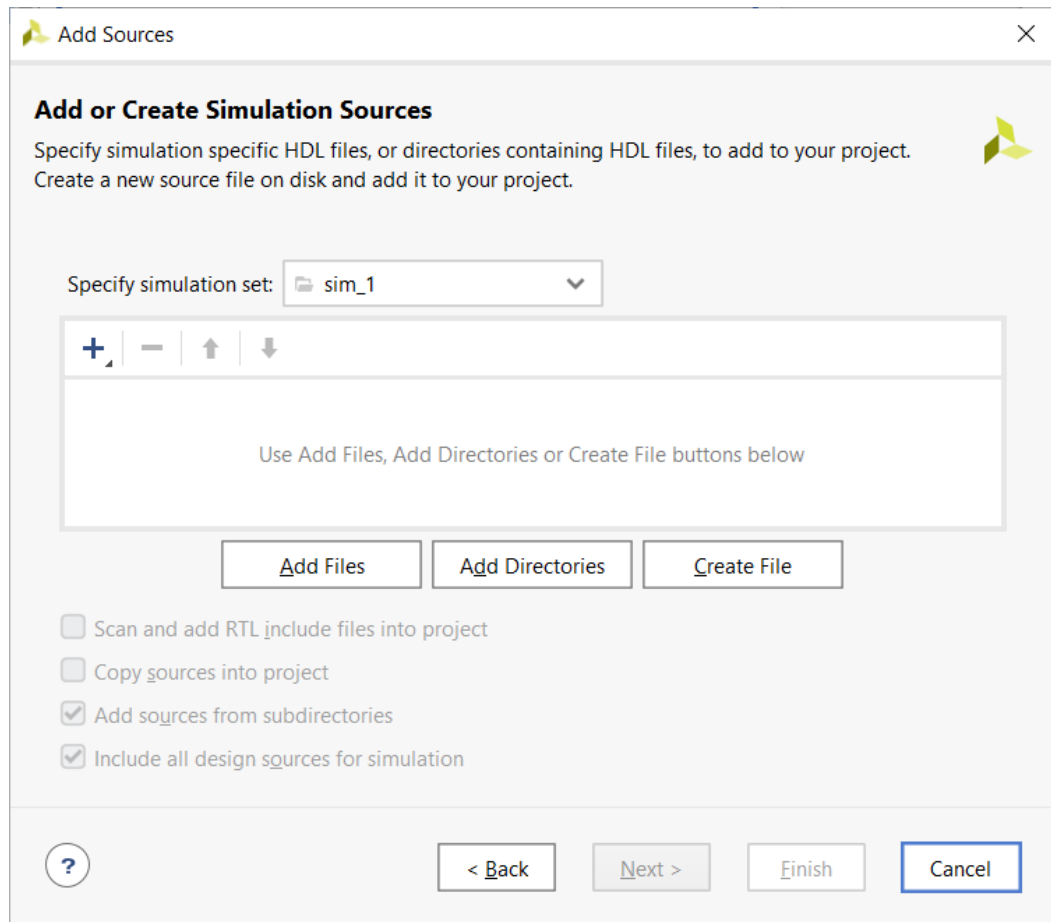
Sugrįžę į Vivado aplinką, bandymų stendą į projektą įtraukiame pasirinkus meniu punktą **Add sources**, punkto **Simulation Sources** iškrentančiame meniu.



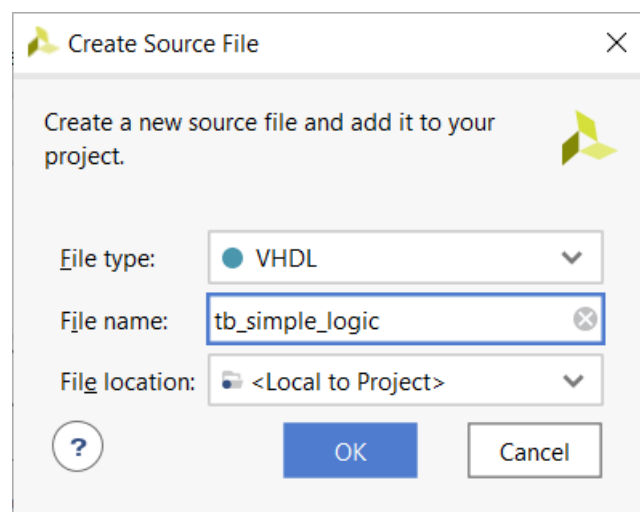
18 pav. Bandymų stendo failo sukūrimas Vivado aplinkoje



19 pav. Bandymų stendo kūrimas Vivado aplinkoje (tęsinys)

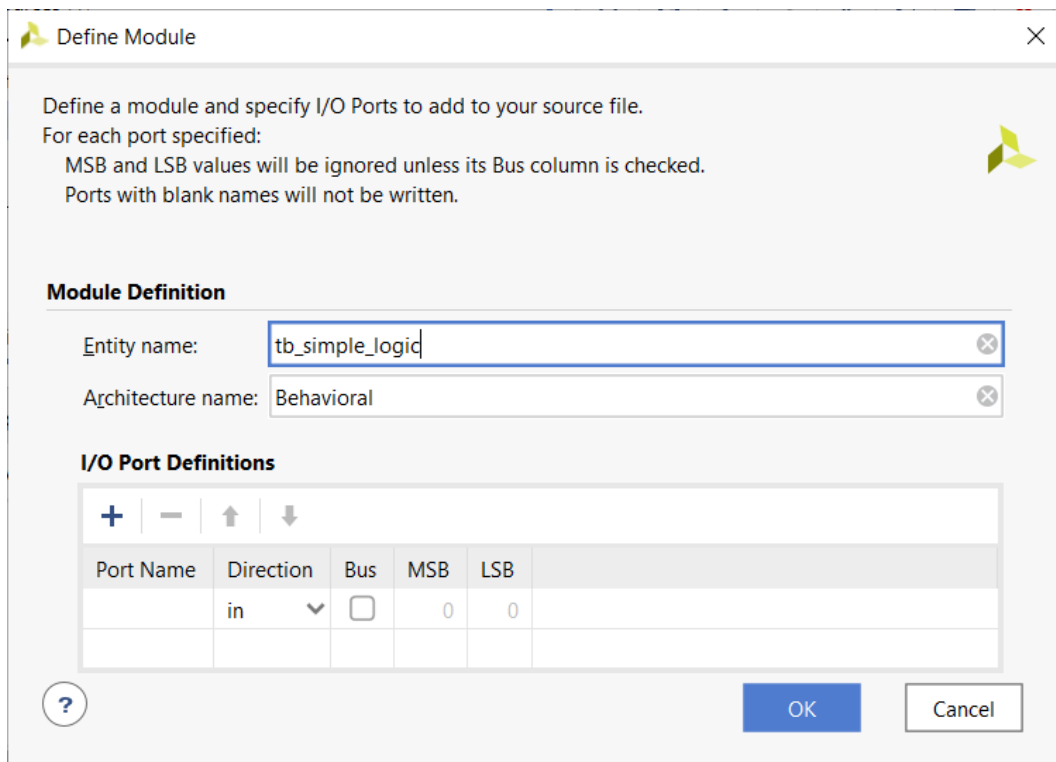


20 pav. Bandymų stendo kūrimas Vivado aplinkoje (tęsinys)



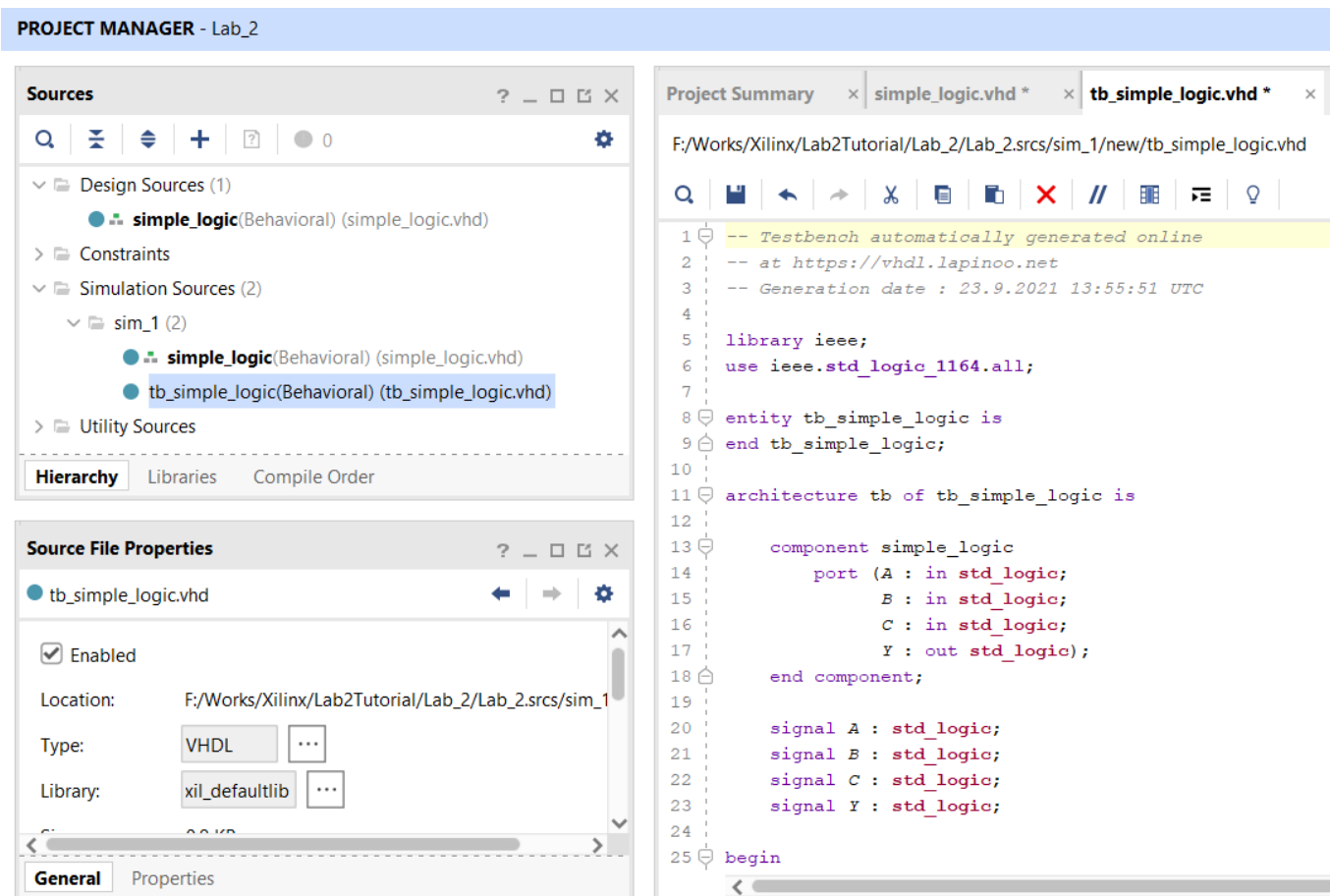
21 pav. Bandymų stendo kūrimas Vivado aplinkoje (pavadinimo įvedimas)

Bandymų stendas neturi **įėjimų/išėjimų**, todėl jų įvedimą praleidžiame ir spaudžiame OK.



22 pav. Bandymų stendo kūrimas Vivado aplinkoje (tęsinys)

Sukūrus bandymų stendo failą, į jį nukopijuojame internete sugeneruoto bandymų stendo šabloną.



23 pav. Bandymų stendo kūrimas Vivado aplinkoje (pabaiga)

Toliau bandymų stendo sakinyje *process* aprašome stimulus VHDL kalba, kaip matyti žemiau. Visą bandymų stendo failą *tb_simple_logic.vhd* galima parsisiųsti iš moodle kurso.

```
-- Testbench automatically generated online
-- at https://vhdl.lapino.net
-- Generation date : 23.9.2021 13:55:51 UTC

library ieee;
use ieee.std_logic_1164.all;

entity tb_simple_logic is
end tb_simple_logic;

architecture tb of tb_simple_logic is

    component simple_logic
        port (A : in std_logic;
              B : in std_logic;
              C : in std_logic;
              Y : out std_logic);
    end component;

    signal A : std_logic := '0';
    signal B : std_logic := '0';
    signal C : std_logic := '0';
    signal Y : std_logic;

begin

    dut : simple_logic
    port map (A => A,
              B => B,
              C => C,
              Y => Y);

    stimuli : process
    begin

        -- EDIT Adapt initialization as needed
        A <= '0';
        B <= '0';
        C <= '0';

        wait for 100 ns;
        A <= '0';
        B <= '0';
        C <= '1';

        wait for 100 ns;
        A <= '0';
        B <= '1';
        C <= '0';

        wait for 100 ns;
        A <= '0';
        B <= '1';
        C <= '1';

        wait for 100 ns;
        A <= '1';
        B <= '1';
        C <= '1';
```



```

-- EDIT Add stimuli here

wait;
end process;

end tb;

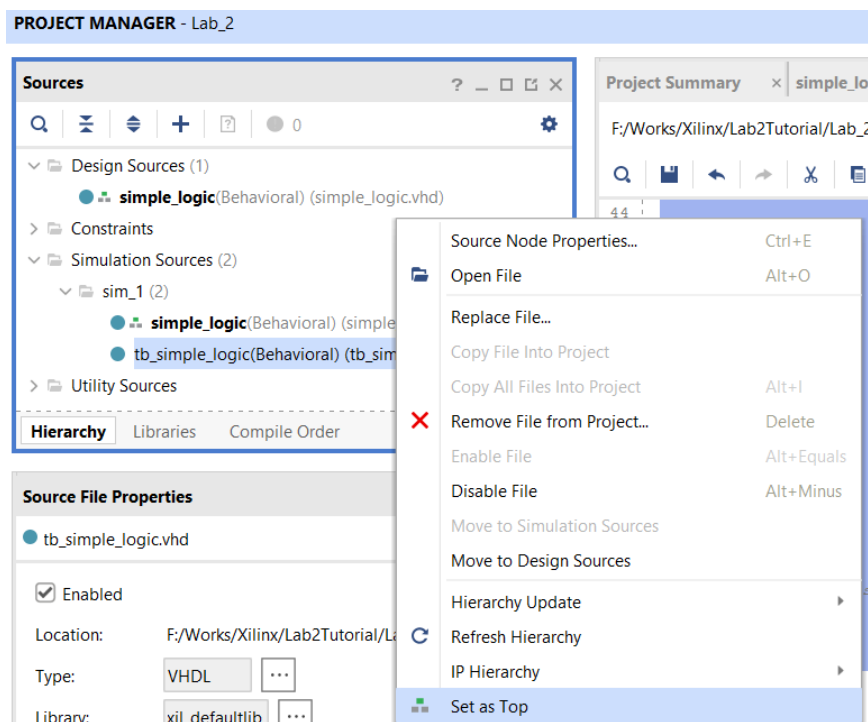
-- Configuration block below is required by some simulators. Usually no need to edit.
configuration cfg_tb_simple_logic of tb_simple_logic is
  for tb
  end for;
end cfg_tb_simple_logic;

```

3.5. Įrenginio modeliavimas

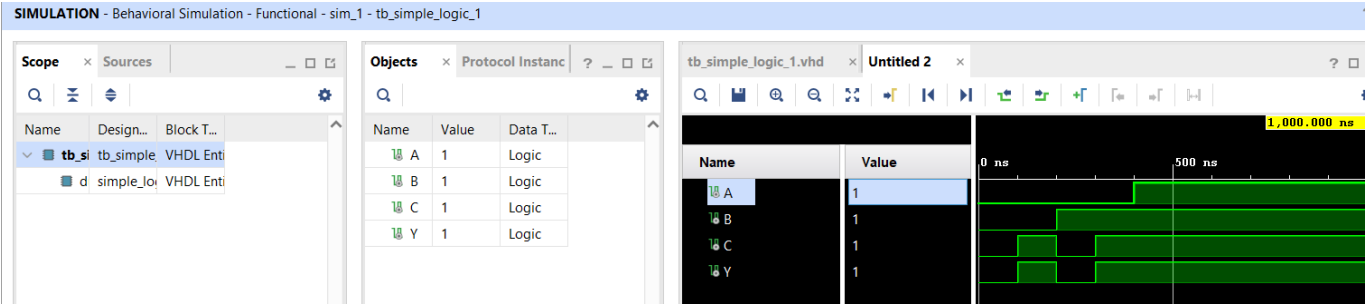
3.5.1. Modeliavimo paleidimas

PROJECT MANAGER panelėje **Settings** dialogo lange pasirinkite **Project Settings / Target Language: VHDL** ir **Tool Settings / Target Language : VHDL**. Bandymų standui **tb_simple_logic.vhd** priskirkite aukščiausią projekto hierarchijos lygį (**Set As Top**).



24 pav. Aukščiausio hierarchijos lygio priskyrimas bandymų standui













Įrenginio modeliavimą paleidžiame pasirinkdami **Flow Navigator / SIMULATION / Run Simulation / Run behavioral simulation**. Modeliavimo rezultatus turime pamatyti laikinių diagramų pavidalu atsivėrusiame lange.



25 pav. Stimulų ir reakcijų signalų diagramos

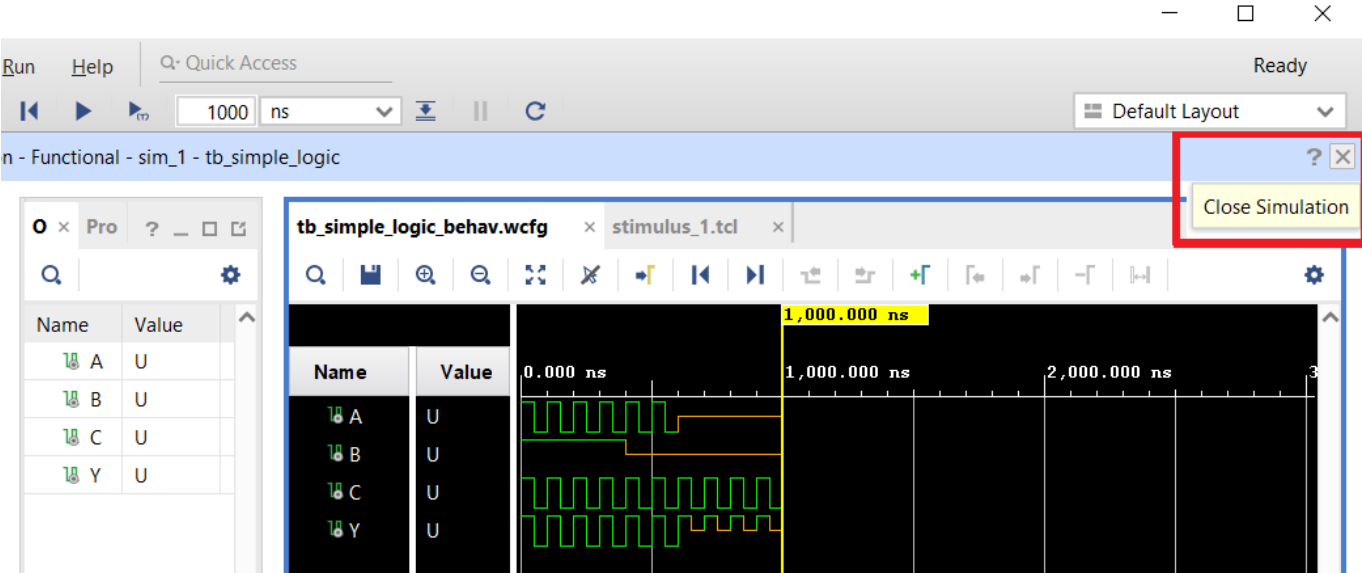
Modeliavimo proceso valdymui naudokitės įrankių liniuotės mygtukais, kurių funkcijos paaiškintos žemiau esančiose lentelėse.

Restart	Run All	Run for 1000 ns	Simulation step	Time units	Step (F8)	Break (F5)	Relaunch simulation

<div></div>											
Search	Save Waveform Configuration	Zoom In	Zoom Out	Zoom Fit	Unselect All	Go to Cursor	Go to time 0	Go to last Time	Previous Transition	Next Transition	Add Marker

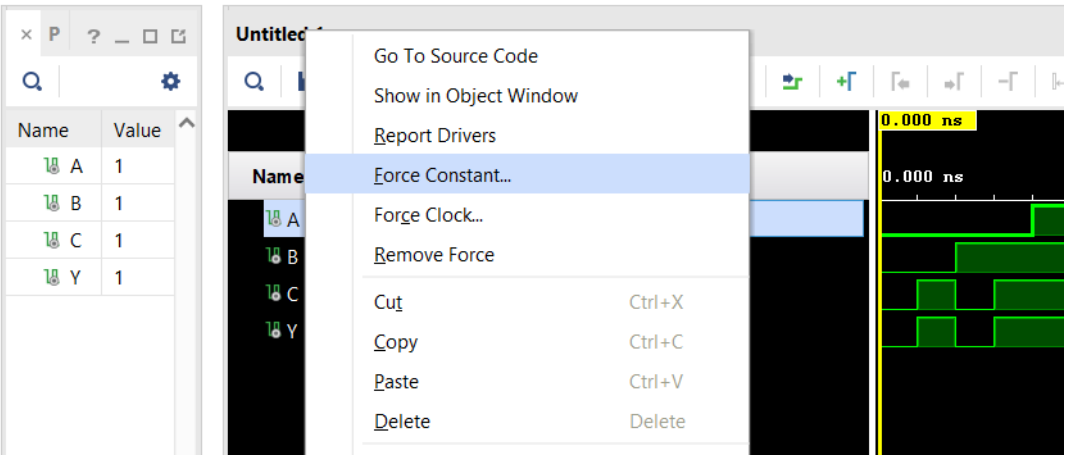
3.5.2. Modeliavimo išjungimas

Sustabdyti modeliavimą ir grįžti į projekto redagavimo režimą galima paspaudus x mygtuką (Close Simulation) modeliavimo lange kaip parodyta žemiau.



3.6. Stimulų generavimas simulatoriaus aplinkoje

Stimulus galima aprašyti ir simulatoriaus aplinkoje (ne su VHDL bandymų stendu). Tam pažymėjus signalą laikinių diagramų lange pasirenkami punktai **Force Constant...** arba **Force Clock...** ir atsivėrusiuose dialogo languose pasirenkami stimulų parametrai.



Lentelėje pateikti stimulų priskyrimai nagrinėjamo pavyzdžio įėjimo signalams A, B ir C.

A signalas – Force Clock ...	B signalas – Force Const ...	C signalas – Force Const...
------------------------------	------------------------------	-----------------------------

Three dialog boxes for forcing signals in a testbench:

- Force Clock: /tb_simple_logic/A**
 - Signal name: /tb_simple_logic/A
 - Value radix: Binary
 - Leading edge value: 1
 - Trailing edge value: 0
 - Starting after time offset: 0ns
 - Cancel after time offset: 600ns
 - Duty cycle (%): 50
 - Period: 100ns
- Force Constant: /tb_simple_logic/B**
 - Signal name: /tb_simple_logic/B
 - Value radix: Binary
 - Force value: 1
 - Starting after time offset: 0ns
 - Cancel after time offset: 400ns
- Force Constant: /tb_simple_logic/C**
 - Signal name: /tb_simple_logic/C
 - Value radix: Binary
 - Force value: 0
 - Starting after time offset: 0ns
 - Cancel after time offset: 1000ns

Parodytą stimulų konfigūravimą galima atlikti ir Tcl Console lange įvykdant Tcl skripto komandas:

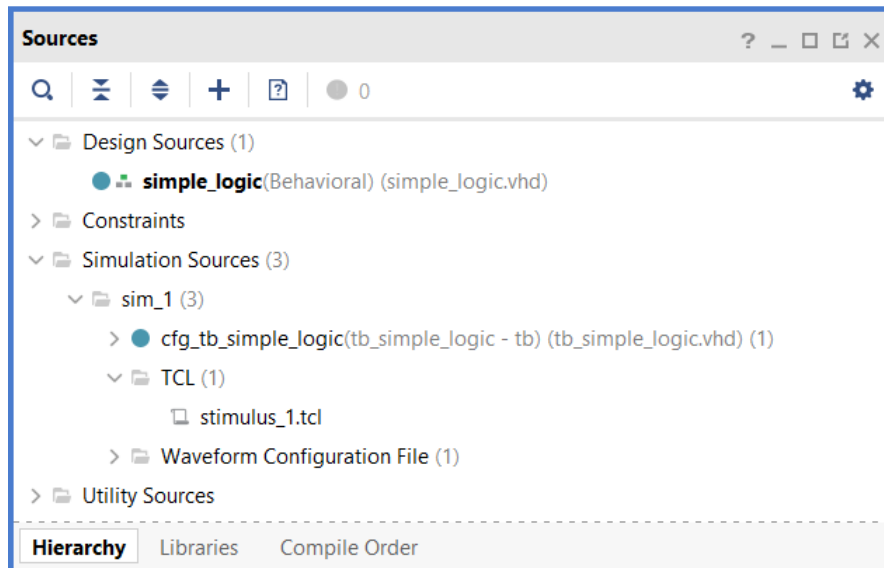
```
add_force {/tb_simple_logic/A} -radix bin {1 0ns} {0 50000ps} -repeat_every 100000ps -cancel_after 600ns
add_force {/tb_simple_logic/B} -radix bin {1 0ns} -cancel_after 400ns
add_force {/tb_simple_logic/C} -radix bin {0 0ns} -cancel_after 1000ns
run 1000 ns
```


Šias eilutes tikslinga išsaugoti į tcl failą, kurį vėliau pakartotinai atliekant modeliavimą galėsime iškviešti kaip komandinės eilutės skriptą. Tam tekstiniu redaktoriumi sukuriame failą stimulus_1.tcl ir įtraukiame jį į projektą pasirinkdami PROJECT MANAGER / Add Sources punktą bei sekdami tolimesnius nurodymus.

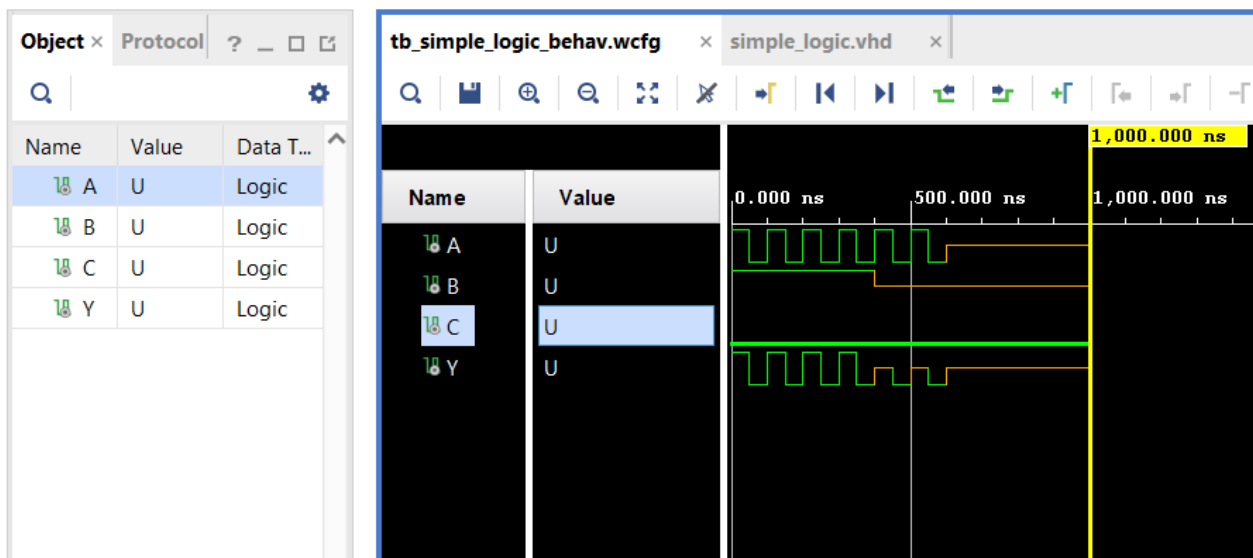
Project Manager window showing the stimulus_1.tcl file added to the project. The file content is visible in the editor:

```
1 add_force {/tb_simple_logic/A} -radix bin {1 0ns} {0 50000ps} -repeat_every 100000ps -cancel_after 600ns
2 add_force {/tb_simple_logic/B} -radix bin {1 0ns} -cancel_after 400ns
3 add_force {/tb_simple_logic/C} -radix bin {0 0ns} -cancel_after 1000ns
4 run 1000 ns
5
```

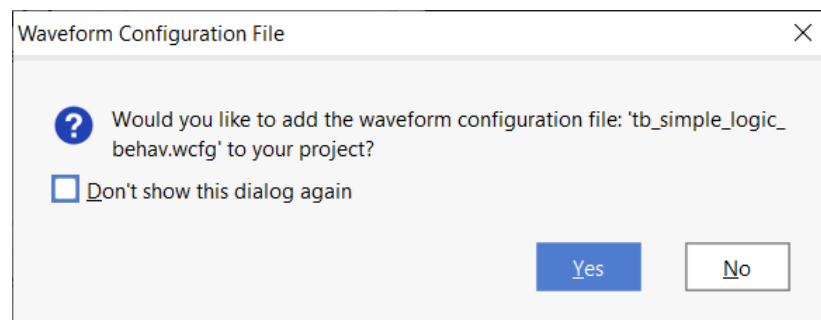
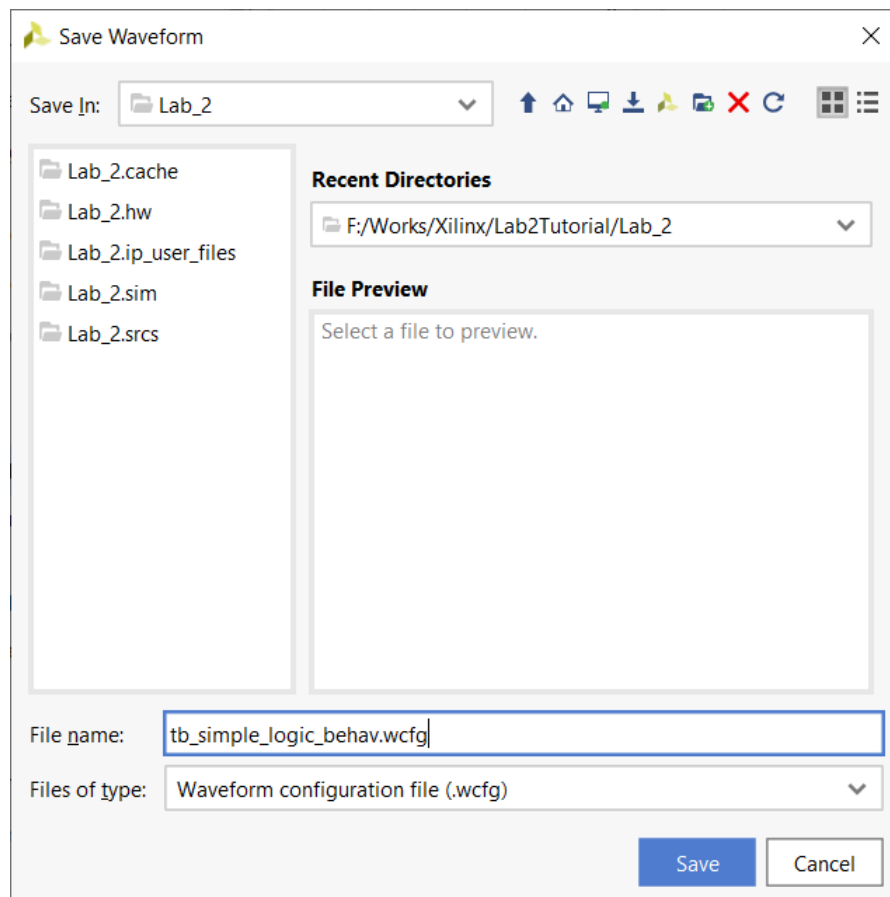
Į projektą įtrauktas tcl skriptas projekto failų medyje matomas, kaip parodyta žemiau.



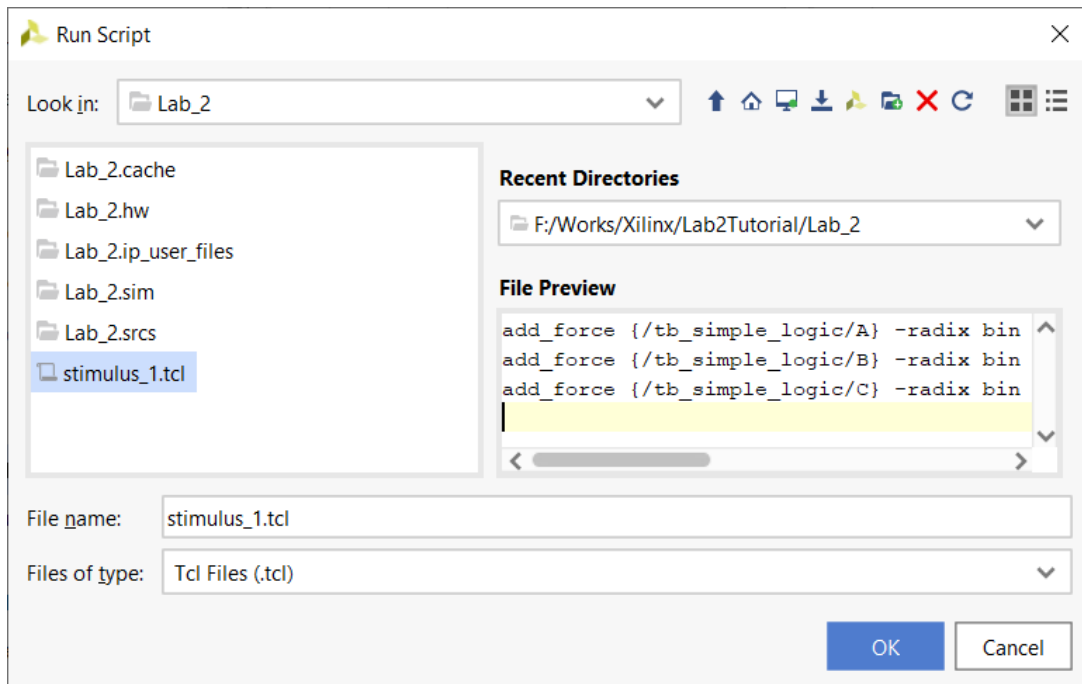
Modeliavimą vykdome paspausdami mygtuką  įrankių liniuotėje. Gautus rezultatus turime pamatyti laikinių diagramų lange.



Pasirinkus įrankių liniuotėje **Save Waveform Configuration** galime išsaugoti laikinių diagramų lango nustatymus.



Pirmą kartą arba iš naujo paleidus modeliavimą (**Relaunch**), pasirinkite meniu punktą **Tools / Run Tcl Script ...** ir nurodykite *stimulus_1.tcl* skriptą.



Paspaudus mygtuką **OK**, skriptas bus įvykdytas (komandas matysime Tcl Console lange): priskirtos stimulų reikšmės ir įvykdytas modeliavimas, kurio trukmė 1000 ns. Rezultatai atvaizduojami laikinių diagramų lange.

SIMULATION - Behavioral Simulation - Functional - sim_1 - cfg_tb_simple_logic

Scope x Sources

Name	Design...	Block T...
tb_	tb_simple	VHDL Enti
c	simple_loi	VHDL Enti

Objec x Proto

Name	Value	Data T...
A	0	Logic
B	U	Logic
C	U	Logic
Y	U	Logic

stimulus_1.tcl x tb_simple_logic_behav.wcfg

Name	Value
A	0
B	U
C	U
Y	U

Tcl Console x Messages Log

```

source F:/Works/Xilinx/Lab2Tutorial/Lab_2/stimulus_1.tcl
# add_force {/tb_simple_logic/A} -radix bin {1 0ns} {0 50000ps} -repeat_every 100000ps -cancel_after 600ns
# add_force {/tb_simple_logic/B} -radix bin {1 0ns} -cancel_after 400ns
# add_force {/tb_simple_logic/C} -radix bin {0 0ns} -cancel_after 600ns
# run 1000 ns
  
```

Jeigu laikinai norite nebenaudoti tcl faile esančio skripto, pažymėkite failą ir iškrentančiame meniu pasirinkite **Disable File**. Vėliau galėsite jį vėl prijungti, jeigu pasirinksite **Enable File**.

3.7. Stimulų aprašų generavimas bandymų stende, panaudojant Vivado šablonus

Panagrinakime papildomus pavyzdžius, kaip stimulai gali būti generuojami bandymų stendo pagalba. Tam atjunkime *stimulus_1.tcl* failą **Disable File** komandos pagalba.

Pridėkite į projektą antrą bandymų stendo failą, pvz., *test_bench_2.vhd* (**PROJECT MANAGER/ Add Sources/ Add or create simulation sources**).

Žemiau pateiktame stendo apraše naujos eilutės (lyginant su pirmuoju bandymų stendu) paženklintos geltonai. Šiame stende viename procese sugeneruojamas A signalo stimulus (su konstanta PERIOD pasirenkamo periodo taktiniai impulsai). Antrame procese sugeneruoto A signalo kylantys frontai yra naudojami nustatyti laiko momentams, kada yra keičiama B signalo stimulo reikšmė. Toks bandymų stendų aprašas yra patogus registrinės logikos modeliavimui (nors šiame pavyzdyje jis taikomas ir kombinacinės logikos modeliavimui).

Visą bandymų stendo failą *test_bench_2.vhd* galima parsisiųsti iš moodle kurso.

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

entity test_bench_2 is
end test_bench_2;

architecture tb of test_bench_2 is
constant PERIOD : time := 50 ns;

    component simple_logic
        port (A : in std_logic;
              B : in std_logic;
              C : in std_logic;
              Y : out std_logic);
    end component;

    signal A : std_logic := '0';
    signal B : std_logic := '0';
    signal C : std_logic := '0';
    signal Y : std_logic;

begin

    dut : simple_logic
    port map (A => A,
              B => B,
              C => C,
              Y => Y);

    A_clock : process
    begin
        A <= '0';
        wait for PERIOD/2;
        A <= '1';
        wait for PERIOD/2;
    end process;

    stimuli : process (A)
    begin
        if (A'event and A = '1') then
            B <= not B;
        end if;

        -- EDIT Adapt initialization as needed
        -- A <= '0';
```



```

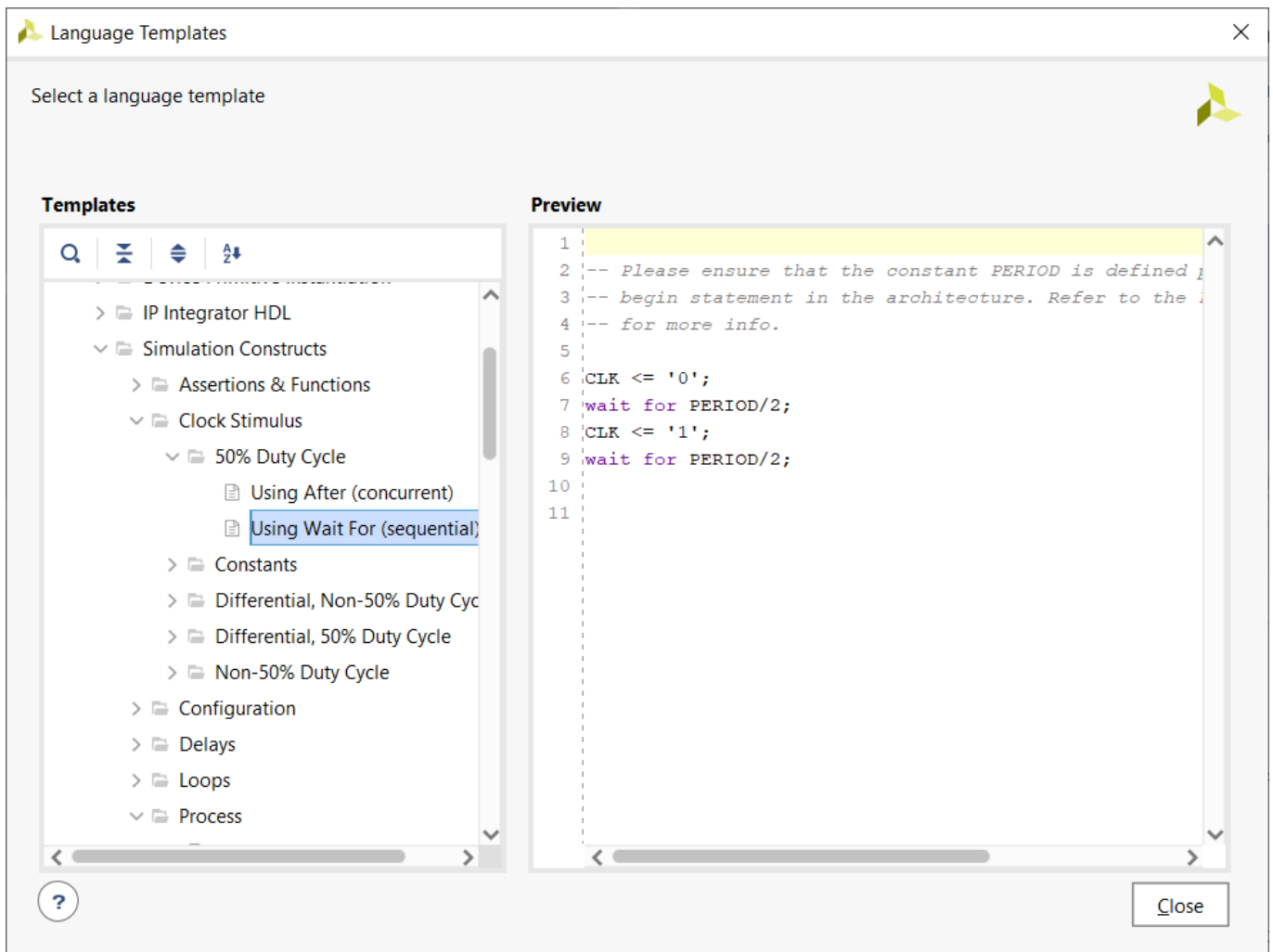
--      B <= '0';
--      C <= '0';

--      wait for 100 ns;
--      A <= '0';
--      B <= '0';
--      C <= '1';
end process;
end tb;

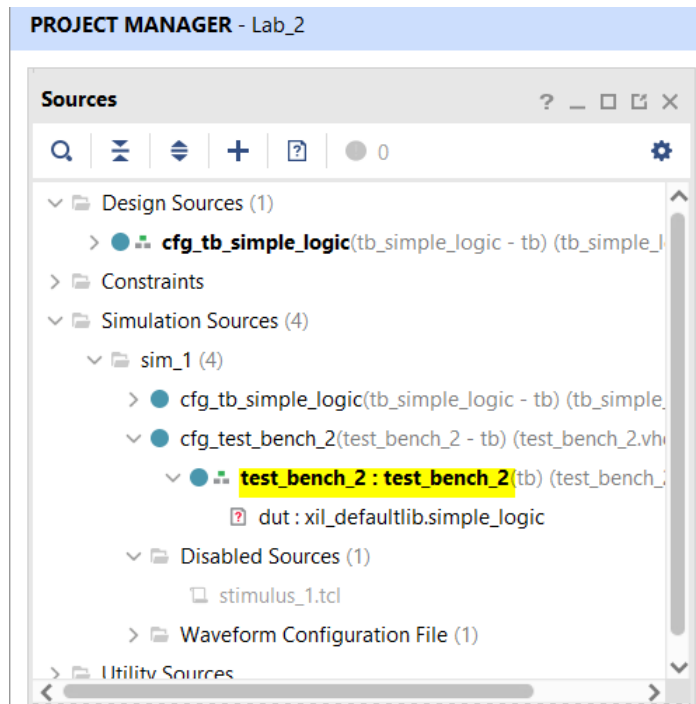
-- Configuration block below is required by some simulators. Usually no need to edit.
configuration cfg_test_bench_2 of test_bench_2 is
  for tb
  end for;
end cfg_test_bench_2;

```

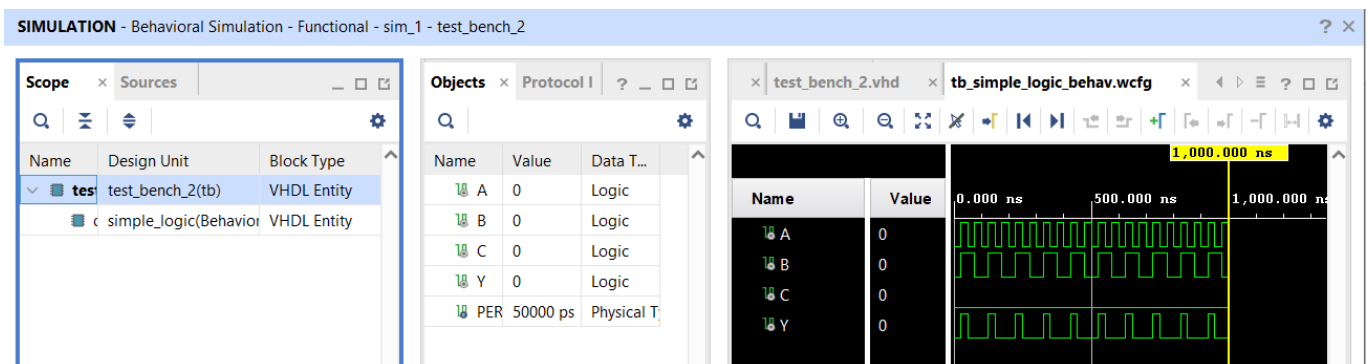
Kuriant bandymo stendą galima naudotis Vivado aplinkos VHDL kodo šablonais (meniu **Tools / Language Templates**) arba papildomais literatūros šaltiniais.



Nepamiškite nustatyti naująjį stendą kaip aukščiausio hierarchijos lygio failą (iškrentančiame meniu Set as Top).



Atlikus modeliavimą turime gauti žemiau parodytas laikines diagramas.



4. Užduotys

VHDL kalba aprašykite ir modeliavimo būdu patikrinkite šių loginių įrenginių veikimą:

1. 4-1 multiplexorius (angl. 4-to-1 multiplexor): keturi duomenų įėjimai, du adreso (select) įėjimai ir vienas išėjimas. Sąsają (interfeisą) realizuoti naudojant vektorių tipo signalus, o ne daug vieno bito signalų.
2. 3-jų bitų operandų aritmetinis sumatorius. Realizuoti panaudojant VHDL kalbos aritmetinių operatorių „+“, o ne ventilių struktūrinio lygmens aprašą. Sąsają (interfeisą) realizuoti naudojant vektorių tipo signalus.
3. 7 bitų postūmio registras. Įėjimo duomenys perduodami į jauniausią postūmio registro bitą, išėjimas – lygiagretus (jo tipas daugiabitis vektorius).

5. Ataskaita ir gynimas

1. Ataskaitoje pateikite sudarytų loginių įrenginių ir jų bandymų standų aprašus bei modeliavimo laikines diagramas.
2. Gynimo metu pasiruoškite sukurti loginių įrenginių pagal sąlygos specifikaciją ir atlikti jo modeliavimą sukuriant bandymų standą.

6. Papildomi šaltiniai

1. Vivado Design Suite User Guide: Logic Simulation UG900 (v2021.1) June 16, 2021.
2. Kombininės logikos įrenginiai https://www.electronics-tutorials.ws/combination/comb_1.html
3. Nuosekliosios logikos įrenginiai https://www.electronics-tutorials.ws/sequential/seq_1.html