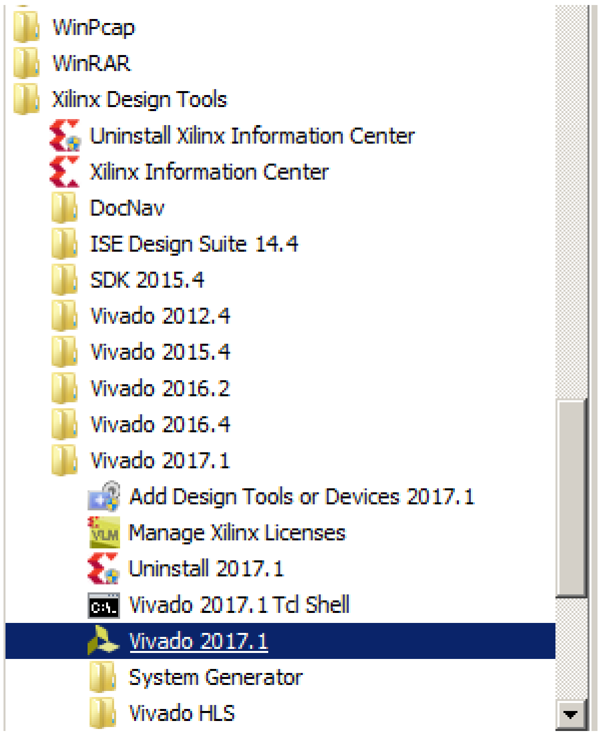
How to use Vivado IDE

1. Running the Vivado IDE

Menu Run => Xilinx Design Tools => Vivado 20xx => Vivado 20xx

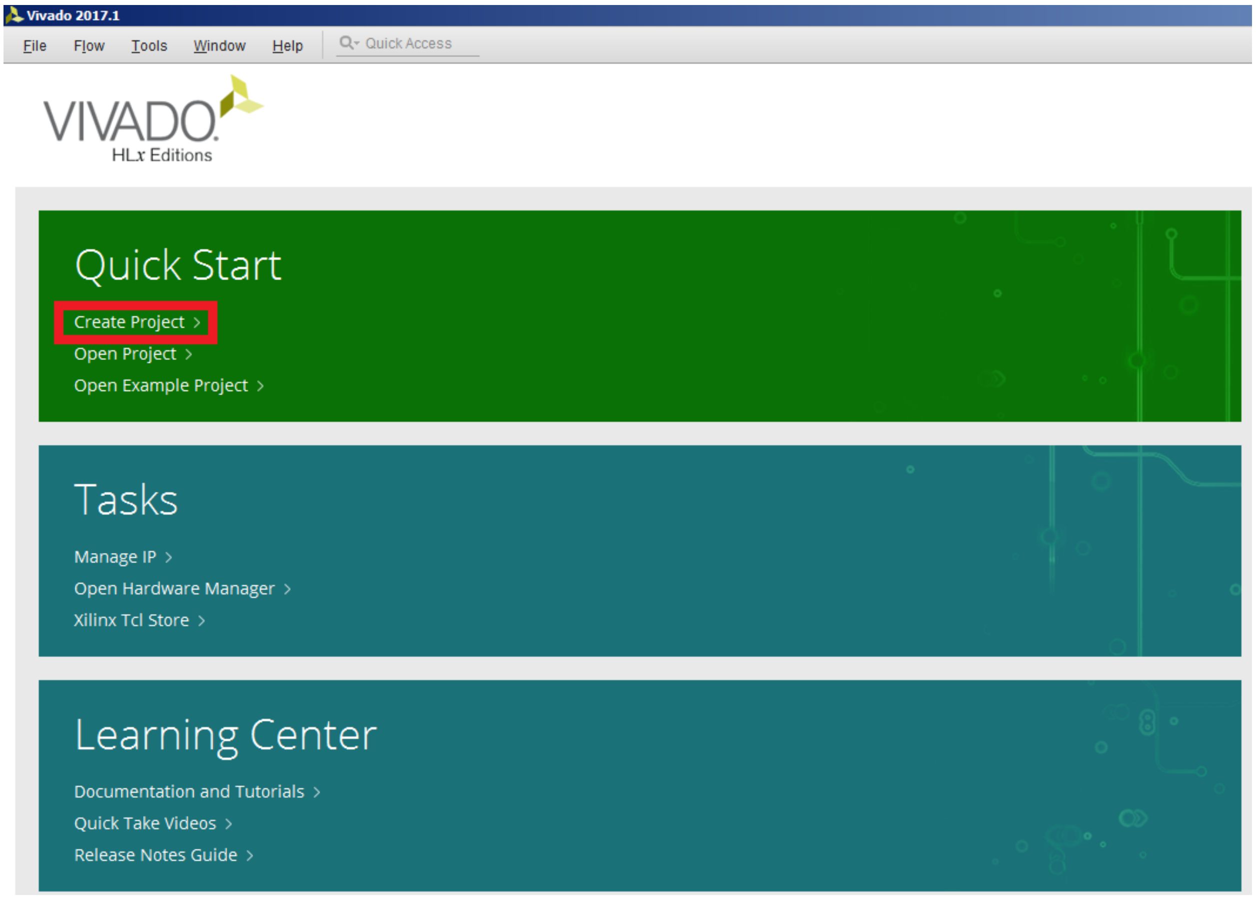


2. Creating the project

Creating the project: **Create Project**.

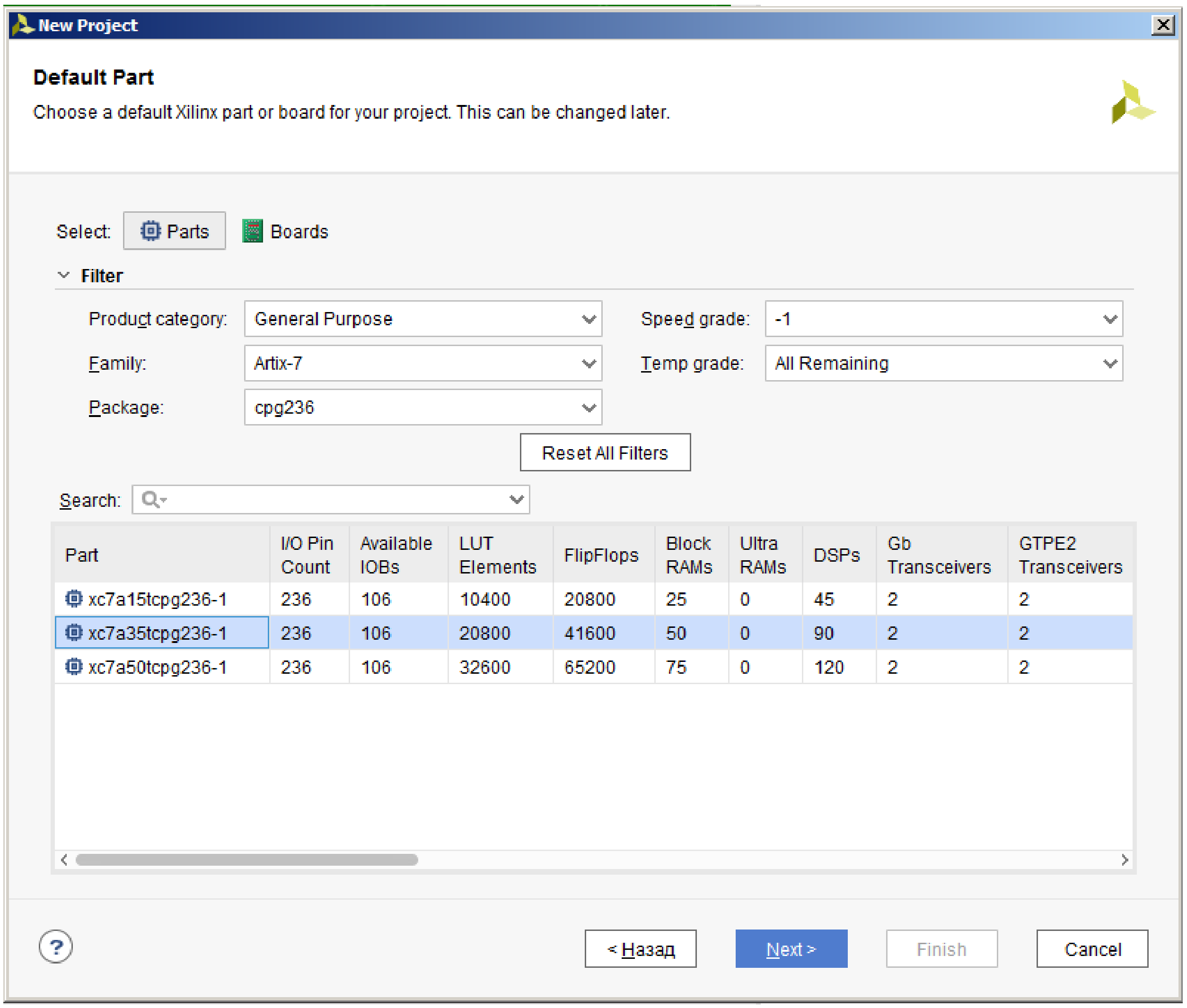
Project type choose to be **RTL Project**.

Always press **Next** up to window where FPGA type specification.



3. Setting project parameters

The type of FPGA, which is used in Basys 3 board, is: Xilinx XC7a35t CPG236



New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Select: Parts Boards

Filter

Product category:	General Purpose	Speed grade:	-1
Family:	Artix-7	Temp grade:	All Remaining
Package:	cpg236		

Reset All Filters

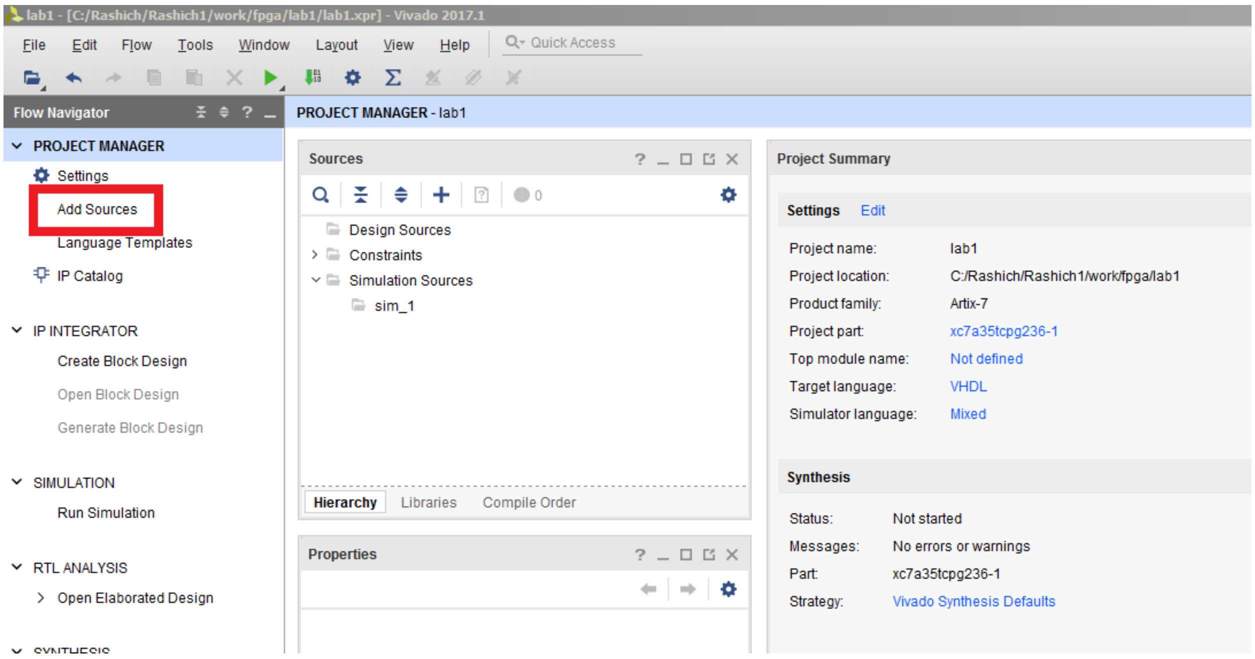
Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transceivers
xc7a15tcpg236-1	236	106	10400	20800	25	0	45	2	2
xc7a35tcpg236-1	236	106	20800	41600	50	0	90	2	2
xc7a50tcpg236-1	236	106	32600	65200	75	0	120	2	2

< Назад | Next > | Finish | Cancel

4. Creating new source file

Creating new source file for RTL inside the project: **Flow Navigator** (to the left side of IDE)=> **Add Sources**.



lab1 - [C:/Rashich/Rashich1/work/fpga/lab1/lab1.xpr] - Vivado 2017.1

Project Summary

Settings Edit

Project name:	lab1
Project location:	C:/Rashich/Rashich1/work/fpga/lab1
Product family:	Artix-7
Project part:	xc7a35tcpg236-1
Top module name:	Not defined
Target language:	VHDL
Simulator language:	Mixed

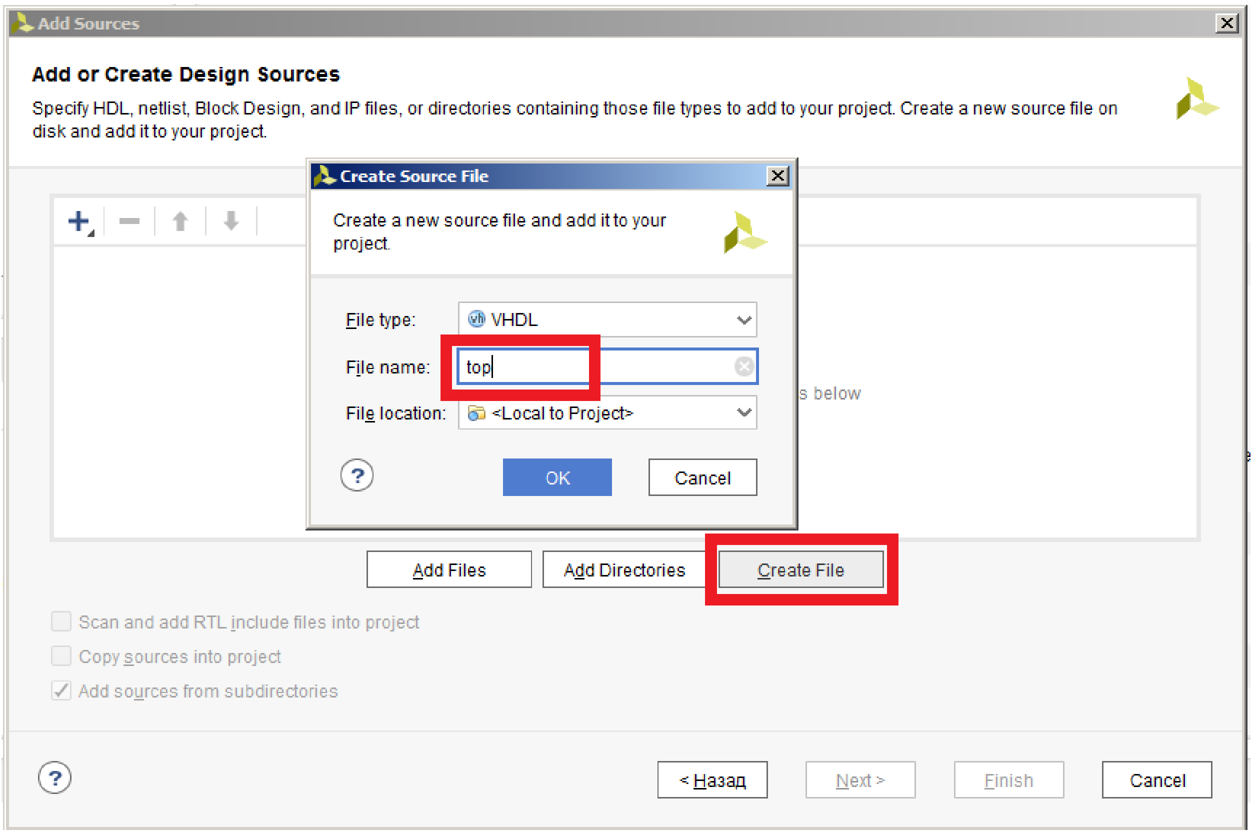
Synthesis

Status:	Not started
Messages:	No errors or warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Synthesis Defaults

Specify:

- **Add or create design sources** to add or create VHDL/Verilog files;
- **Add or create constraints** to add or create *.xdc files with constraints description (mainly – FPGA pins connections to entity ports);
- **Add or create simulation sources** to add or create the source files, which are used only in simulation.

Further press **Create File**.



Add Sources

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

Create Source File

Create a new source file and add it to your project.

File type: VHDL

File name: top

File location: <Local to Project>

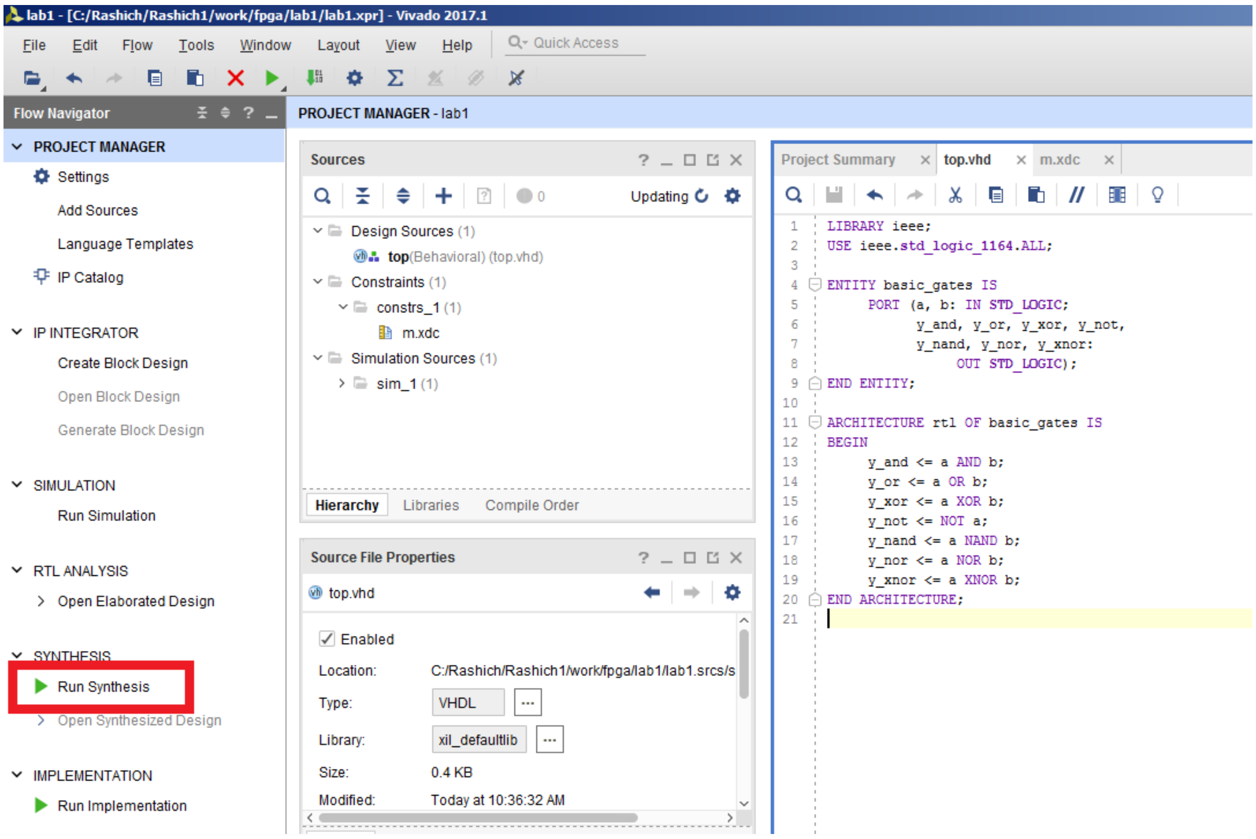
OK
Cancel
Add Files
Add Directories
Create File
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
< Назад
Next >
Finish
Cancel

5. Project synthesis

Press **Run Synthesis** in the Flow Navigator. In the appeared window press **OK**.

In general case, one should check if the synthesized module is the project *top* module (next to its' name 2 small white squares and 1 green one are shown).

After the synthesis is finished, a window appears where one may press **Open Synthesized Device** for some analysis. Also one may continue building the project, i.e. proceed to **implementation (place and route)** step.


lab1 - [C:/Rashich/Rashich1/work/fpga/lab1/lab1.xpr] - Vivado 2017.1

File Edit Flow Tools Window Layout View Help Quick Access

Flow Navigator | PROJECT MANAGER - lab1

- PROJECT MANAGER
  - Settings
  - Add Sources
  - Language Templates
  - IP Catalog
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION
  - Run Simulation
- RTL ANALYSIS
  - Open Elaborated Design
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
- IMPLEMENTATION
  - Run Implementation

Sources

- Design Sources (1)
  - top(Behavioral) (top.vhd)
- Constraints (1)
  - constrs_1 (1)
    - m.xdc
- Simulation Sources (1)
  - sim_1 (1)

Hierarchy | Libraries | Compile Order

Source File Properties

top.vhd

Enabled	✓
Location:	C:/Rashich/Rashich1/work/fpga/lab1/lab1.srcs/s
Type:	VHDL
Library:	xil_defaultlib
Size:	0.4 KB
Modified:	Today at 10:36:32 AM

Project Summary | top.vhd | m.xdc

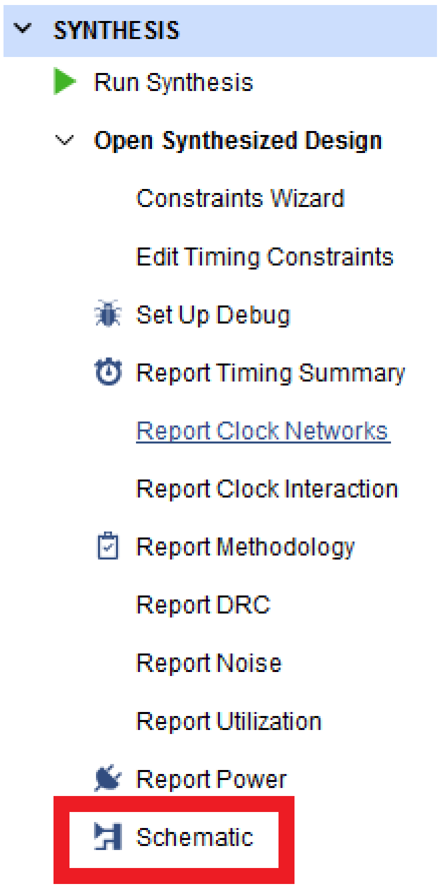
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY basic_gates IS
     PORT (a, b: IN STD_LOGIC;
           y_and, y_or, y_xor, y_not,
           y_nand, y_nor, y_xnor:
                OUT STD_LOGIC);
END ENTITY;

ARCHITECTURE rtl OF basic_gates IS
BEGIN
     y_and <= a AND b;
     y_or <= a OR b;
     y_xor <= a XOR b;
     y_not <= NOT a;
     y_nand <= a NAND b;
     y_nor <= a NOR b;
     y_xnor <= a XNOR b;
END ARCHITECTURE;
```

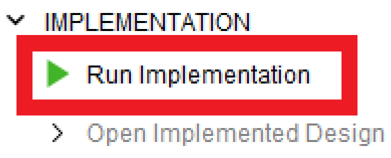
6. Looking at the device schematics after synthesis

Press **Schematic** in the Flow Navigator/Synthesis.


- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
    - Constraints Wizard
    - Edit Timing Constraints
    - Set Up Debug
    - Report Timing Summary
    - Report Clock Networks
    - Report Clock Interaction
    - Report Methodology
    - Report DRC
    - Report Noise
    - Report Utilization
    - Report Power
    - Schematic

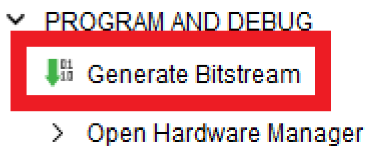
7. Project implementation (place&route)

Press **Run Implementation** in the Flow Navigator.



8. Bitstream generation

Press **Generate Bitstream** in the Flow Navigator.



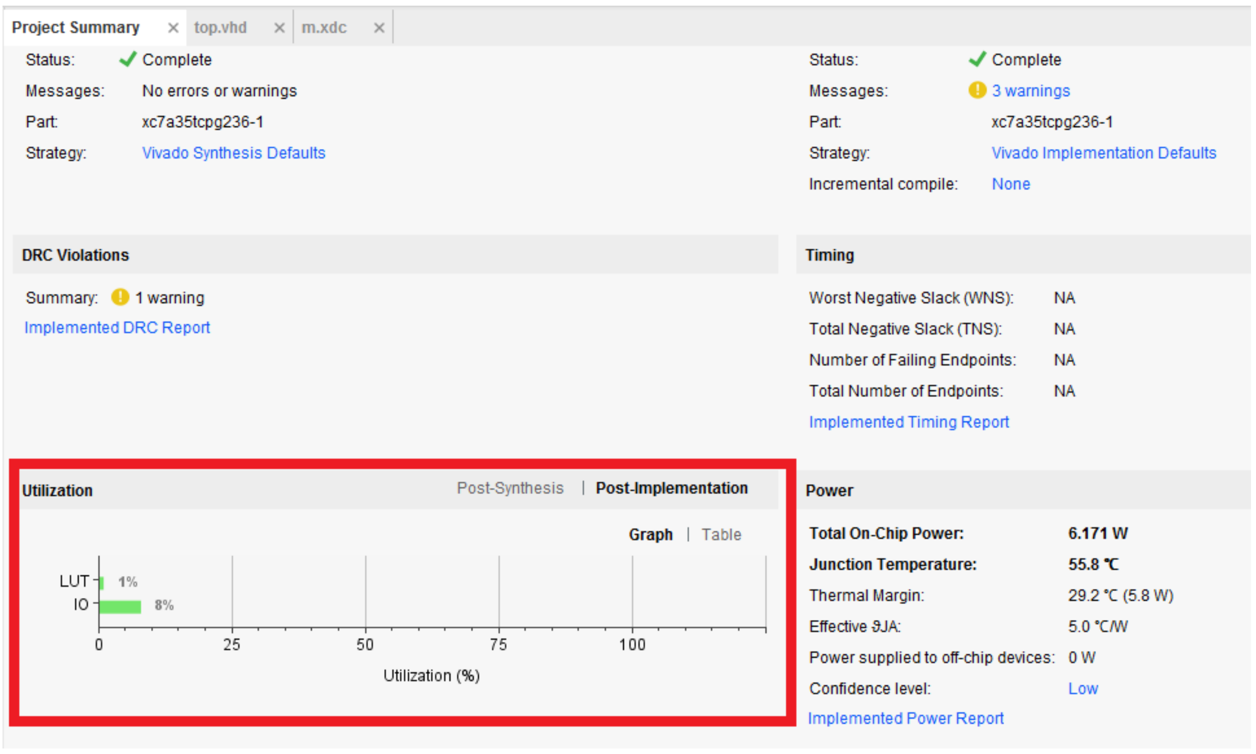
9. Looking at the device schematics after implementation

Press **Schematic** in the Flow Navigator/Implementation.

10. Checking the number of used resources

Find the table **Utilization** in the Project Summary tab in the bottom (one may switch between **Graph** and **Table** views).

In general case, the **Utilization** table contents may differ after the synthesis and implementation steps. The synthesis step (especially when IP cores are used) provides only the estimates of resources usage.



Project Summary | top.vhd | m.xdc

Status:	✓ Complete
Messages:	No errors or warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Synthesis Defaults

Status:	✓ Complete
Messages:	3 warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Implementation Defaults
Incremental compile:	None

DRC Violations

Summary: 1 warning

Implemented DRC Report

Timing

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization Post-Synthesis | **Post-Implementation**

Graph | Table

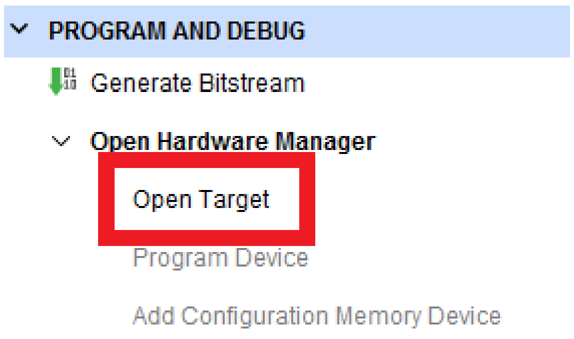
Power

Total On-Chip Power:	**6.171 W**
Junction Temperature:	**55.8 °C**
Thermal Margin:	29.2 °C (5.8 W)
Effective ϑJA:	5.0 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

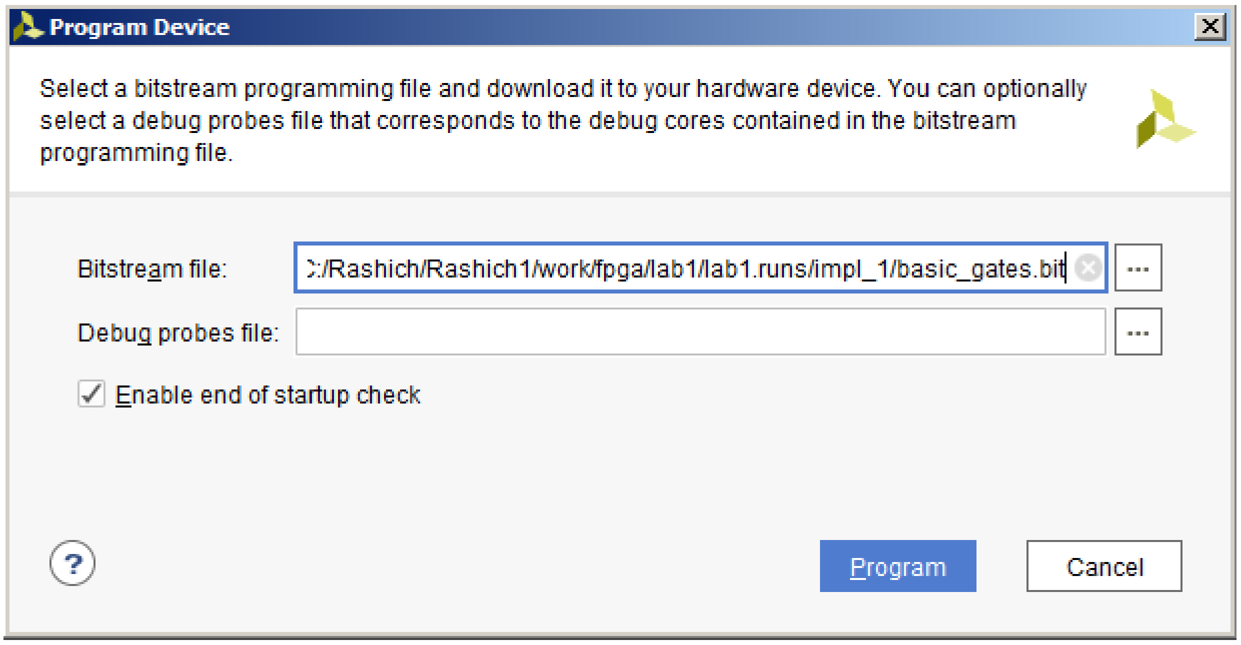
11. Programming the FPGA with bitstream

Connect the Basys 3 development board to your PC (some drivers installation may run automatically at the first connection). Turn on the board. In the Flow Navigator in tab **Program and Debug/Open Hardware Manager** press **Open Target (Auto Connect)**.



Vivado will find the connected Xilinx devices.

In the Flow Navigator in **Program and Debug/Open Hardware Manager** press **Program Device**. In the appeared window check if the necessary bitstream is specified and press **Program**.


Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: :/Rashich/Rashich1/work/fpga/lab1/lab1.runs/impl_1/basic_gates.bit

Debug probes file:

✓ Enable end of startup check

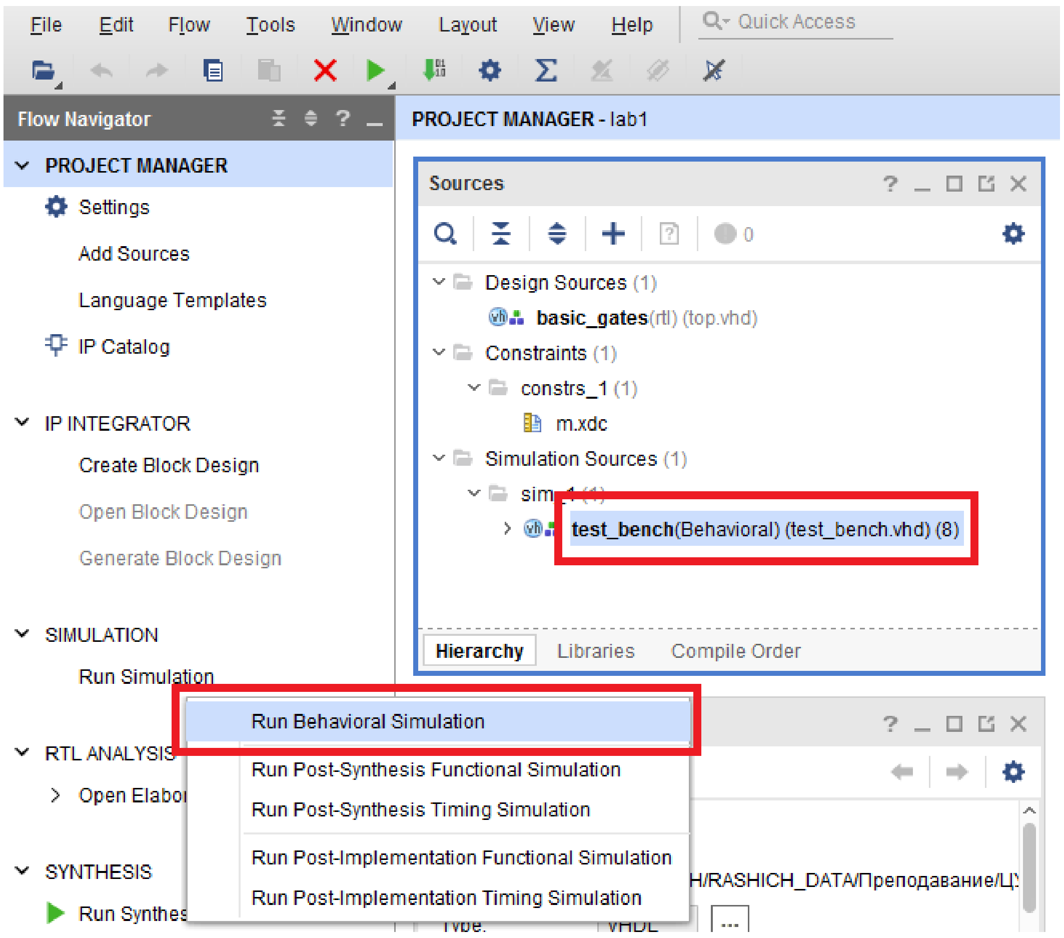
12. Simulation

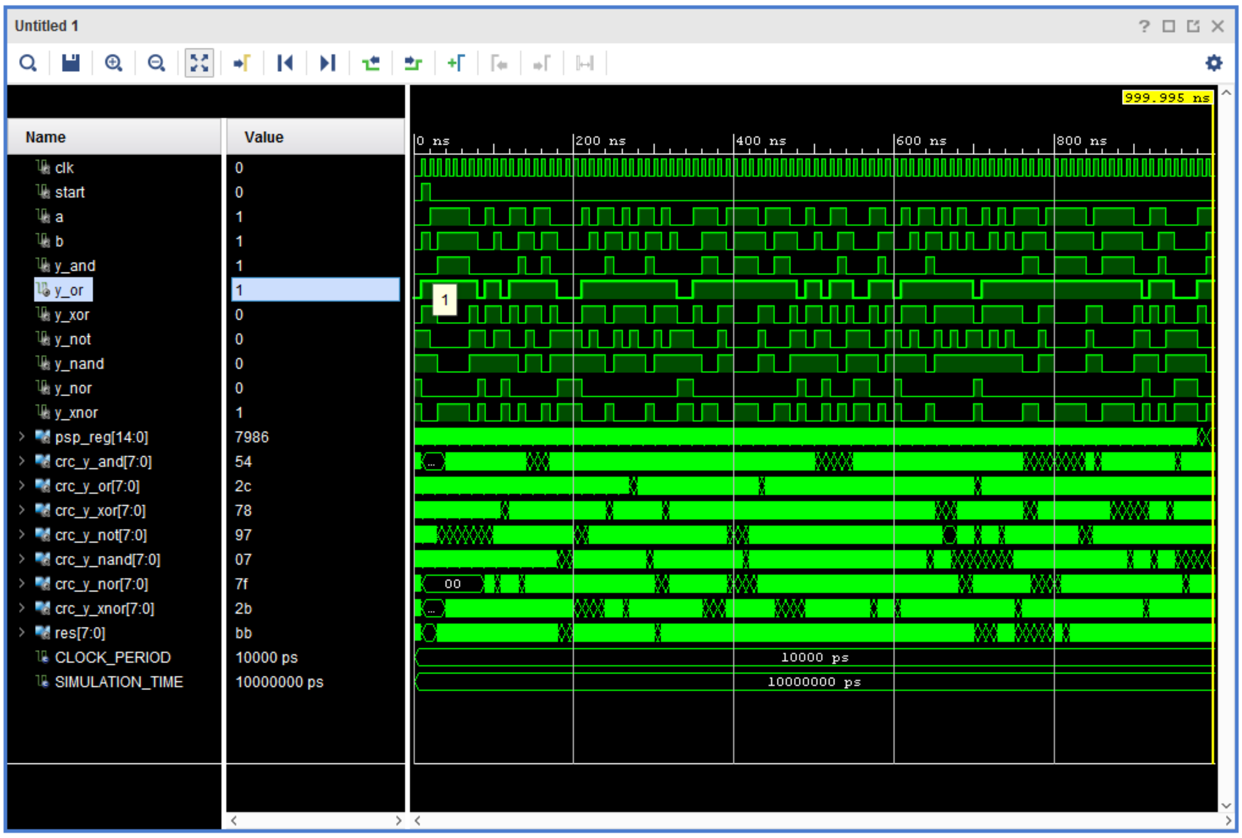
At first, choose the entity, which you want to run for simulation in **Sources** tab in subtab **Simulation Sources**. After that, run the simulator: **Flow Navigator**: **SIMULATION=>Run Behavioral Simulation**.

At this step the Vivado simulator will automatically run and the device operation would be simulated for 1000 ns time interval.

For simulation one should create a test bench: a special VHDL/Verilog module, which instantiates the simulated module (DUT, device under test). Also the test bench generates all the stimuli signals: clocks, resets, input data. The testing inputs may be calculated inside by the test bench itself, or in another IDE (e.g., Matlab). In the latter case test vectors should be written into a file, which is read by the test bench.

Time waveforms for the signals inside the test bench and DUT appear in the time diagrams window after simulation starts.

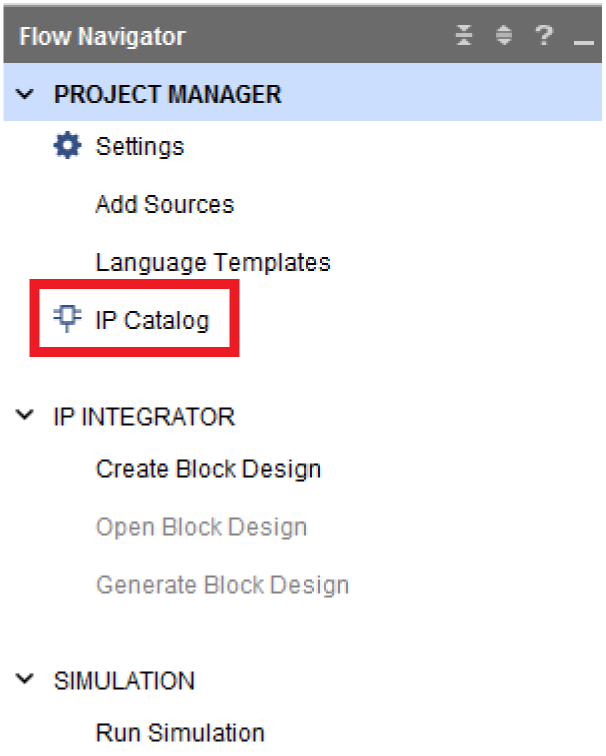




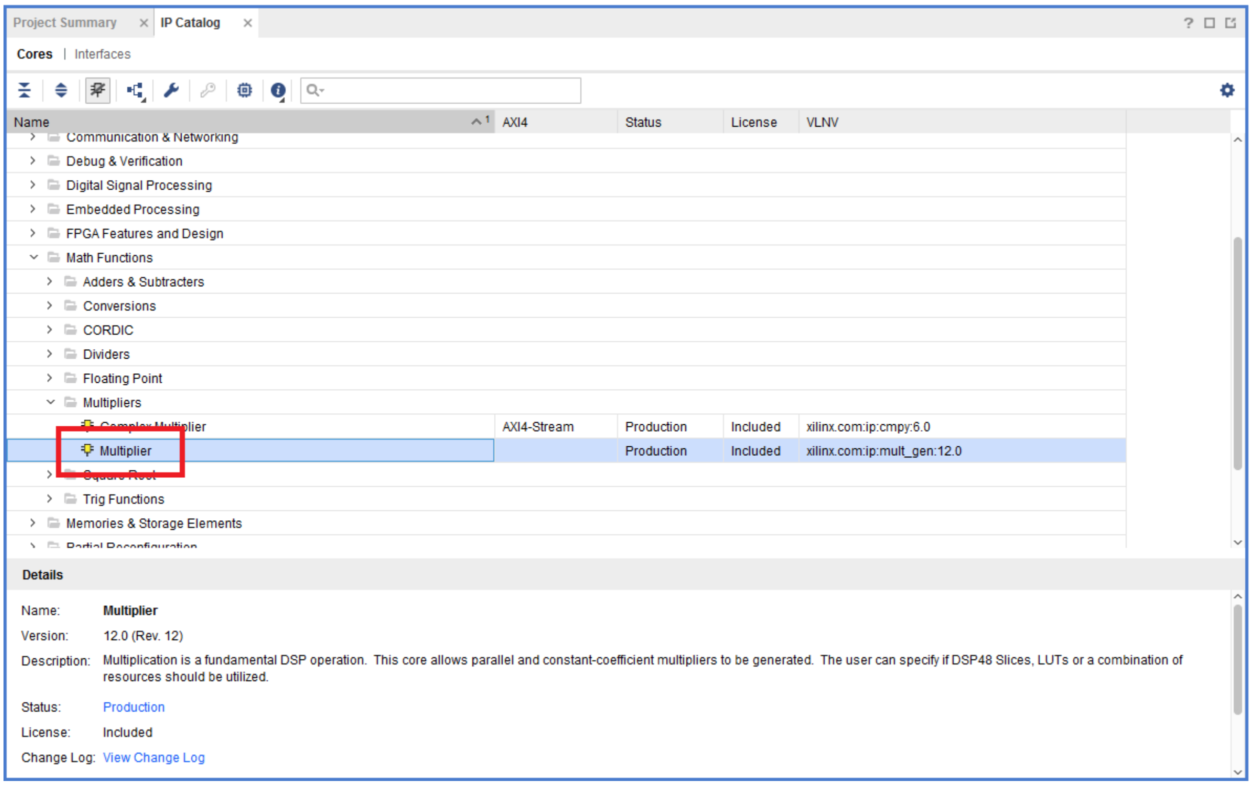
Name	Value
clk	0
start	0
a	1
b	1
y_and	1
y_or	1
y_xor	0
y_not	0
y_nand	0
y_nor	0
y_xnor	1
psp_reg[14:0]	7986
crc_y_and[7:0]	54
crc_y_or[7:0]	2c
crc_y_xor[7:0]	78
crc_y_not[7:0]	97
crc_y_nand[7:0]	07
crc_y_nor[7:0]	7f
crc_y_xnor[7:0]	2b
res[7:0]	bb
CLOCK_PERIOD	10000 ps
SIMULATION_TIME	10000000 ps

13. IP-cores instantiation

Press **IP Catalog** in the Flow Navigator to open the Vivado IP-cores repository. The list of all available IP-cores would open in the Project Manager (to the right).



Choose the required IP-core and click twice on its name to open the IP-core configurator.

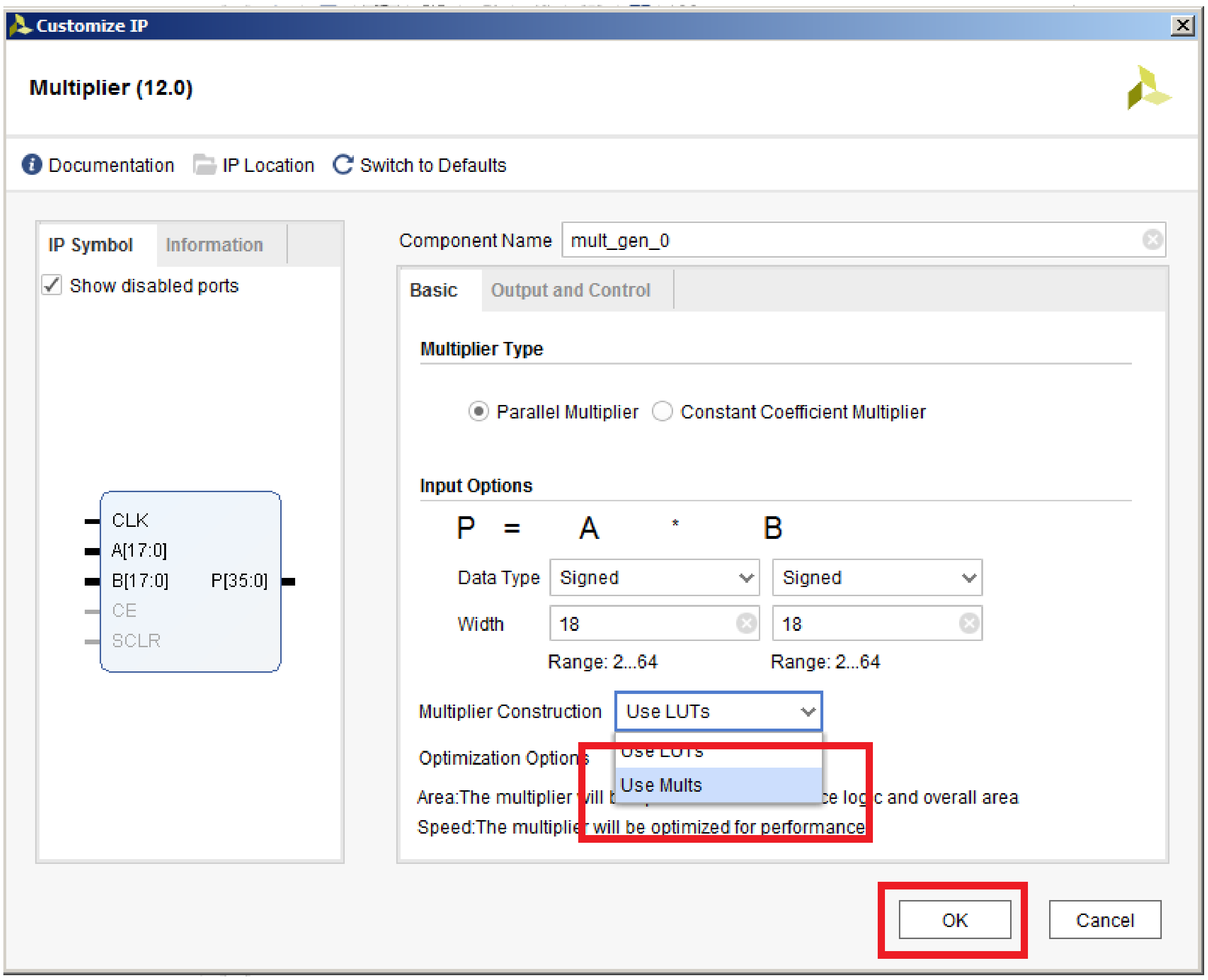


Name	AXI4	Status	License	VLNV
Communication & Networking				
Debug & Verification				
Digital Signal Processing				
Embedded Processing				
FPGA Features and Design				
Math Functions				
Adders & Subtracters				
Conversions				
CORDIC				
Dividers				
Floating Point				
Multipliers				
Complex Multiplier	AXI4-Stream	Production	Included	xilinx.com:ip:cmpy:6.0
Multiplier		Production	Included	xilinx.com:ip:mult_gen:12.0
Square Root				
Trig Functions				
Memories & Storage Elements				
Partial Reconfiguration				

Details

Name:	**Multiplier**
Version:	12.0 (Rev. 12)
Description:	Multiplication is a fundamental DSP operation. This core allows parallel and constant-coefficient multipliers to be generated. The user can specify if DSP48 Slices, LUTs or a combination of resources should be utilized.
Status:	Production
License:	Included
Change Log:	View Change Log

E.g., open the IP-core configurator for multiplier. In the **Customize IP** window specify the parameter values and the name of the IP-core. Press OK.



Component Name: mult_gen_0

Multiplier Type: [✓ Parallel Multiplier] Constant Coefficient Multiplier

Input Options

P = A * B

	A	B
Data Type	Signed	Signed
Width	18	18
	Range: 2...64	Range: 2...64

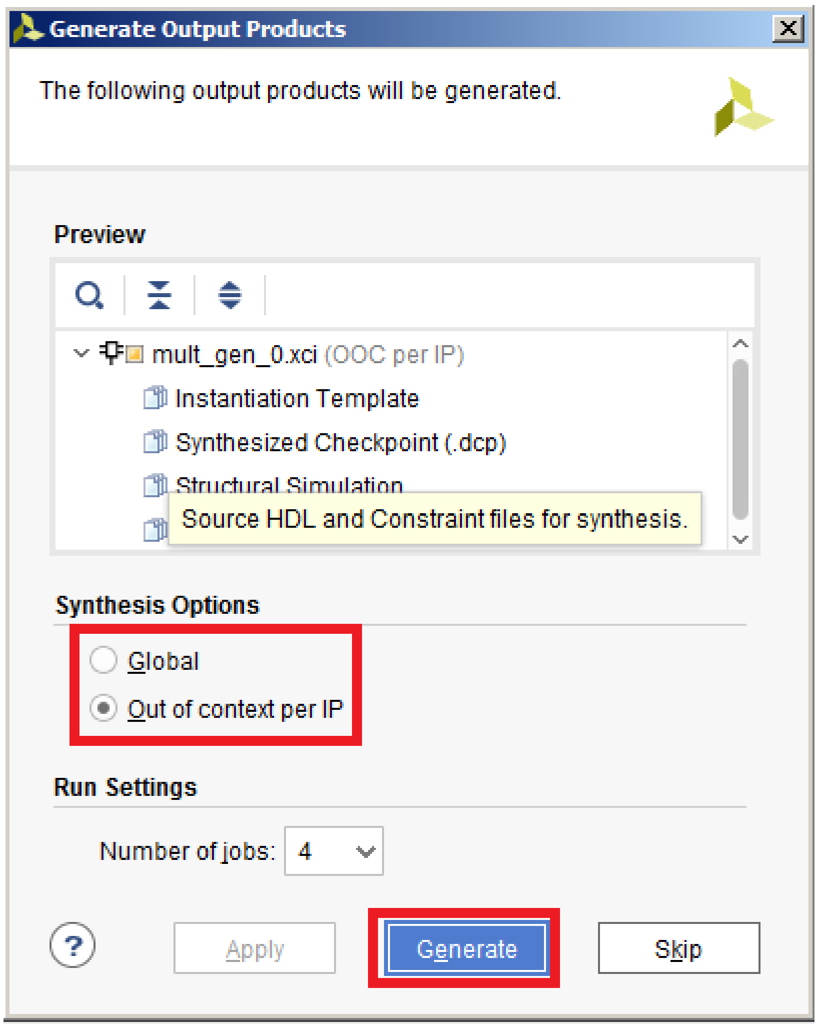
Multiplier Construction: Use LUTs (Use LUTs / Use Mults)

Optimization Options

Area:The multiplier will b... ...ce logic and overall area

Speed:The multiplier will be optimized for performance

Specify the way the core would be synthesized: during the synthesis step of the whole project (Global, the synthesis strategy is the same as for the whole project) or separately (Out of context, the synthesis strategy may differ from the one of whole project). Press **Generate**, wait until synthesis is finished (may take lots of time).


Generate Output Products

The following output products will be generated.

Preview

- mult_gen_0.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Structural Simulation

Source HDL and Constraint files for synthesis.

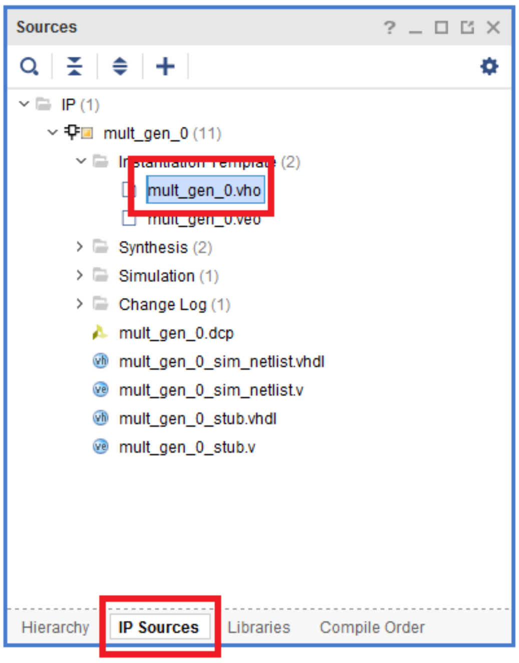
Synthesis Options

Global [✓ Out of context per IP]

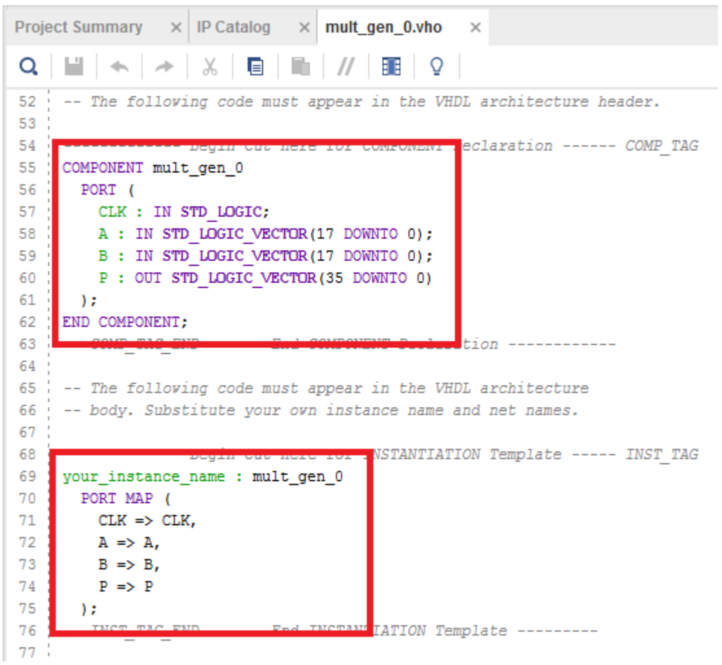
Run Settings

Number of jobs: 4

Switch to the **IP Sources** view in the **Sources** tab. Find the generated files. Of special interest are instantiation templates – the templates for IP core instantiation into another module (in VHDL and Verilog languages).


Sources

- IP (1)
  - mult_gen_0 (11)
    - Instantiation Template (2)
      - mult_gen_0.vho
      - mult_gen_0.veo
    - Synthesis (2)
    - Simulation (1)
    - Change Log (1)
    - mult_gen_0.dcp
    - mult_gen_0_sim_netlist.vhdl
    - mult_gen_0_sim_netlist.v
    - mult_gen_0_stub.vhdl
    - mult_gen_0_stub.v



```
-- The following code must appear in the VHDL architecture header.

------------- Begin Cut here for COMPONENT Declaration ------ COMP_TAG
COMPONENT mult_gen_0
  PORT (
    CLK : IN STD_LOGIC;
    A : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
    B : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
    P : OUT STD_LOGIC_VECTOR(35 DOWNTO 0)
  );
END COMPONENT;
-- COMP_TAG_END ------ End COMPONENT Declaration ------------

-- The following code must appear in the VHDL architecture
-- body. Substitute your own instance name and net names.

------------- Begin Cut here for INSTANTIATION Template ----- INST_TAG
your_instance_name : mult_gen_0
  PORT MAP (
    CLK => CLK,
    A => A,
    B => B,
    P => P
  );
-- INST_TAG_END ------ End INSTANTIATION Template ---------
```