Xilinx Vivado Basics

Ahmet Can Mert
ahmet.mert@iaik.tugraz.at

Overview

- Xilinx Vivado tool is a software for simulation, synthesis, implementation and analysis of HDL designs for Xilinx FPGAs.
  - HW development

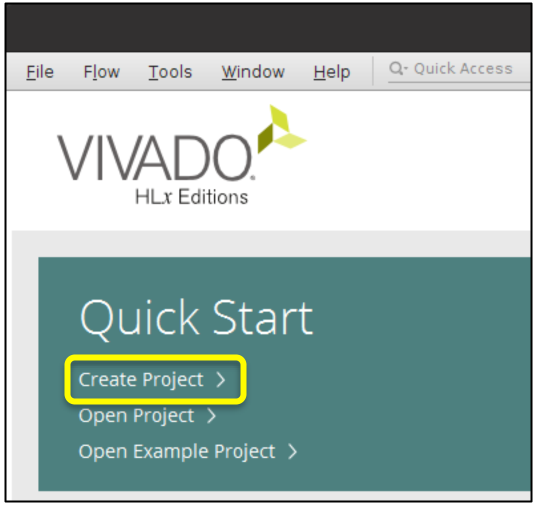
- In this tutorial, you will only focus on HW development:
  - Xilinx Vivado interface
  - How to create a project
  - How to add/create design and simulation files
  - How to run simulation/debug
  - How to perform synthesis/implementation (with constraint files)
  - How to add Xilinx IPs to your project

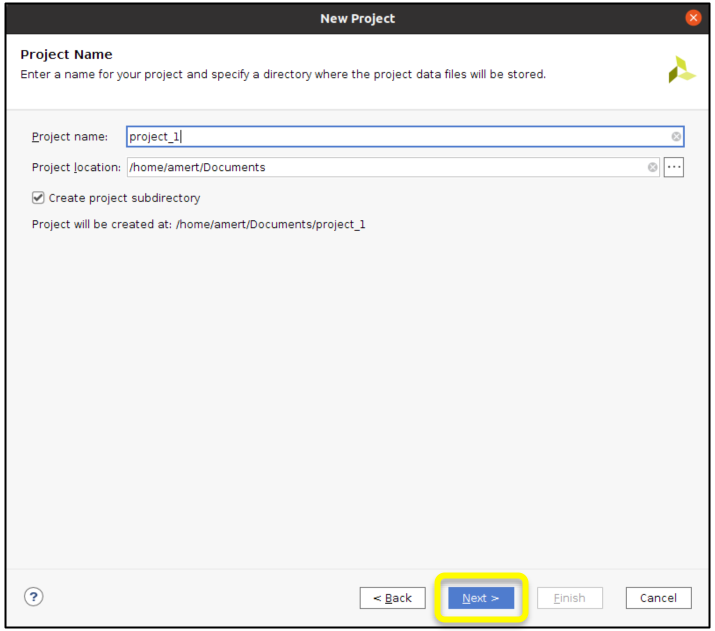
- Vivado Design Suite User Guide: https://docs.xilinx.com/r/2021.1-English/ug893-vivado-ide/Introduction

Creating a New Project

Creating a New Project

- Launch Xilinx Vivado and click on *Create Project* in Quick Start tab (or click on *File -> Project -> New*). Set your project name and location, then click on *Next*.
  - You can also open an existing project by clicking on *Open Project* in Quick Start tab (or click on *File -> Project -> Open*), then browsing and selecting *project_name.xpr* file.




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_1

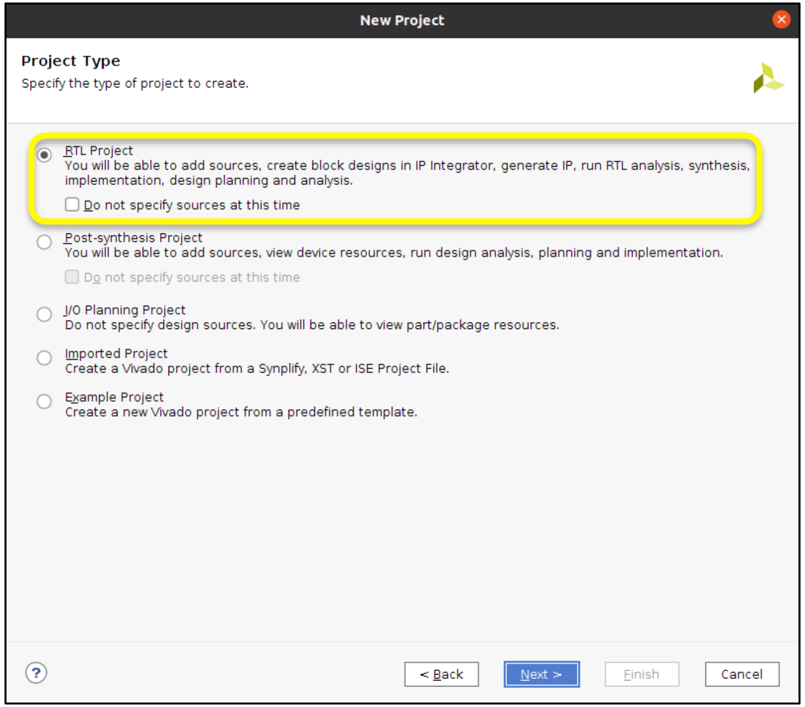
Project location: /home/amert/Documents

☑ Create project subdirectory

Project will be created at: /home/amert/Documents/project_1

Creating a New Project

- Select project type as *RTL Project* and click on Next. In the next step, you can either add design sources/constraint files by clicking on *Create File* button or you can skip this step by clicking on *Next* (you can create files after you created the project).



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

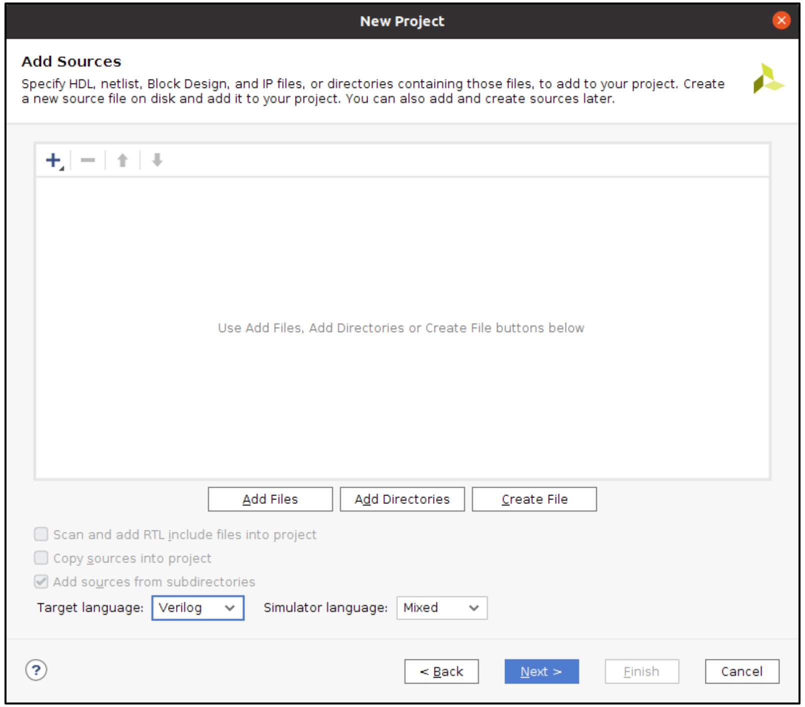
Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.

☐ 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.



New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Use Add Files, Add Directories or Create File buttons below

☐ Scan and add RTL include files into project

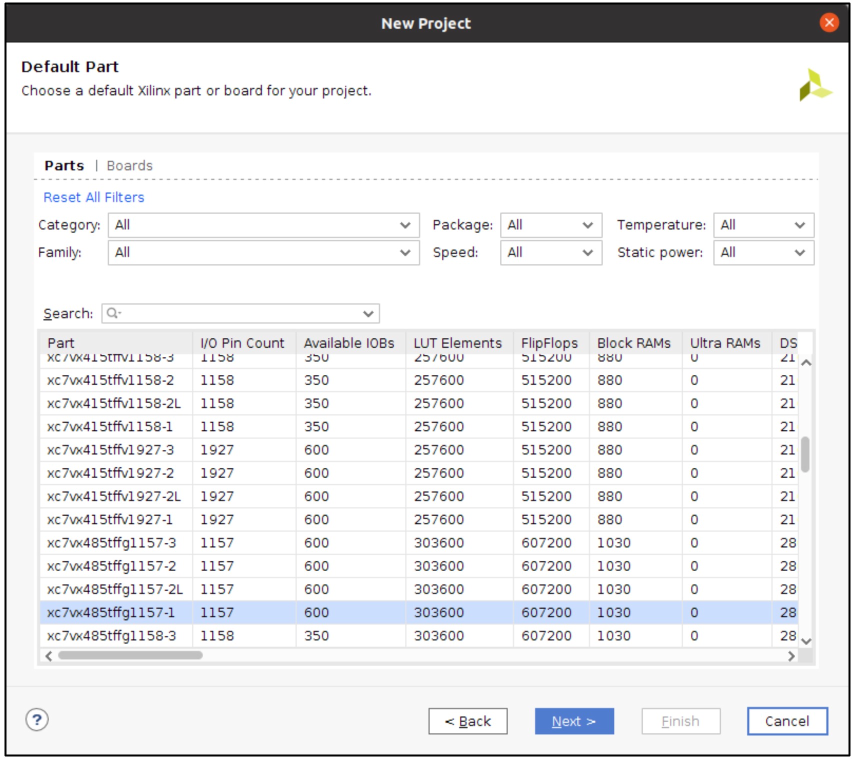
☐ Copy sources into project

☑ Add sources from subdirectories

Target language: Verilog   Simulator language: Mixed

Creating a New Project

- You should select an *FPGA device* or *Board* for your project. In order to select PYNQ-Z2 board, switch to *Boards* tab and select *pynq-z2*. Then click on *Next* and *Finish*.



New Project

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

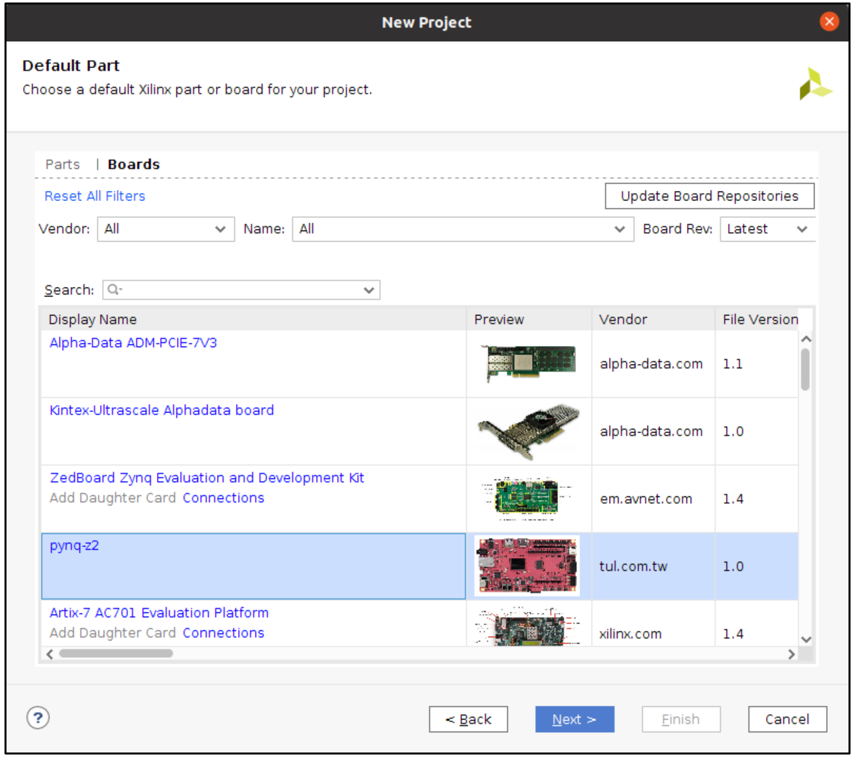
Parts | Boards

Reset All Filters

Category: All
Family: All
Package: All
Speed: All
Temperature: All
Static power: All

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DS
xc7vx415tffv1158-3	1158	350	257600	515200	880	0	21
xc7vx415tffv1158-2	1158	350	257600	515200	880	0	21
xc7vx415tffv1158-2L	1158	350	257600	515200	880	0	21
xc7vx415tffv1158-1	1158	350	257600	515200	880	0	21
xc7vx415tffv1927-3	1927	600	257600	515200	880	0	21
xc7vx415tffv1927-2	1927	600	257600	515200	880	0	21
xc7vx415tffv1927-2L	1927	600	257600	515200	880	0	21
xc7vx415tffv1927-1	1927	600	257600	515200	880	0	21
xc7vx485tffg1157-3	1157	600	303600	607200	1030	0	28
xc7vx485tffg1157-2	1157	600	303600	607200	1030	0	28
xc7vx485tffg1157-2L	1157	600	303600	607200	1030	0	28
xc7vx485tffg1157-1	1157	600	303600	607200	1030	0	28
xc7vx485tffg1158-3	1158	350	303600	607200	1030	0	28



New Project

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

Parts | **Boards**

Reset All Filters

Update Board Repositories

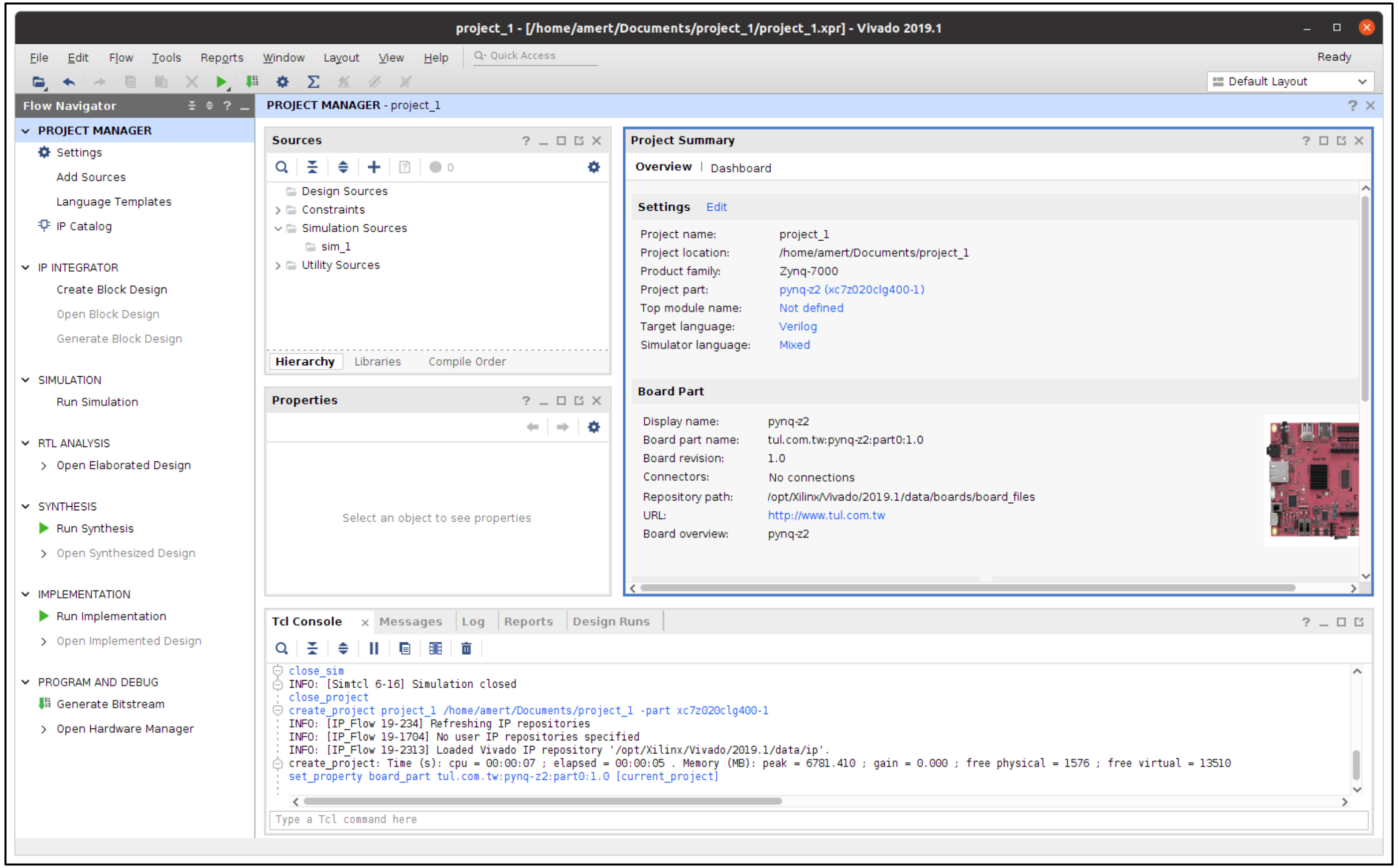
Vendor: All
Name: All
Board Rev: Latest

Search:

Display Name	Preview	Vendor	File Version
Alpha-Data ADM-PCIE-7V3		alpha-data.com	1.1
Kintex-Ultrascale Alphadata board		alpha-data.com	1.0
ZedBoard Zynq Evaluation and Development Kit Add Daughter Card Connections		em.avnet.com	1.4
pynq-z2		tul.com.tw	1.0
Artix-7 AC701 Evaluation Platform Add Daughter Card Connections		xilinx.com	1.4

Vivado Interface

Vivado Interface



Settings Edit

Project name:	project_1
Project location:	/home/amert/Documents/project_1
Product family:	Zynq-7000
Project part:	pynq-z2 (xc7z020clg400-1)
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

Board Part

Display name:	pynq-z2
Board part name:	tul.com.tw:pynq-z2:part0:1.0
Board revision:	1.0
Connectors:	No connections
Repository path:	/opt/Xilinx/Vivado/2019.1/data/boards/board_files
URL:	http://www.tul.com.tw
Board overview:	pynq-z2

```
close_sim
INFO: [Simtcl 6-16] Simulation closed
close_project
create_project project_1 /home/amert/Documents/project_1 -part xc7z020clg400-1
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository '/opt/Xilinx/Vivado/2019.1/data/ip'.
create_project: Time (s): cpu = 00:00:07 ; elapsed = 00:00:05 . Memory (MB): peak = 6781.410 ; gain = 0.000 ; free physical = 1576 ; free virtual = 13510
set_property board_part tul.com.tw:pynq-z2:part0:1.0 [current_project]
```

Vivado Interface

Project Settings

Runs Behavioral Simulation

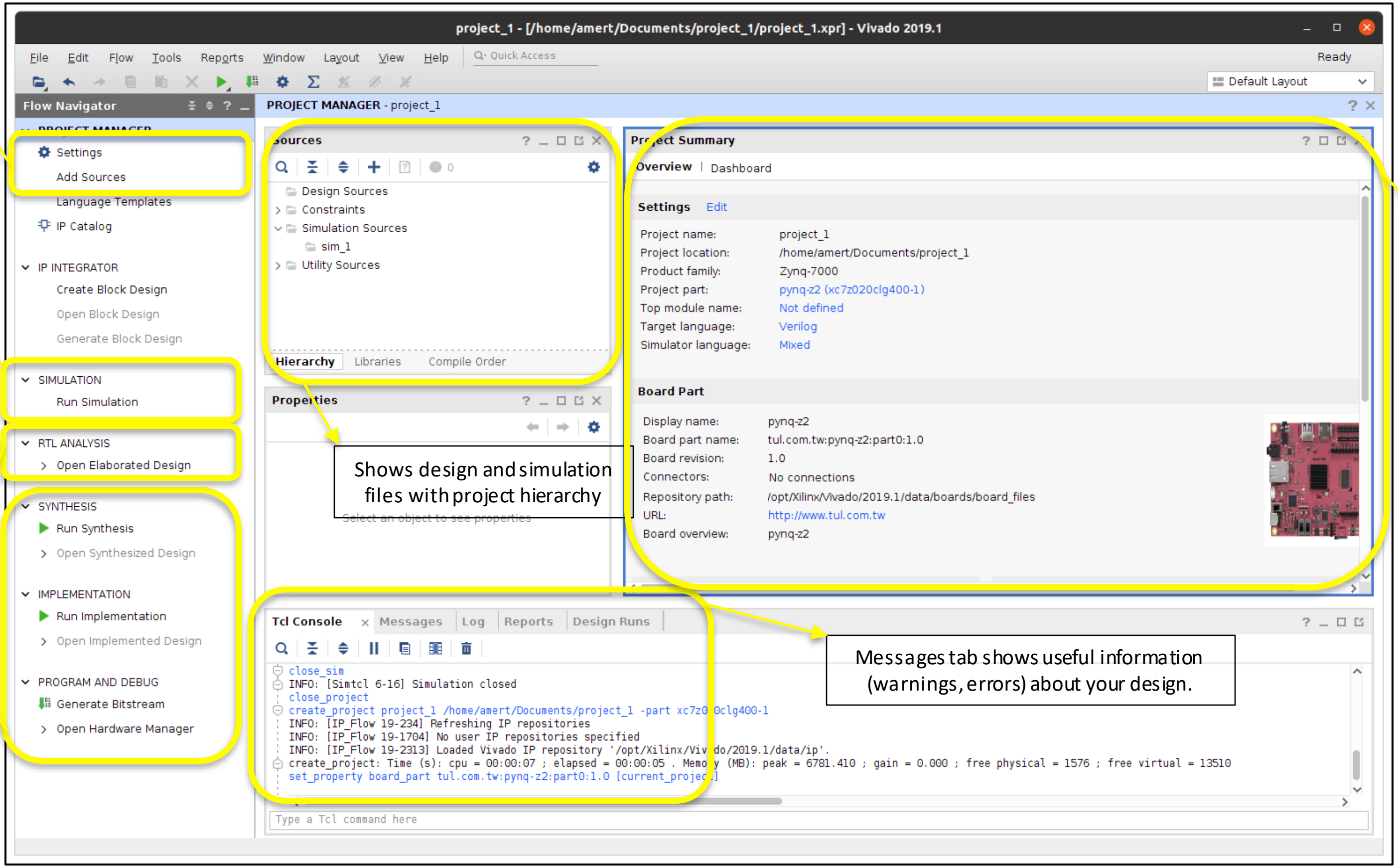
Compiles RTL code and loads RTL netlist

Implementation run and settings

Shows design and simulation files with project hierarchy

Summary of project and results

Messages tab shows useful information (warnings, errors) about your design.



Settings

Project name:	project_1
Project location:	/home/amert/Documents/project_1
Product family:	Zynq-7000
Project part:	pynq-z2 (xc7z020clg400-1)
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

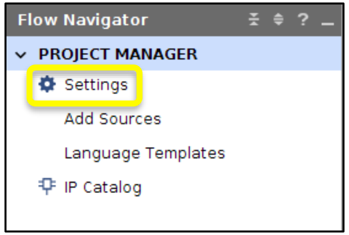
Board Part

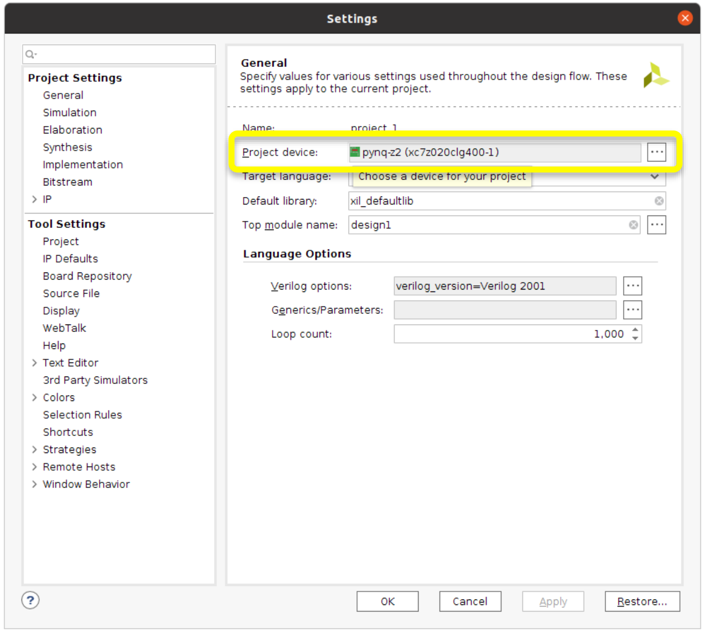
Display name:	pynq-z2
Board part name:	tul.com.tw:pynq-z2:part0:1.0
Board revision:	1.0
Connectors:	No connections
Repository path:	/opt/Xilinx/Vivado/2019.1/data/boards/board_files
URL:	http://www.tul.com.tw
Board overview:	pynq-z2

```
close_sim
INFO: [Simtcl 6-16] Simulation closed
close_project
create_project project_1 /home/amert/Documents/project_1 -part xc7z020clg400-1
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository '/opt/Xilinx/Vivado/2019.1/data/ip'.
create_project: Time (s): cpu = 00:00:07 ; elapsed = 00:00:05 . Memory (MB): peak = 6781.410 ; gain = 0.000 ; free physical = 1576 ; free virtual = 13510
set_property board_part tul.com.tw:pynq-z2:part0:1.0 [current_project]
```

Vivado Interface

- From *Settings* under *PROJECT MANAGER* tab, you can change the settings of your project (e.g., Board or FPGA).





General
Specify values for various settings used throughout the design flow. These settings apply to the current project.

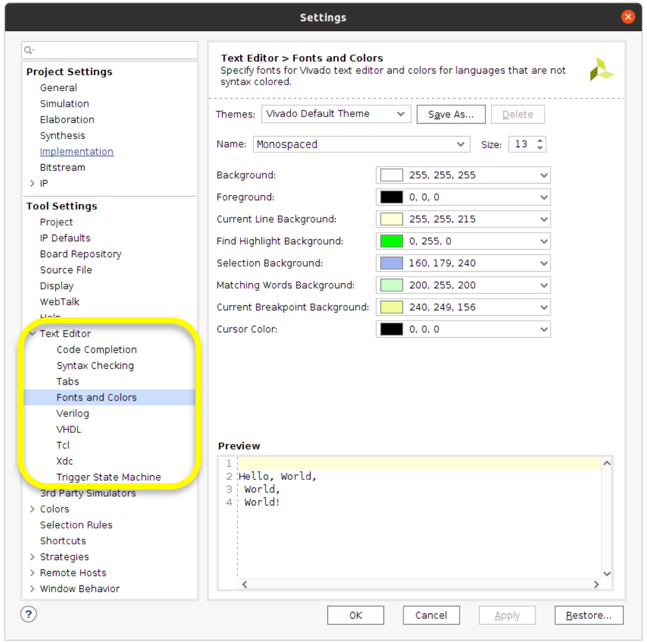
Name:	project_1
Project device:	pynq-z2 (xc7z020clg400-1)
Target language:	Choose a device for your project
Default library:	xil_defaultlib
Top module name:	design1

Language Options

Verilog options:	verilog_version=Verilog 2001
Generics/Parameters:	
Loop count:	1,000

Vivado Interface

- From *Settings* under *PROJECT MANAGER* tab, you can change the settings of your project (e.g., Board or FPGA).
  - For example, you can change the font type and size of the text editor.



Settings

Project Settings
- General
- Simulation
- Elaboration
- Synthesis
- Implementation
- Bitstream
- IP

Tool Settings
- Project
- IP Defaults
- Board Repository
- Source File
- Display
- WebTalk
- Help
- Text Editor
  - Code Completion
  - Syntax Checking
  - Tabs
  - Fonts and Colors
  - Verilog
  - VHDL
  - Tcl
  - Xdc
  - Trigger State Machine
- 3rd Party Simulators
- Colors
- Selection Rules
- Shortcuts
- Strategies
- Remote Hosts
- Window Behavior

Text Editor > Fonts and Colors
Specify fonts for Vivado text editor and colors for languages that are not syntax colored.

Themes: Vivado Default Theme | Save As... | Delete

Name: Monospaced Size: 13

Background:	255, 255, 255
Foreground:	0, 0, 0
Current Line Background:	255, 255, 215
Find Highlight Background:	0, 255, 0
Selection Background:	160, 179, 240
Matching Words Background:	200, 255, 200
Current Breakpoint Background:	240, 249, 156
Cursor Color:	0, 0, 0

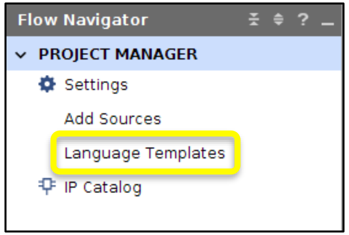
Preview

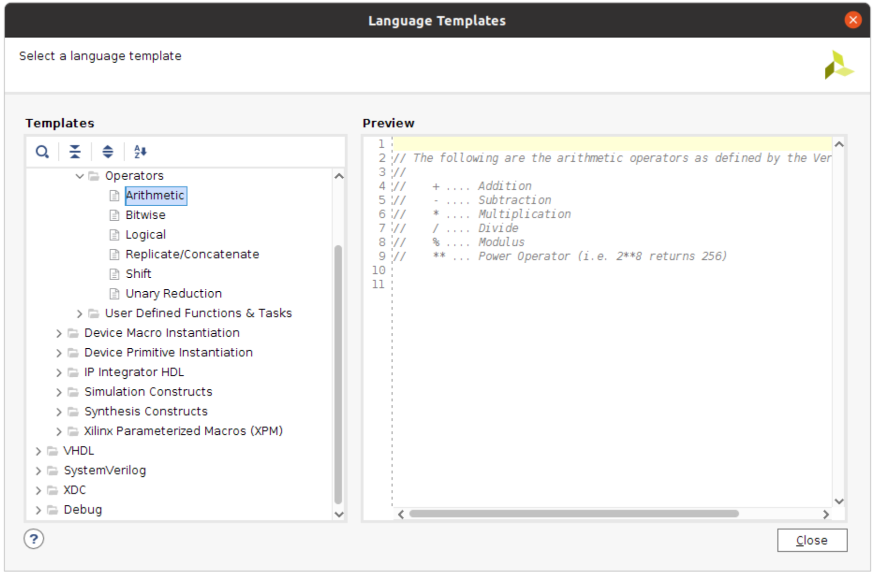
```

Hello, World,
 World,
 World!
```

OK | Cancel | Apply | Restore...

Vivado Interface

- *Language Templates* under *PROJECT MANAGER* tab shows useful Verilog and Xilinx IP constructions.

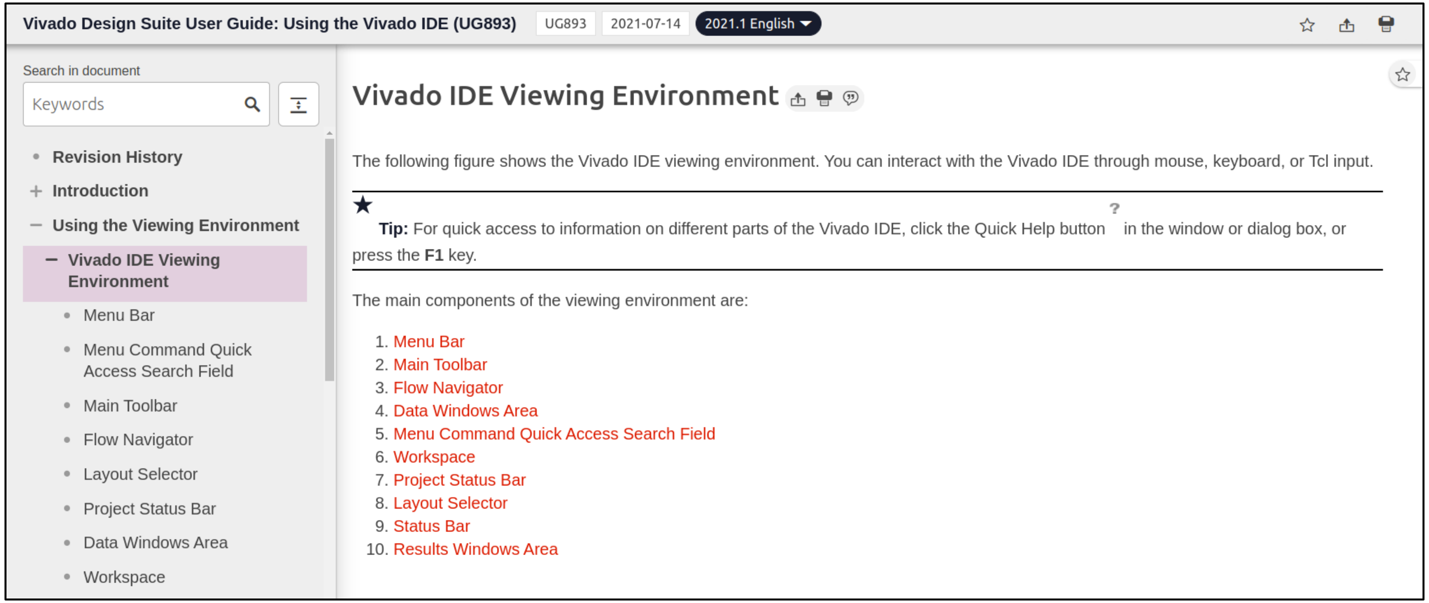




```

// The following are the arithmetic operators as defined by the Ver
//
//    + .... Addition
//    - .... Subtraction
//    * .... Multiplication
//    / .... Divide
//    % .... Modulus
//    ** ... Power Operator (i.e. 2**8 returns 256)


```

Vivado Interface

- See *Vivado IDE Viewing Enviroment* chapter of *Vivado Design Suite User Guide*[1].



Vivado Design Suite User Guide: Using the Vivado IDE (UG893) UG893 2021-07-14 2021.1 English

Search in document

Keywords

- Revision History
- Introduction
- Using the Viewing Environment
  - Vivado IDE Viewing Environment
    - Menu Bar
    - Menu Command Quick Access Search Field
    - Main Toolbar
    - Flow Navigator
    - Layout Selector
    - Project Status Bar
    - Data Windows Area
    - Workspace

Vivado IDE Viewing Environment

The following figure shows the Vivado IDE viewing environment. You can interact with the Vivado IDE through mouse, keyboard, or Tcl input.

★ **Tip:** For quick access to information on different parts of the Vivado IDE, click the Quick Help button ? in the window or dialog box, or press the **F1** key.

The main components of the viewing environment are:

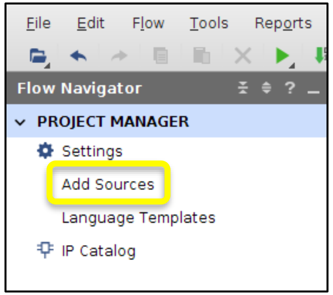
1. Menu Bar
2. Main Toolbar
3. Flow Navigator
4. Data Windows Area
5. Menu Command Quick Access Search Field
6. Workspace
7. Project Status Bar
8. Layout Selector
9. Status Bar
10. Results Windows Area

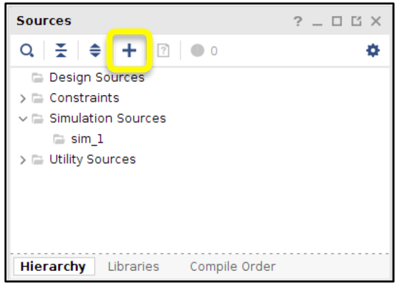
[1] https://docs.xilinx.com/r/2021.1-English/ug893-vivado-ide/Vivado-IDE-Viewing-Environment

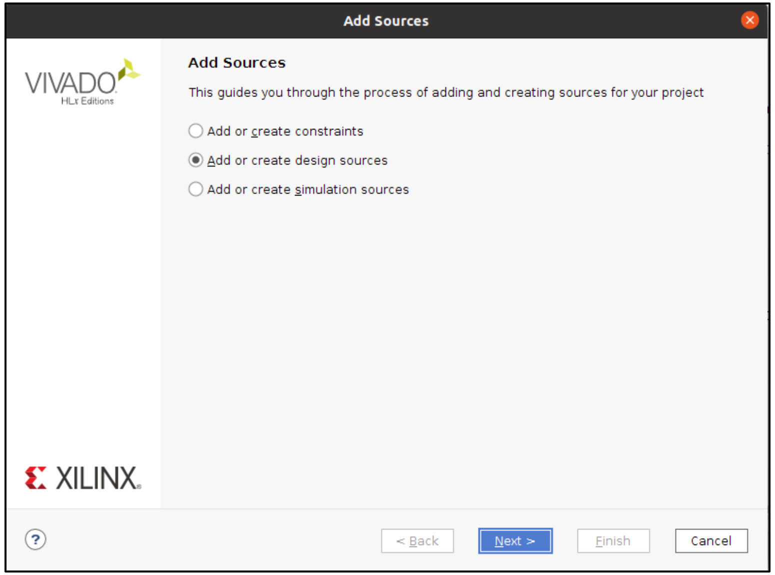
Adding/Creating Design and Simulation Files

Create a Design Source

- For your design, you should add a design source (*.v file). Click on *Add Sources* in Flow Navigator Window or "+" button on Sources window or *File -> Add Sources* on menu bar, select *Add or create design resources* and then click on Next.





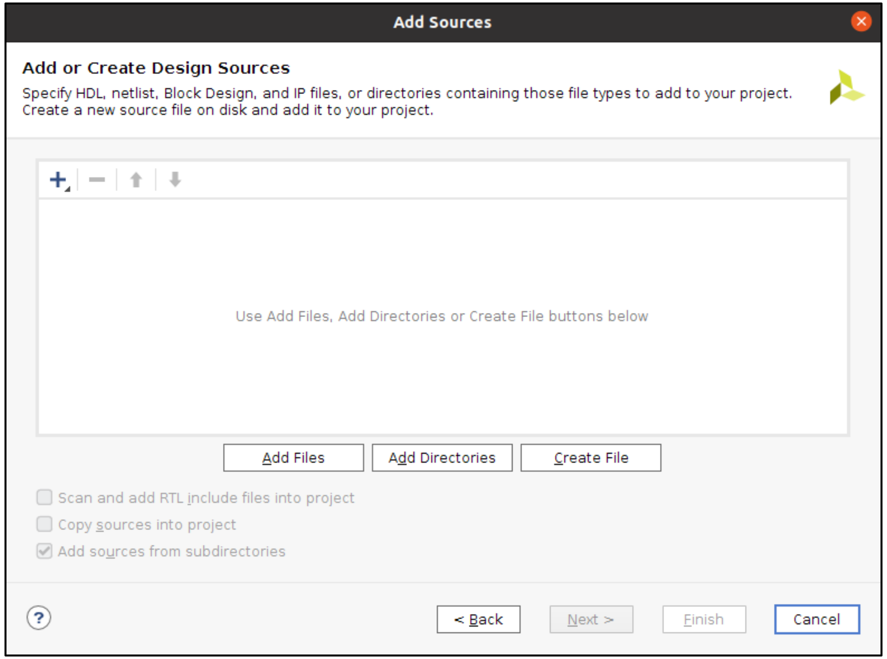

Add Sources

This guides you through the process of adding and creating sources for your project

- ( ) Add or create constraints
- (●) Add or create design sources
- ( ) Add or create simulation sources

Create a Design Source

- Click on *Create File* button, name your source file (e.g., design1.v) and click on *Next*.
    - If you want to add an existing file to your project, click on *Add Files* and select the files that you want to add to the project.

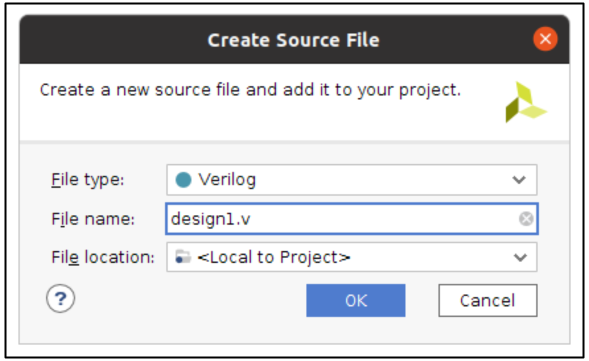

Add Sources

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.

Use Add Files, Add Directories or Create File buttons below

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [x] Add sources from subdirectories

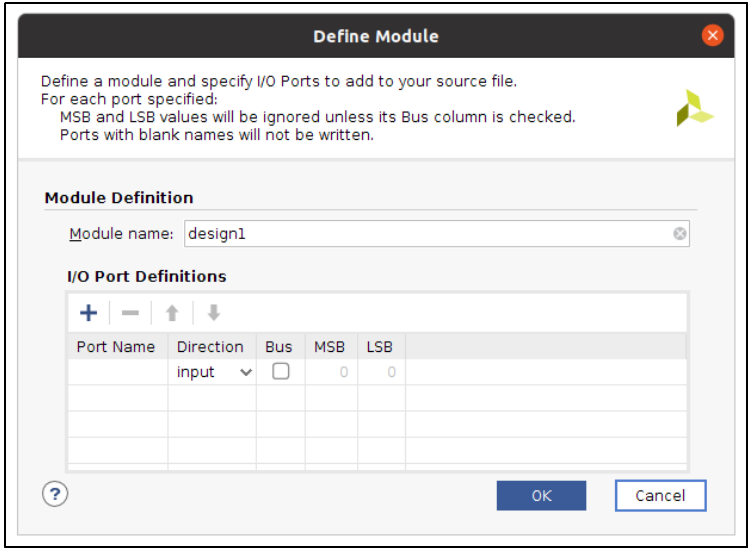

Create Source File

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

File type: Verilog
File name: design1.v
File location: <Local to Project>

Create a Design Source

- Set your module name (it is set as source file name per default) and click on OK. Then, you will see your source file under Design Sources tab on Sources window.



Define Module

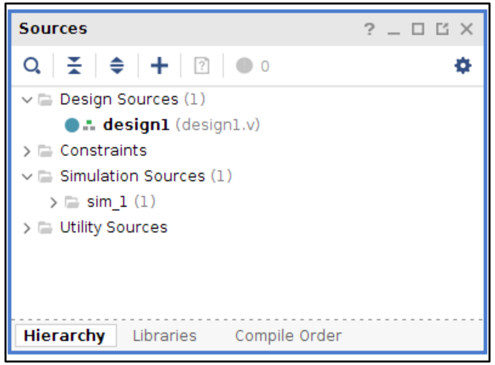
Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Module name: design1

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
	input	☐	0	0



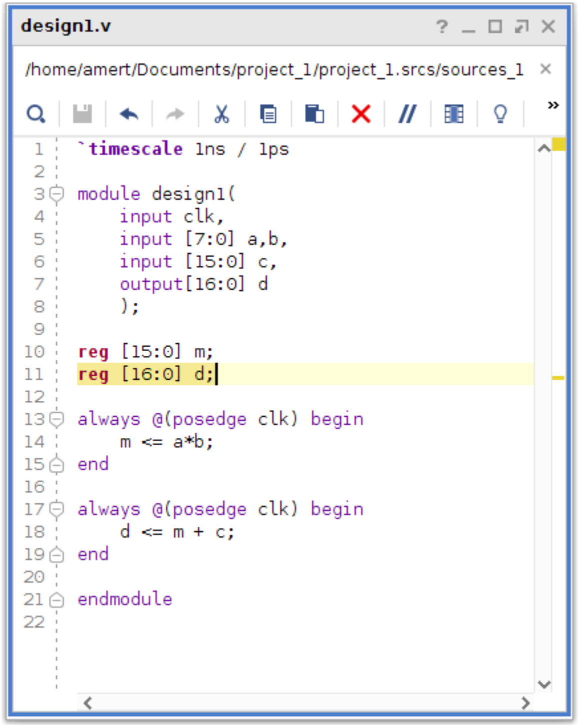
Sources

- Design Sources (1)
  - design1 (design1.v)
- Constraints
- Simulation Sources (1)
  - sim_1 (1)
- Utility Sources

Create a Design Source

- Open design1.v and write the following (or any other) Verilog code as an example and save it.

```verilog
`timescale 1ns / 1ps

module design1(
    input clk,
    input [7:0] a,b,
    input [15:0] c,
    output[16:0] d
    );

reg [15:0] m;
reg [16:0] d;

always @(posedge clk) begin
    m <= a*b;
end

always @(posedge clk) begin
    d <= m + c;
end

endmodule
```

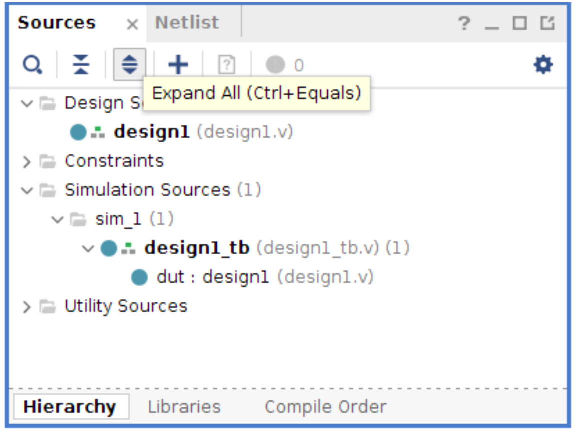

design1.v

/home/amert/Documents/project_1/project_1.srcs/sources_1

```verilog
`timescale 1ns / 1ps

module design1(
    input clk,
    input [7:0] a,b,
    input [15:0] c,
    output[16:0] d
    );

reg [15:0] m;
reg [16:0] d;

always @(posedge clk) begin
    m <= a*b;
end

always @(posedge clk) begin
    d <= m + c;
end

endmodule

```

Create a Simulation Source

- In order to test your design, you should add a simulation source. Click on *Add Sources* in Flow Navigator Window, select *Add or create simulation resources* and then click on Next.
- Click on *Create File* button, name your simulation file (e.g., design1_tb.v) and click on *Next*.
- Set your module name (it is set as simulation file name per default) and click on OK. Then, you will see your simulation file under Simulation Sources tab on Sources window.



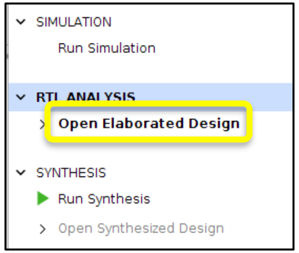
Create a Simulation Source

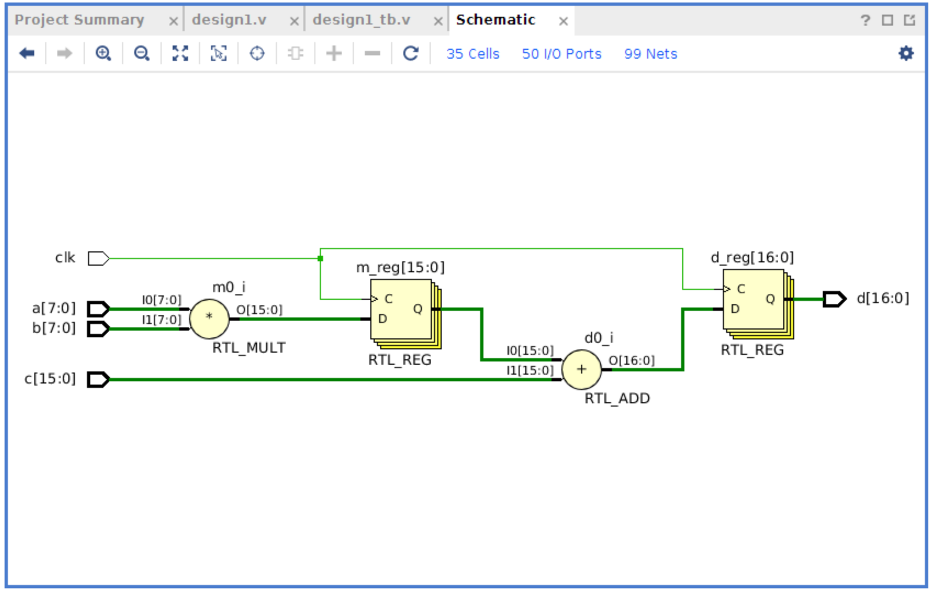
- Open design1_tb.v and write the following example Verilog testbench code and save it.

```
` timescale 1ns / 1ps

module design1_tb();

reg [7:0] a,b;
reg [15:0] c;
wire [16:0] d;

design1 dut (a,b,c,d);

initial begin
      a=0; b=0; c=0;
      #10;
      a=10; b=40; c=25;
      #10;
      a=53; b=19; c=100;
end

endmodule
```

Design Elaboration

- Design Elaboration and RTL Analysis
  - It compiles RTL code and loads RTL netlist
  - You can check RTL structure, syntax, and logic definitions
  - You can view the schematic of your design

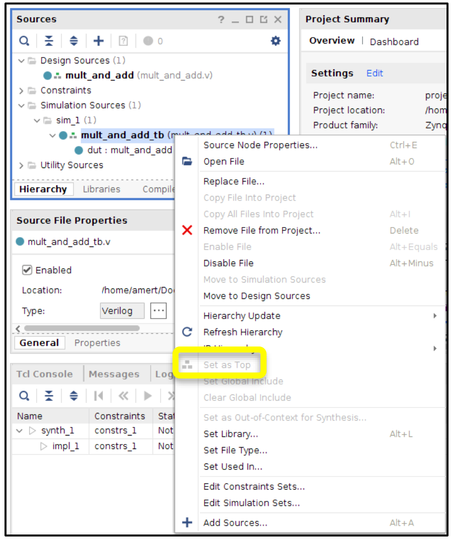


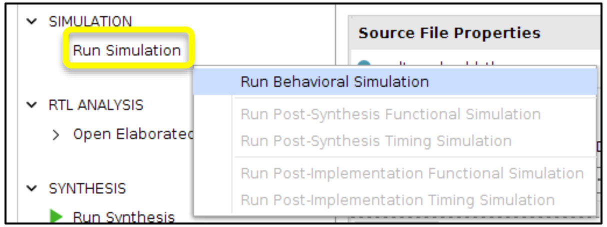


Running Behavioral Simulation

Behavioral Simulation

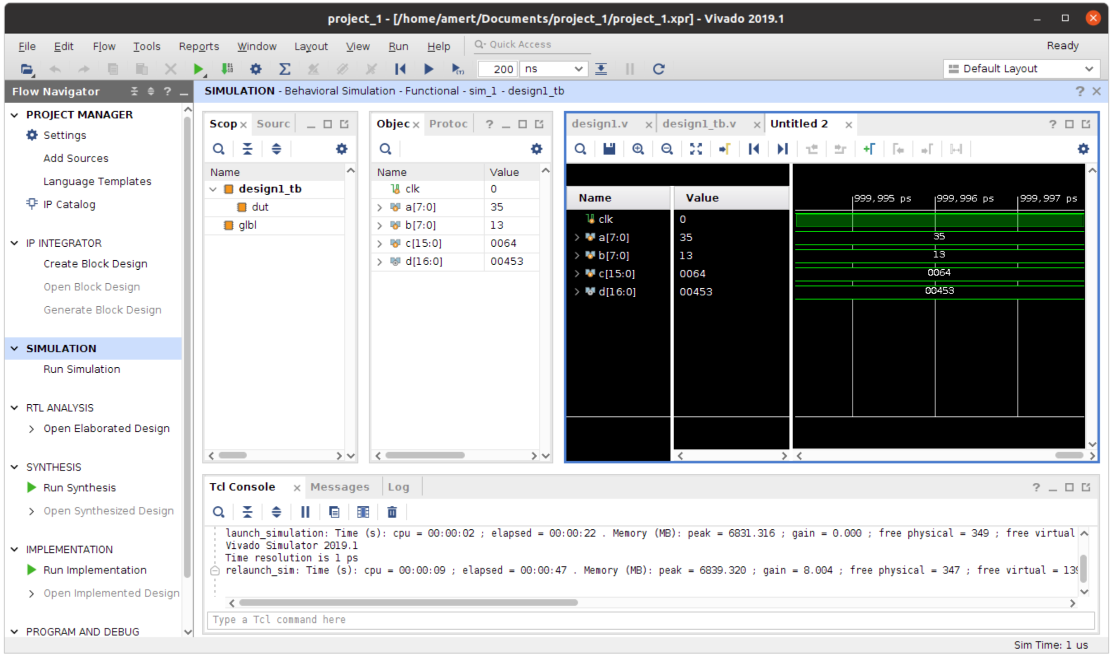
- Before running simulation, first make sure that your simulation source (testbench) is selected as top simulation module (its module name should be in boldcase letters) under Simulation Sources tab on Sources window.
- If it is not, right click on its name and click on *Set as Top*.
- Then, click on *Run Simulation* in Flow Navigator Window and click *Run Behavioral Simulation*.




SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated
SYNTHESIS
Run Synthesis
Source File Properties
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation

Behavioral Simulation

- If there is no error in your design and testbench files, you will see the following simulation screen.



Name	Value
clk	0
a[7:0]	35
b[7:0]	13
c[15:0]	0064
d[16:0]	00453

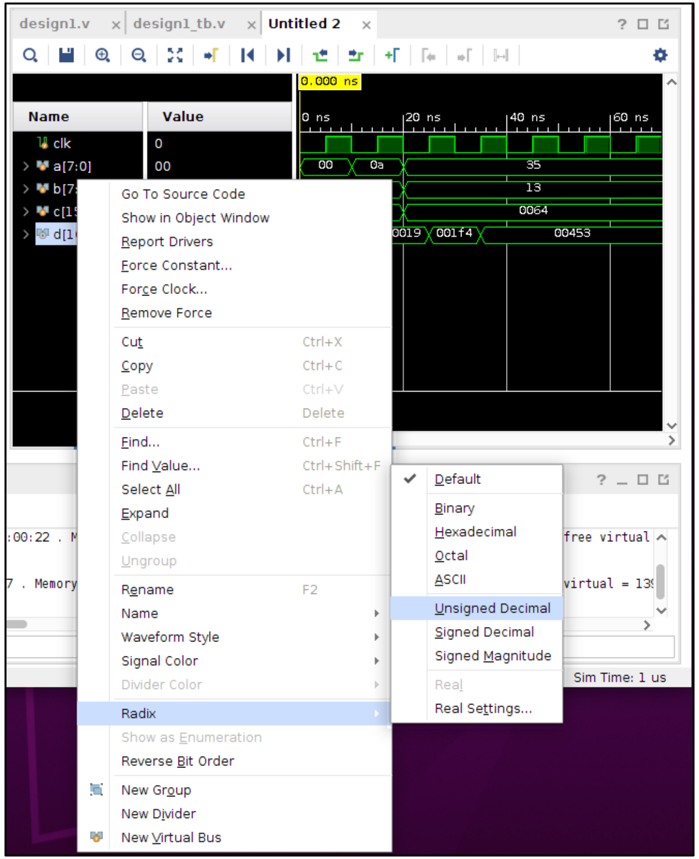
```
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:22 . Memory (MB): peak = 6831.316 ; gain = 0.000 ; free physical = 349 ; free virtual
Vivado Simulator 2019.1
Time resolution is 1 ps
relaunch_sim: Time (s): cpu = 00:00:09 ; elapsed = 00:00:47 . Memory (MB): peak = 6839.320 ; gain = 8.004 ; free physical = 347 ; free virtual = 13
```

💾	Save waveform configuration
🔍+	Zoom in to area where cursor is
🔍−	Zoom out from area where cursor is
⛶	Fit whole run into waveform
⏮	Go to time=0
⏭	Go to the last time

Behavioral Simulation

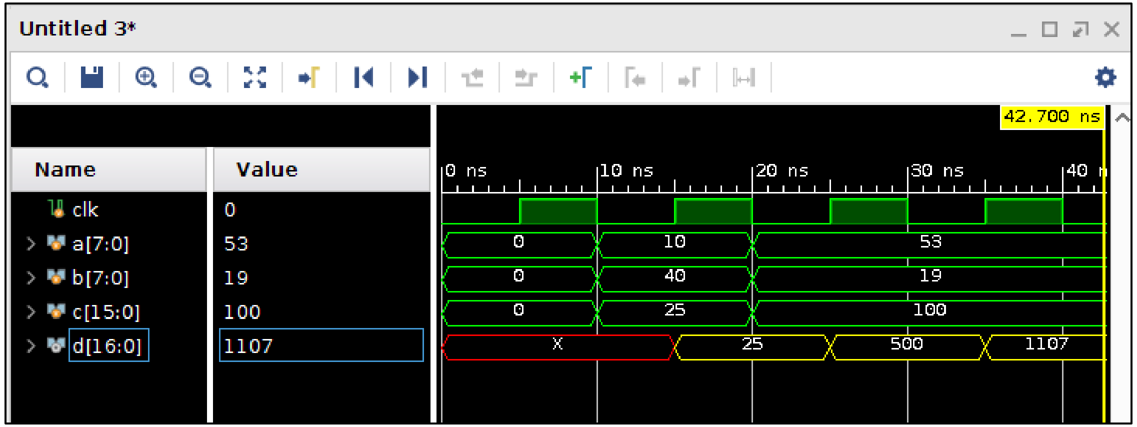
- You can change radix of the signals in the waveform window. For example, right click on signal name and then select *Radix -> Unsigned decimal* to change representation of this signal from hexadecimal to unsigned decimal.

- Similarly, you can change the color of the signal. For example, right click on signal name and then select *Signal Color -> Yellow* to change the color of this signal from lime to yellow.



Behavioral Simulation

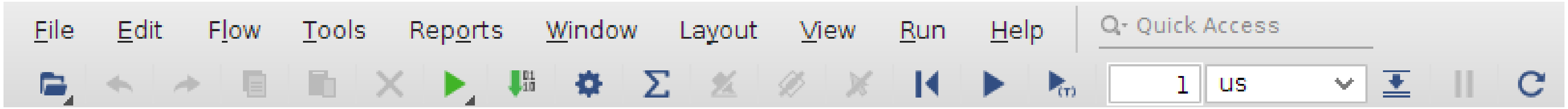
- Change radix of the signals to the unsigned decimal. Then, zoom in to the beginning of the simulation. Now you can observe output *d*.



```
Name      Value
clk       0
a[7:0]    53      0 | 10 | 53
b[7:0]    19      0 | 40 | 19
c[15:0]   100     0 | 25 | 100
d[16:0]   1107    X | 25 | 500 | 1107
```

Behavioral Simulation

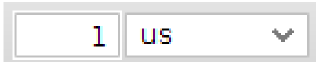
- In the toolbar on the top, you will see some shortcut buttons for simulation.



Restart the simulation. Use this to restart the simulation with current state of the design.

Run All. Use this to run simulation until it reaches a stop/finish command in testbench .

Run for specified time. Use this to run simulation for specified time.

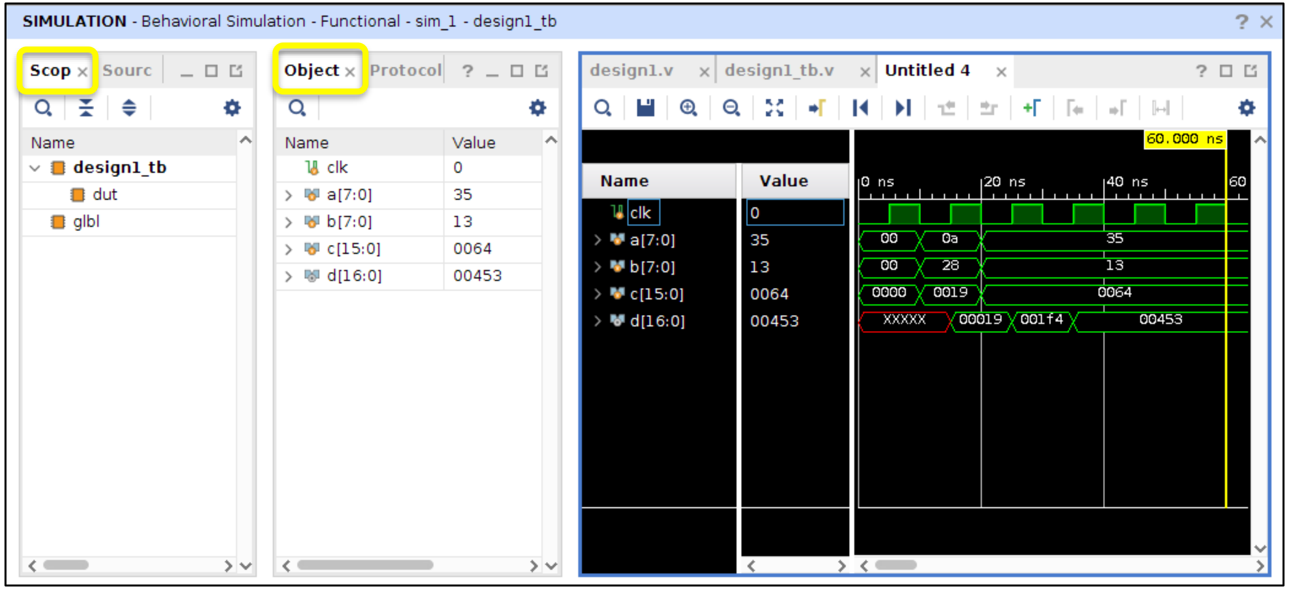


Time and unit. Use this to specify run time and unit for the command above.

Relaunch the simulation. Use this to relaunch the simulation for elaborating the changes you made in your design.

Behavioral Simulation

- In the Scope window, you can see the design hierarchy. When you select a scope in the Scope window, all HDL objects visible from that scope appear in the Objects window.



SIMULATION - Behavioral Simulation - Functional - sim_1 - design1_tb

Scope window:

Name
design1_tb
dut
glbl

Objects window:

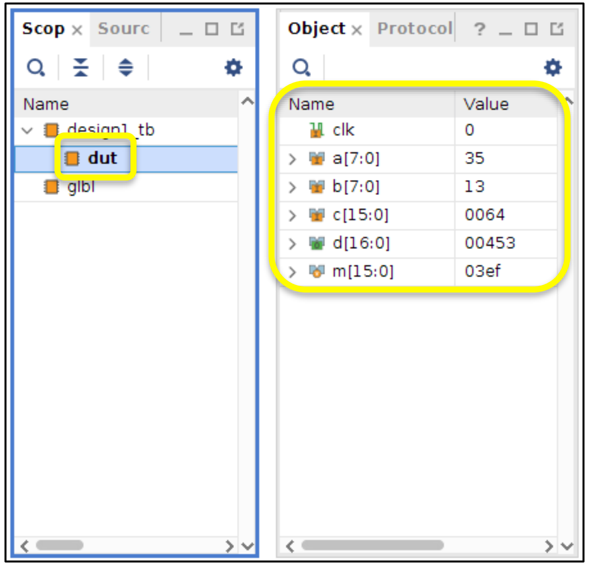
Name	Value
clk	0
a[7:0]	35
b[7:0]	13
c[15:0]	0064
d[16:0]	00453

Waveform window (Untitled 4):

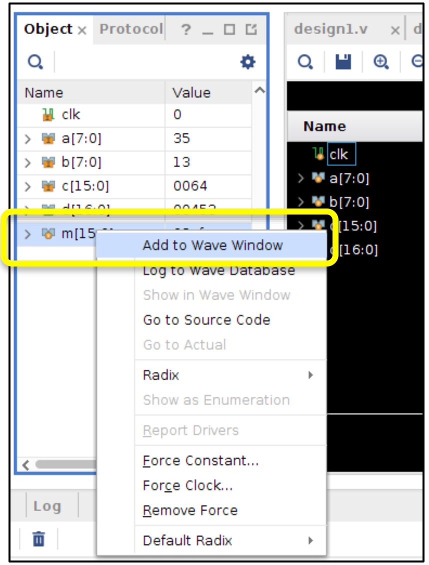
Name	Value
clk	0
a[7:0]	35
b[7:0]	13
c[15:0]	0064
d[16:0]	00453

Behavioral Simulation

- When you start simulation, you will see signals defined in testbench on the waveform. In order to observe internal signals (for debugging), click on the module you want to investigate on Scope window. Then right click on the signal you want to observe in the *Objects window* and click on *Add to wave window.*



Name	Value
clk	0
a[7:0]	35
b[7:0]	13
c[15:0]	0064
d[16:0]	00453
m[15:0]	03ef



Name	Value
clk	0
a[7:0]	35
b[7:0]	13
c[15:0]	0064
d[16:0]	00453
m[15	

Synthesis/Implementation

Constraint File

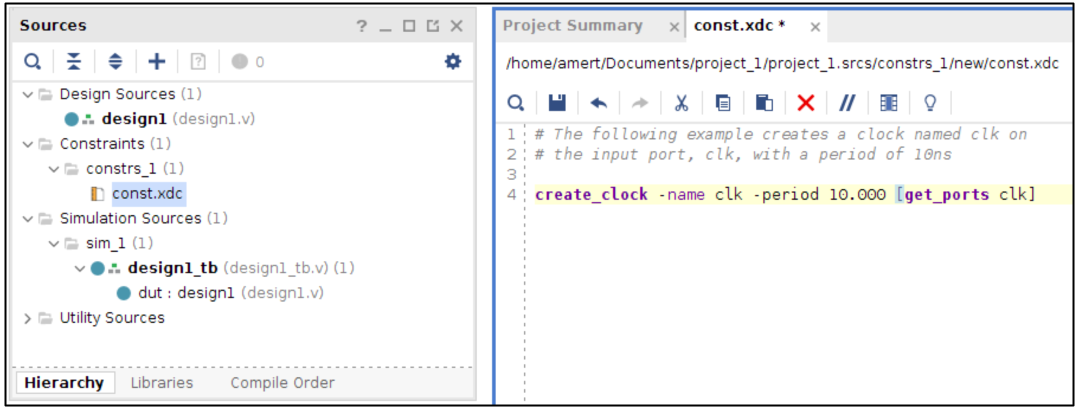
- Constraint file
    - Timing: *For setting clock frequency of your design*
    - Placement: *For floorplanning*
    - I/O: *Assigning your design inputs/outputs to FPGA pins*
    - Other user-defined constraints (i.e., *false paths*)

- For adding the constraint file, click on *Add Sources* in Flow Navigator Window, select *Add or create constraints* and then click on Next.
- Click on *Create File* button, name your simulation file (e.g., const.xdc) and click on *Next*.
- Then, you will see your source file under Design Sources tab on Sources window.

Constraint File

- We will only use timing constraint for clock frequency/period [1]

```
create_clock -name clk -period PERIOD [get_ports PORT_NAME]
```



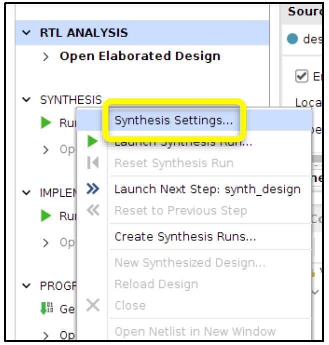
/home/amert/Documents/project_1/project_1.srcs/constrs_1/new/const.xdc

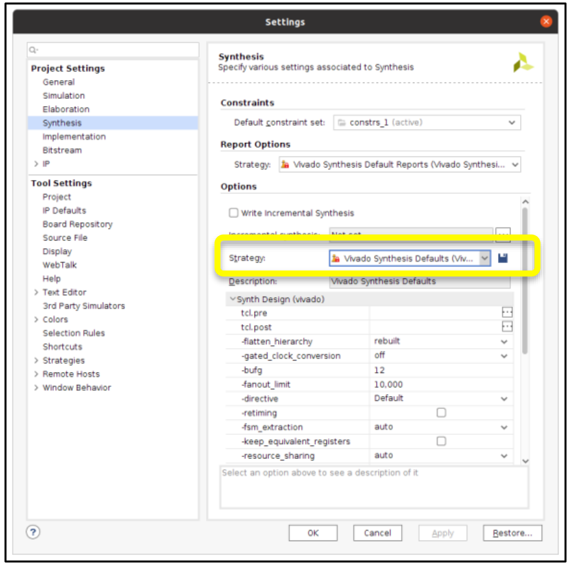
```
# The following example creates a clock named clk on
# the input port, clk, with a period of 10ns

create_clock -name clk -period 10.000 [get_ports clk]
```

[1] https://docs.xilinx.com/r/2021.2-English/ug835-vivado-tcl-commands/create_clock

Synthesis

- Synthesis translates RTL code to a netlist which defines the circuit[1]
    - You can change synthesis strategy (i.e., area optimized or performance optimized) from synthesis settings (right click on *SYNTHESIS -> Synthesis Settings*)
    - For starting synthesis, click on *Run Synthesis*.





Synthesis
Specify various settings associated to Synthesis

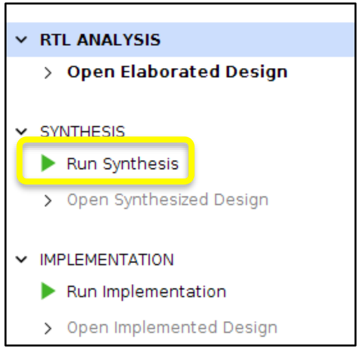
Constraints
Default constraint set: constrs_1 (active)

Report Options
Strategy: Vivado Synthesis Default Reports (Vivado Synthesi...

Options
☐ Write Incremental Synthesis
Strategy: Vivado Synthesis Defaults (Viv...
Description: Vivado Synthesis Defaults

Synth Design (vivado)	
tcl.pre	
tcl.post	
-flatten_hierarchy	rebuilt
-gated_clock_conversion	off
-bufg	12
-fanout_limit	10,000
-directive	Default
-retiming	☐
-fsm_extraction	auto
-keep_equivalent_registers	☐
-resource_sharing	auto

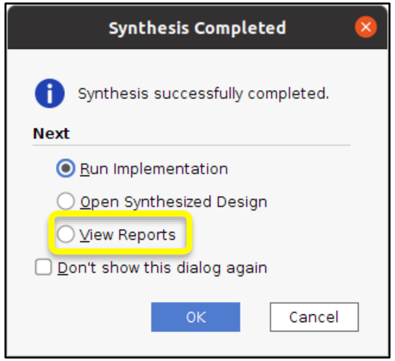
Select an option above to see a description of it



[1] https://docs.xilinx.com/v/u/2021.1-English/ug901-vivado-synthesis

Synthesis

- After synthesis is finished, you can directly start implementation, open the synthesized design or view synthesis reports.
  - You can see synthesis report summary in *Project Summary*.



Synthesis Completed

Synthesis successfully completed.

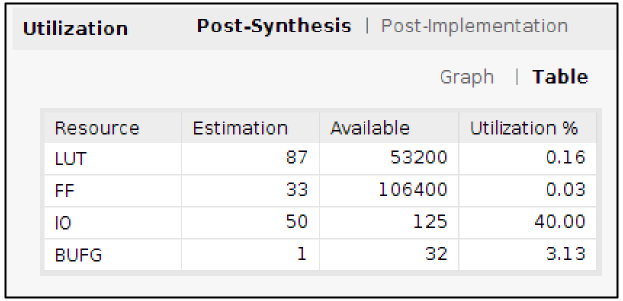
Next

[✓ Run Implementation]

Open Synthesized Design

View Reports

Don't show this dialog again



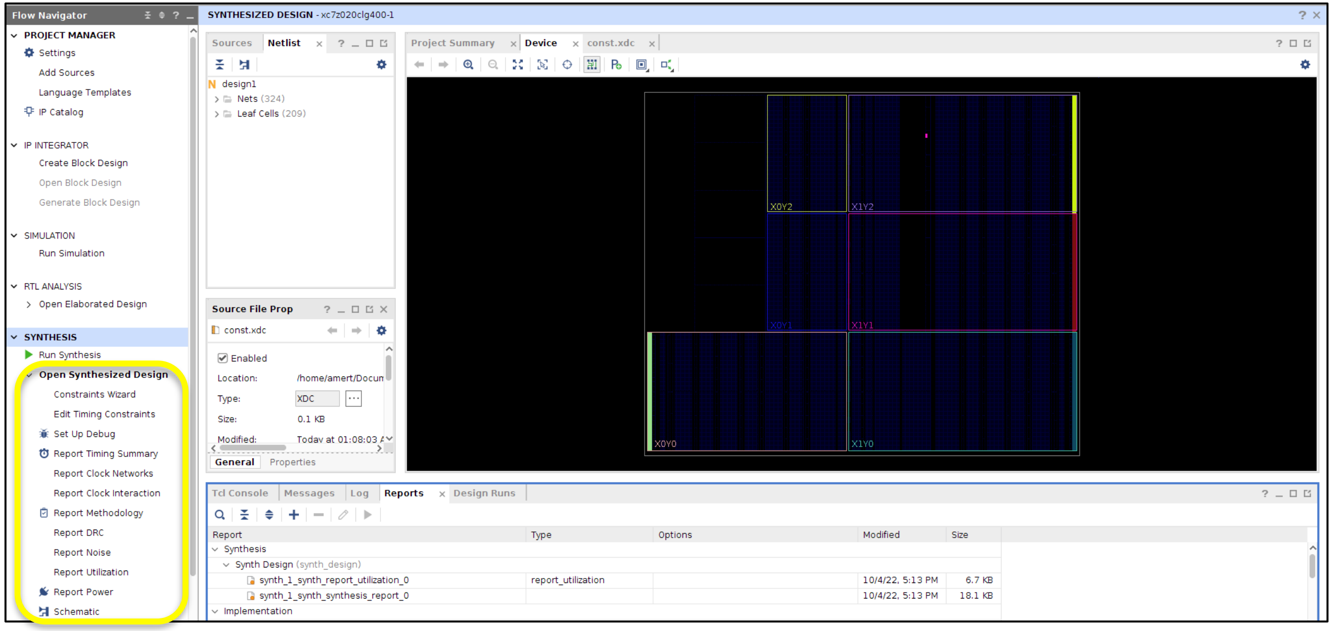
Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	87	53200	0.16
FF	33	106400	0.03
IO	50	125	40.00
BUFG	1	32	3.13

Synthesis

- After synthesis is finished, you can directly start implementation, open the synthesized design or view synthesis reports.
  - You can see synthesis report summary in *Project Summary*.
  - For detailed report/results, open the synthesized design.

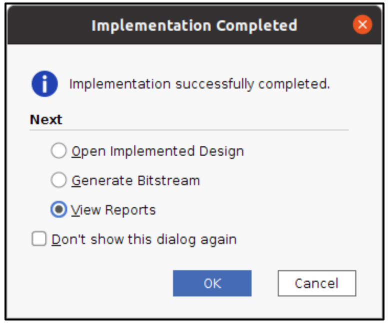


Report	Type	Options	Modified	Size
Synthesis				
Synth Design (synth_design)				
synth_1_synth_report_utilization_0	report_utilization		10/4/22, 5:13 PM	6.7 KB
synth_1_synth_synthesis_report_0			10/4/22, 5:13 PM	18.1 KB
Implementation				

Implementation

- Implementation takes the netlist with user constraint and maps it to actual FPGA components[1].
  - Similar to Synthesis, you can change implementation strategy
  - For starting implementation, click on *Run Implementation*.

- After implementation is finished, you can start bitstream generation, open the implemented design or view implementation report.
  - You can see implementation report (area and timing) in *Project Summary*.

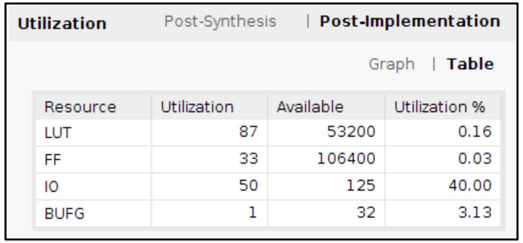


Implementation Completed

Implementation successfully completed.

Next

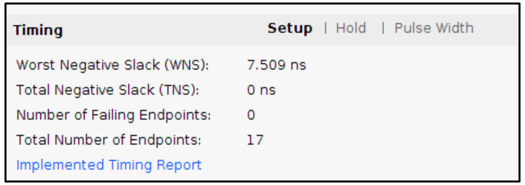
( ) Open Implemented Design
( ) Generate Bitstream
[✓ View Reports]
[ ] Don't show this dialog again



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

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	87	53200	0.16
FF	33	106400	0.03
IO	50	125	40.00
BUFG	1	32	3.13



Timing **Setup** | Hold | Pulse Width

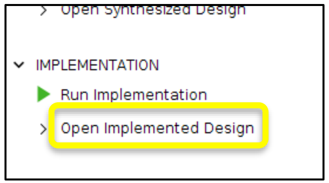
Worst Negative Slack (WNS):	7.509 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	17

Implemented Timing Report

[1] https://docs.xilinx.com/r/2021.1-English/ug904-vivado-implementation/Revision-History

Implementation

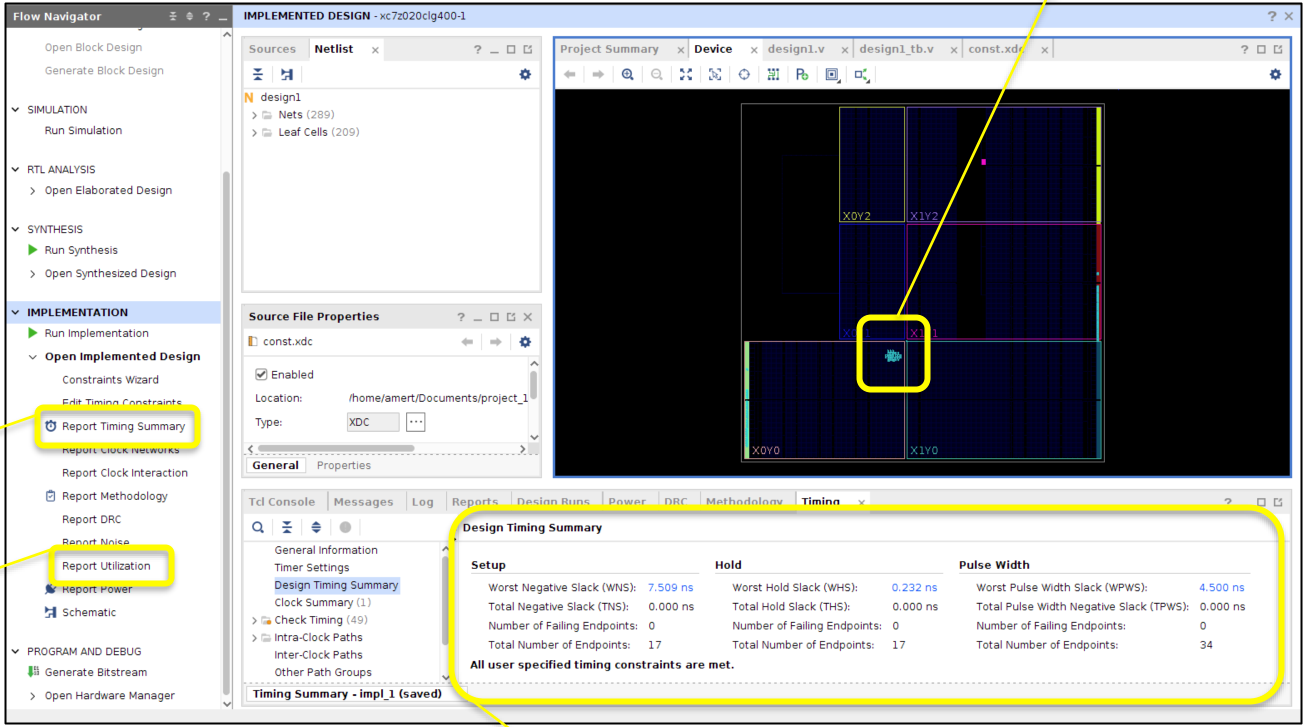
- For detailed report/results and the placed & routed design, open the implemented design.



Placed design

For detailed timing report

For detailed utilization report



Design Timing Summary

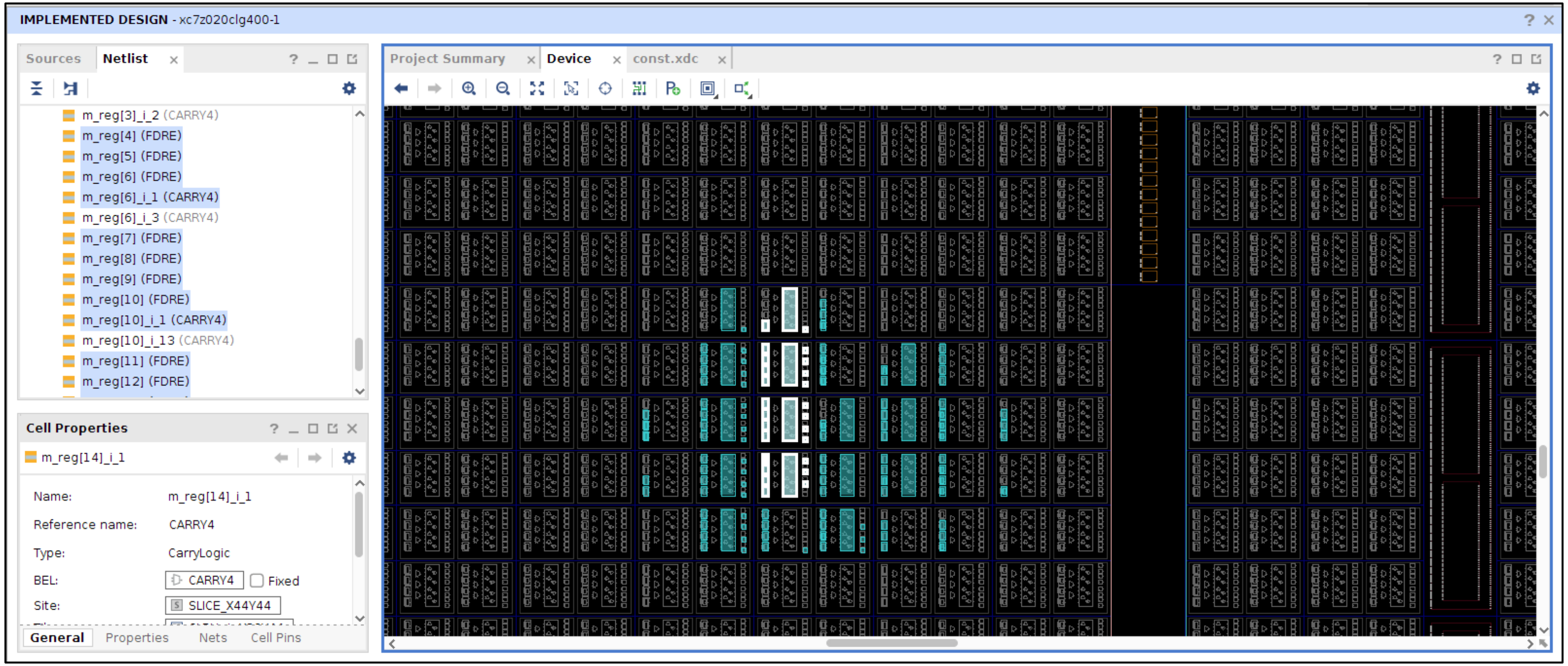
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.509 ns	Worst Hold Slack (WHS):	0.232 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	17	Total Number of Endpoints:	17	Total Number of Endpoints:	34

All user specified timing constraints are met.

Timing summary

Implementation

- For detailed report/results and the placed & routed design, open the implemented design.
  - Placed design


IMPLEMENTED DESIGN - xc7z020clg400-1

Sources | Netlist

m_reg[3]_i_2 (CARRY4)
m_reg[4] (FDRE)
m_reg[5] (FDRE)
m_reg[6] (FDRE)
m_reg[6]_i_1 (CARRY4)
m_reg[6]_i_3 (CARRY4)
m_reg[7] (FDRE)
m_reg[8] (FDRE)
m_reg[9] (FDRE)
m_reg[10] (FDRE)
m_reg[10]_i_1 (CARRY4)
m_reg[10]_i_13 (CARRY4)
m_reg[11] (FDRE)
m_reg[12] (FDRE)

Cell Properties

m_reg[14]_i_1

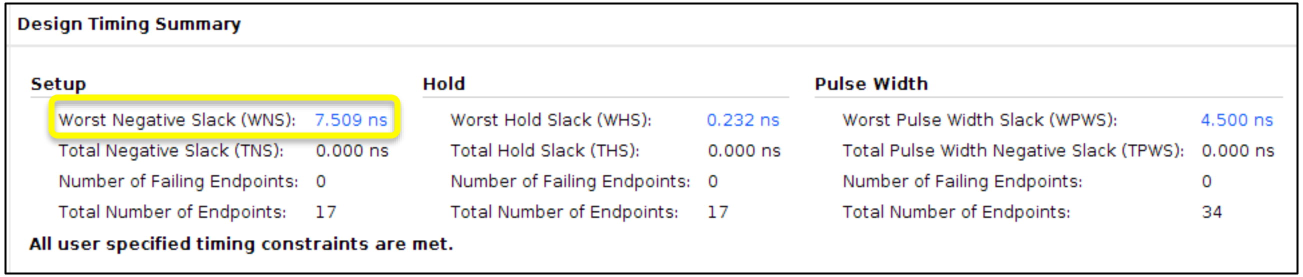
Name:	m_reg[14]_i_1
Reference name:	CARRY4
Type:	CarryLogic
BEL:	CARRY4 ☐ Fixed
Site:	SLICE_X44Y44

General | Properties | Nets | Cell Pins

Project Summary | Device | const.xdc

Implementation

- For detailed report/results and the placed & routed design, open the implemented design.
    - Detailed timing result (showing critical paths in your design).
    - Worst Negative Slack (WNS): If it is negative, then it means your design could not meet the timing requirement (given in your constraint file) and your design has failing paths.
        - Click on WNS value to see critical paths.



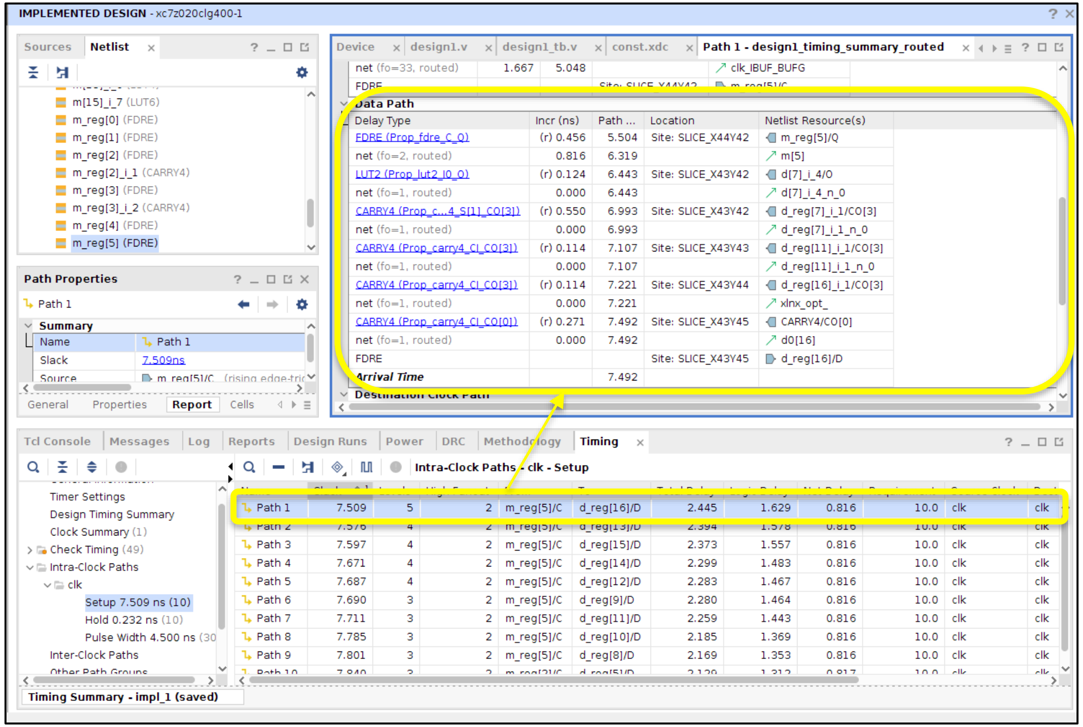
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.509 ns	Worst Hold Slack (WHS):	0.232 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	17	Total Number of Endpoints:	17	Total Number of Endpoints:	34

All user specified timing constraints are met.

Implementation

- For detailed report/results and the placed & routed design, open the implemented design.
    - Detailed timing result (showing critical paths in your design).



IMPLEMENTED DESIGN - xc7z020clg400-1

Sources | Netlist

- m[15]_i_7 (LUT6)
- m_reg[0] (FDRE)
- m_reg[1] (FDRE)
- m_reg[2] (FDRE)
- m_reg[2]_i_1 (CARRY4)
- m_reg[3] (FDRE)
- m_reg[3]_i_2 (CARRY4)
- m_reg[4] (FDRE)
- m_reg[5] (FDRE)

Path Properties

Path 1

Summary

Name	Path 1
Slack	7.509ns
Source	m_reg[5]/C (rising edge-tri

General | Properties | Report | Cells

Device | design1.v | design1_tb.v | const.xdc | Path 1 - design1_timing_summary_routed

net (fo=33, routed)	1.667	5.048		clk_IBUF_BUFG

Data Path

Delay Type	Incr (ns)	Path ...	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(r) 0.456	5.504	Site: SLICE_X44Y42	m_reg[5]/Q
net (fo=2, routed)	0.816	6.319		m[5]
LUT2 (Prop_lut2_I0_O)	(r) 0.124	6.443	Site: SLICE_X43Y42	d[7]_i_4/O
net (fo=1, routed)	0.000	6.443		d[7]_i_4_n_0
CARRY4 (Prop_c...4_S[1]_CO[3])	(r) 0.550	6.993	Site: SLICE_X43Y42	d_reg[7]_i_1/CO[3]
net (fo=1, routed)	0.000	6.993		d_reg[7]_i_1_n_0
CARRY4 (Prop_carry4_CI_CO[3])	(r) 0.114	7.107	Site: SLICE_X43Y43	d_reg[11]_i_1/CO[3]
net (fo=1, routed)	0.000	7.107		d_reg[11]_i_1_n_0
CARRY4 (Prop_carry4_CI_CO[3])	(r) 0.114	7.221	Site: SLICE_X43Y44	d_reg[16]_i_1/CO[3]
net (fo=1, routed)	0.000	7.221		xlnx_opt_
CARRY4 (Prop_carry4_CI_CO[0])	(r) 0.271	7.492	Site: SLICE_X43Y45	CARRY4/CO[0]
net (fo=1, routed)	0.000	7.492		d0[16]
FDRE			Site: SLICE_X43Y45	d_reg[16]/D
Arrival Time		7.492		

Destination Clock Path

Tcl Console | Messages | Log | Reports | Design Runs | Power | DRC | Methodology | Timing

- Timer Settings
- Design Timing Summary
- Clock Summary (1)
- Check Timing (49)
- Intra-Clock Paths
    - clk
        - Setup 7.509 ns (10)
        - Hold 0.232 ns (10)
        - Pulse Width 4.500 ns (30
- Inter-Clock Paths
- Other Path Groups

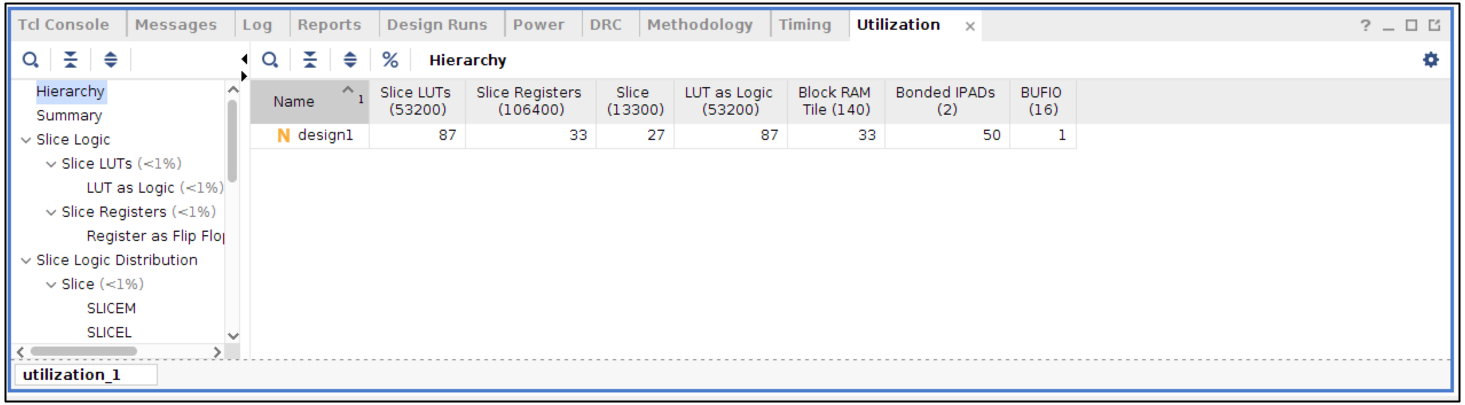
Intra-Clock Paths - clk - Setup

Path 1	7.509	5	2	m_reg[5]/C	d_reg[16]/D	2.445	1.629	0.816	10.0	clk	clk
Path 2	7.576	4	2	m_reg[5]/C	d_reg[13]/D	2.394	1.578	0.816	10.0	clk	clk
Path 3	7.597	4	2	m_reg[5]/C	d_reg[15]/D	2.373	1.557	0.816	10.0	clk	clk
Path 4	7.671	4	2	m_reg[5]/C	d_reg[14]/D	2.299	1.483	0.816	10.0	clk	clk
Path 5	7.687	4	2	m_reg[5]/C	d_reg[12]/D	2.283	1.467	0.816	10.0	clk	clk
Path 6	7.690	3	2	m_reg[5]/C	d_reg[9]/D	2.280	1.464	0.816	10.0	clk	clk
Path 7	7.711	3	2	m_reg[5]/C	d_reg[11]/D	2.259	1.443	0.816	10.0	clk	clk
Path 8	7.785	3	2	m_reg[5]/C	d_reg[10]/D	2.185	1.369	0.816	10.0	clk	clk
Path 9	7.801	3	2	m_reg[5]/C	d_reg[8]/D	2.169	1.353	0.816	10.0	clk	clk
Path 10	7.840	3	2	m_reg[2]/C	d_reg[5]/D	2.129	1.312	0.817	10.0	clk	clk

Timing Summary - impl_1 (saved)

Implementation

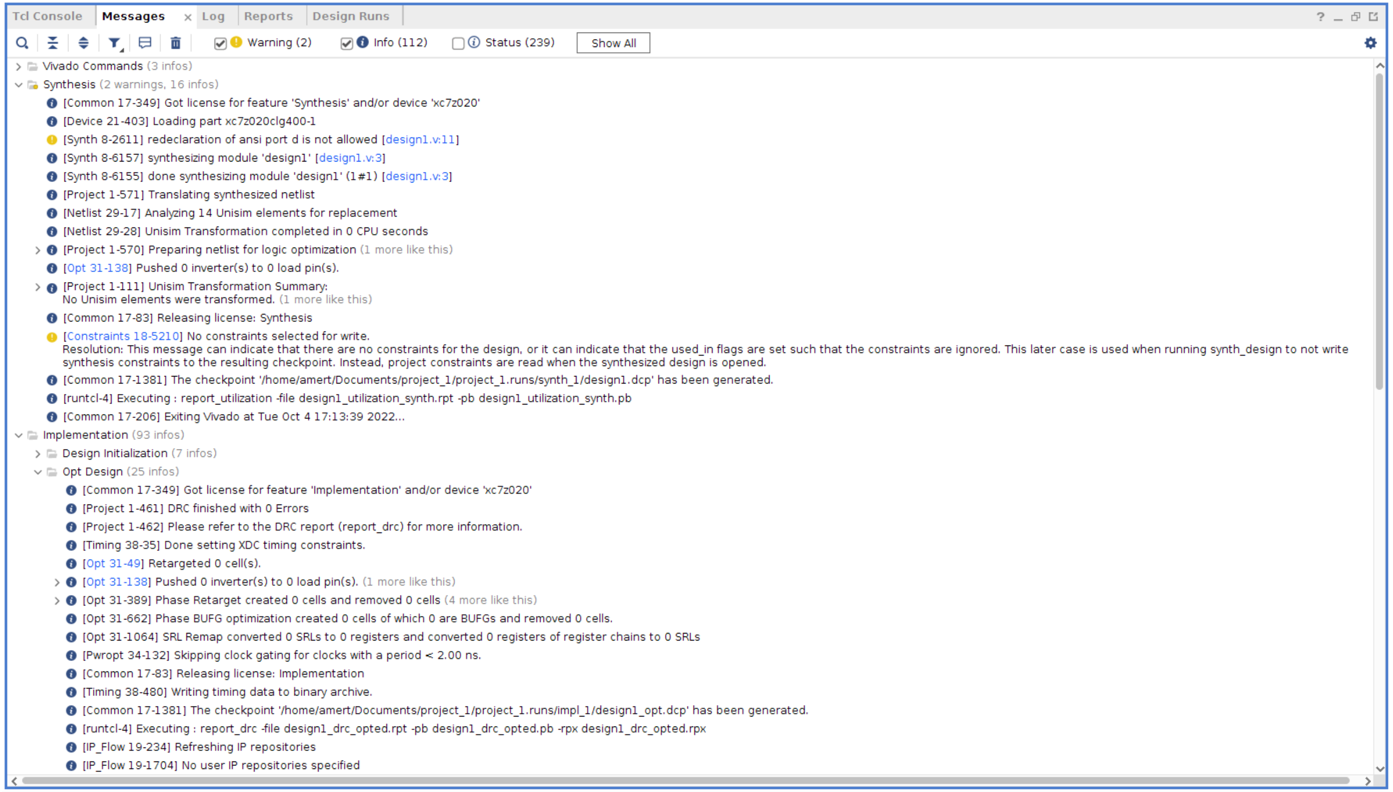
- For detailed report/results and the placed & routed design, open the implemented design.
  - Detailed area (utilization) result.



Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	Block RAM Tile (140)	Bonded IPADs (2)	BUFIO (16)
design1	87	33	27	87	33	50	1

Synthesis & Implementation Messages

- Messages window displays a filtered list of the Vivado log. This list includes useful information such as the main messages, warnings, errors.

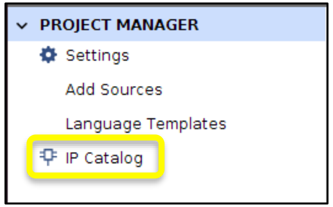


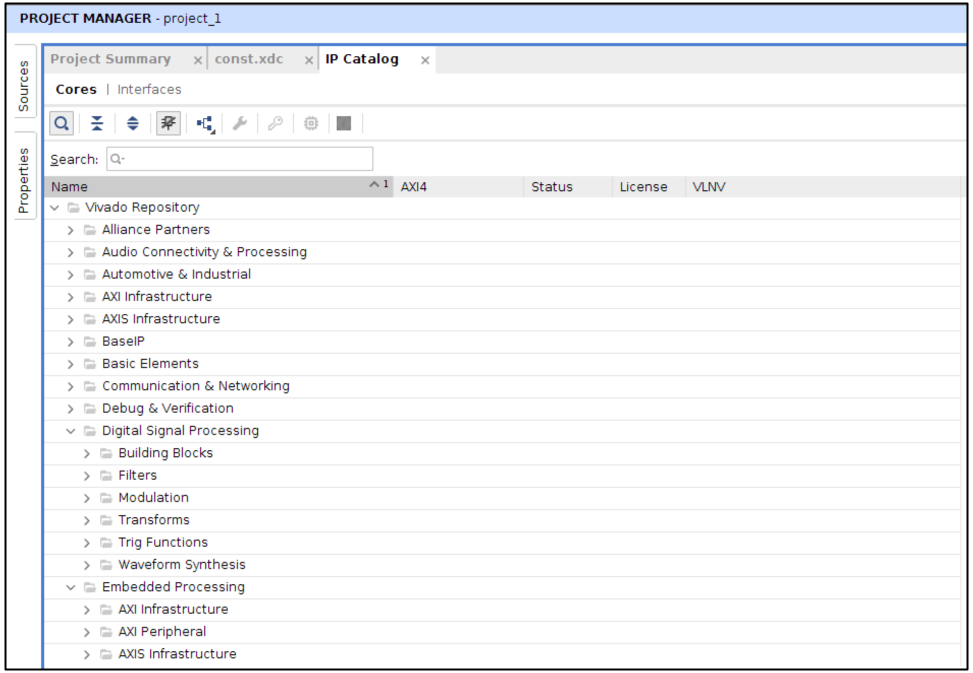
```
> Vivado Commands (3 infos)
v Synthesis (2 warnings, 16 infos)
    [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7z020'
    [Device 21-403] Loading part xc7z020clg400-1
    [Synth 8-2611] redeclaration of ansi port d is not allowed [design1.v:11]
    [Synth 8-6157] synthesizing module 'design1' [design1.v:3]
    [Synth 8-6155] done synthesizing module 'design1' (1#1) [design1.v:3]
    [Project 1-571] Translating synthesized netlist
    [Netlist 29-17] Analyzing 14 Unisim elements for replacement
    [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
  > [Project 1-570] Preparing netlist for logic optimization (1 more like this)
    [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
  > [Project 1-111] Unisim Transformation Summary:
    No Unisim elements were transformed. (1 more like this)
    [Common 17-83] Releasing license: Synthesis
    [Constraints 18-5210] No constraints selected for write.
    Resolution: This message can indicate that there are no constraints for the design, or it can indicate that the used_in flags are set such that the constraints are ignored. This later case is used when running synth_design to not write
    synthesis constraints to the resulting checkpoint. Instead, project constraints are read when the synthesized design is opened.
    [Common 17-1381] The checkpoint '/home/amert/Documents/project_1/project_1.runs/synth_1/design1.dcp' has been generated.
    [runtcl-4] Executing : report_utilization -file design1_utilization_synth.rpt -pb design1_utilization_synth.pb
    [Common 17-206] Exiting Vivado at Tue Oct 4 17:13:39 2022...
v Implementation (93 infos)
  > Design Initialization (7 infos)
  v Opt Design (25 infos)
      [Common 17-349] Got license for feature 'Implementation' and/or device 'xc7z020'
      [Project 1-461] DRC finished with 0 Errors
      [Project 1-462] Please refer to the DRC report (report_drc) for more information.
      [Timing 38-35] Done setting XDC timing constraints.
      [Opt 31-49] Retargeted 0 cell(s).
    > [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s). (1 more like this)
    > [Opt 31-389] Phase Retarget created 0 cells and removed 0 cells (4 more like this)
      [Opt 31-662] Phase BUFG optimization created 0 cells of which 0 are BUFGs and removed 0 cells.
      [Opt 31-1064] SRL Remap converted 0 SRLs to 0 registers and converted 0 registers of register chains to 0 SRLs
      [Pwropt 34-132] Skipping clock gating for clocks with a period < 2.00 ns.
      [Common 17-83] Releasing license: Implementation
      [Timing 38-480] Writing timing data to binary archive.
      [Common 17-1381] The checkpoint '/home/amert/Documents/project_1/project_1.runs/impl_1/design1_opt.dcp' has been generated.
      [runtcl-4] Executing : report_drc -file design1_drc_opted.rpt -pb design1_drc_opted.pb -rpx design1_drc_opted.rpx
      [IP_Flow 19-234] Refreshing IP repositories
      [IP_Flow 19-1704] No user IP repositories specified
```

Adding Xilinx IPs to your Design

IP Catalog

- Click on IP Catalog under Flow Navigator to see the list of available Xilinx IPs.
  - You can use Search bar to find IPs.



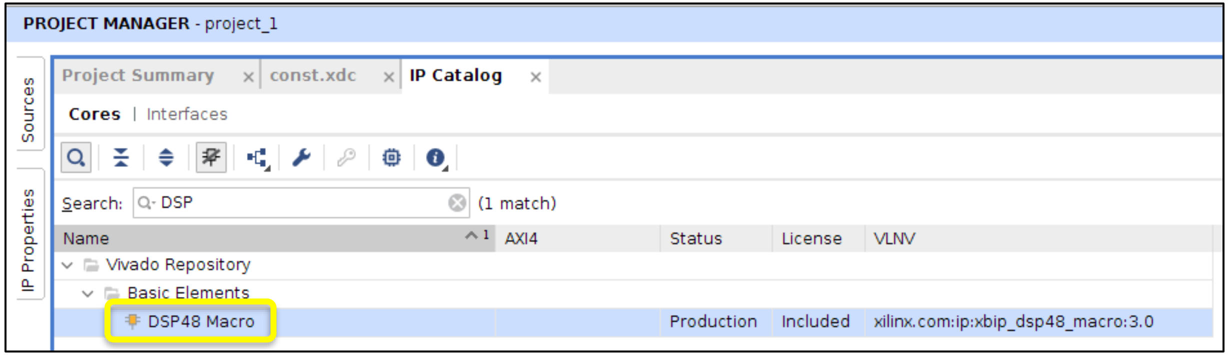


- Vivado Repository
  - Alliance Partners
  - Audio Connectivity & Processing
  - Automotive & Industrial
  - AXI Infrastructure
  - AXIS Infrastructure
  - BaseIP
  - Basic Elements
  - Communication & Networking
  - Debug & Verification
  - Digital Signal Processing
    - Building Blocks
    - Filters
    - Modulation
    - Transforms
    - Trig Functions
    - Waveform Synthesis
  - Embedded Processing
    - AXI Infrastructure
    - AXI Peripheral
    - AXIS Infrastructure

Adding Xilinx IPs to your Design (Example: DSP48 Macro)

IP Catalog Example: DSP48 Macro

- Xilinx FPGAs have dedicated, full-custom, low-power DSP slices.
  - 7 Series DSP48E1 Slice User Guide
    https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1
  - Search for *DSP48 Macro* and double click on it.

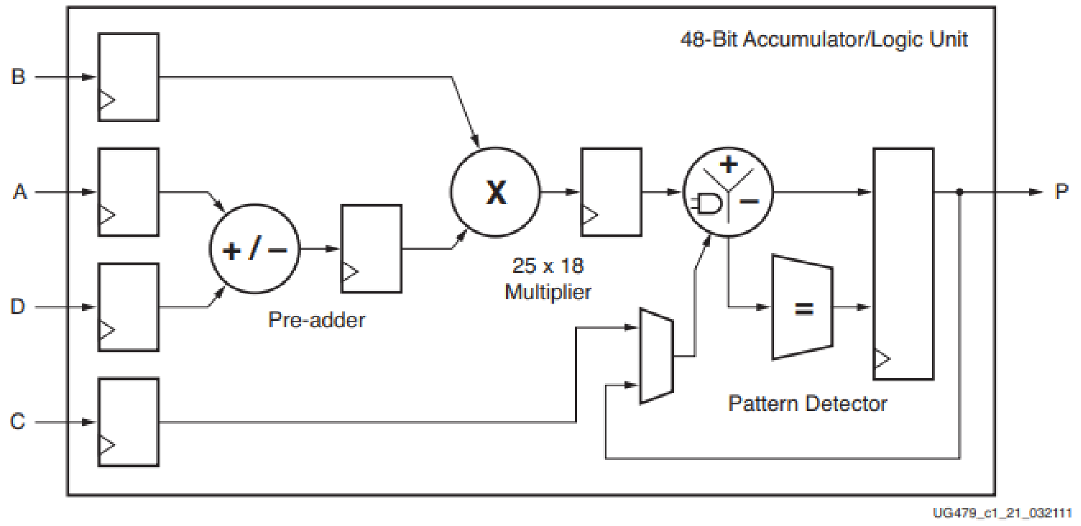


Search: DSP (1 match)

Name	AXI4	Status	License	VLNV
Vivado Repository				
Basic Elements				
DSP48 Macro		Production	Included	xilinx.com:ip:xbip_dsp48_macro:3.0

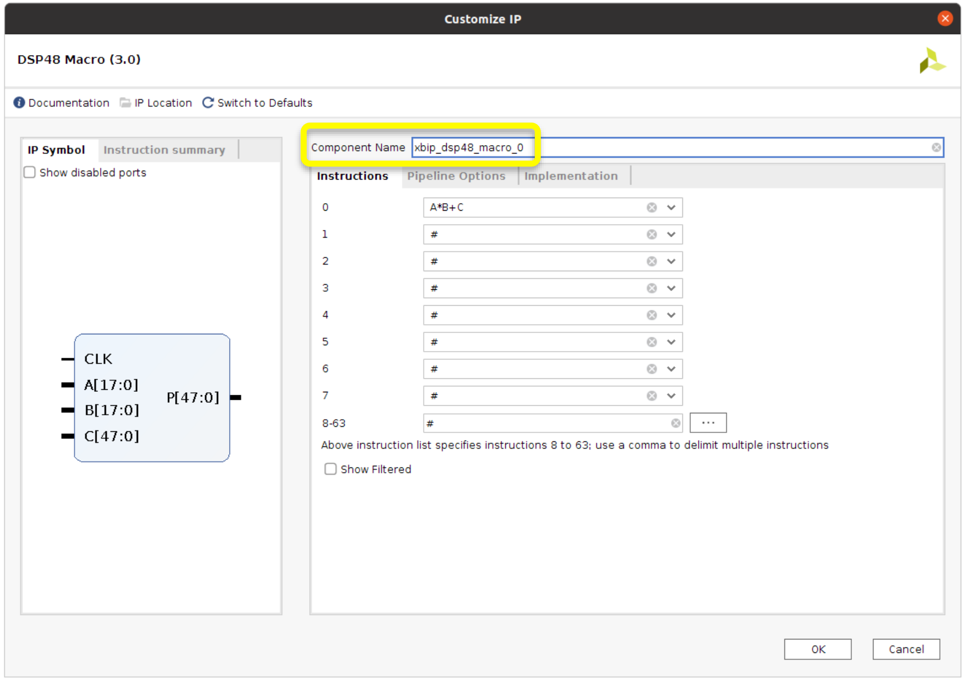
IP Catalog Example: DSP48 Macro

- Xilinx FPGAs have dedicated, full-custom, low-power DSP slices.
  - 7 Series DSP48E1 Slice User Guide
    https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1
  - Search for DSP48 Macro and double click on it.



IP Catalog Example: DSP48 Macro

- Xilinx FPGAs have dedicated, full-custom, low-power DSP slices.
    - First, set the component name.



Customize IP

DSP48 Macro (3.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Instruction summary

Show disabled ports

Component Name: xbip_dsp48_macro_0

Instructions | Pipeline Options | Implementation

0	A*B+C
1	#
2	#
3	#
4	#
5	#
6	#
7	#
8-63	#

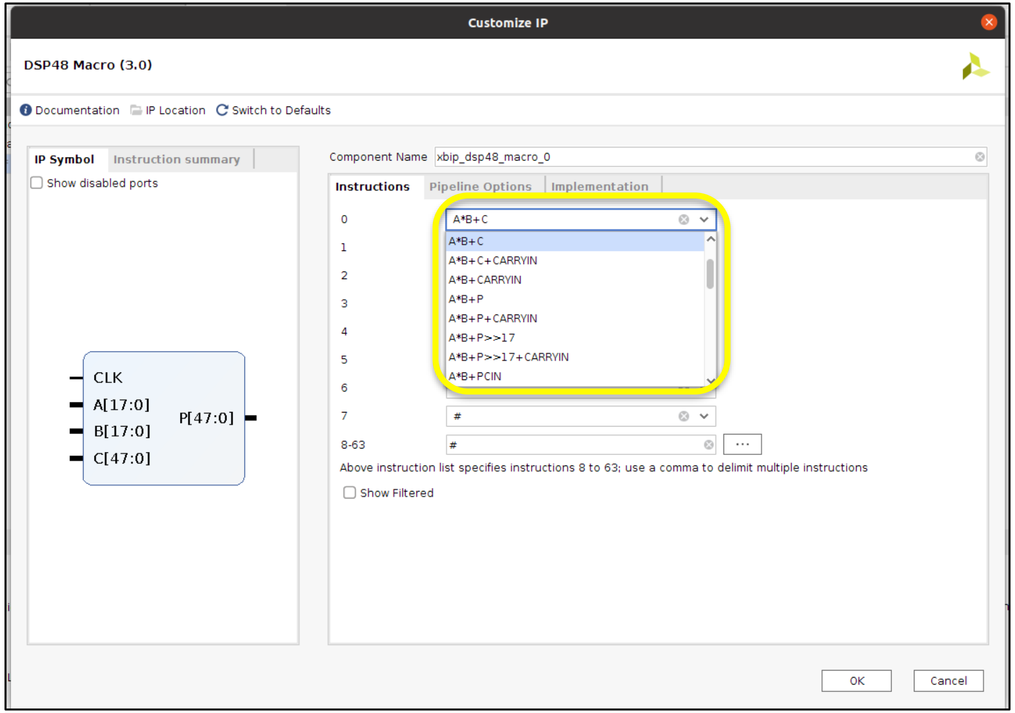
Above instruction list specifies instructions 8 to 63; use a comma to delimit multiple instructions

Show Filtered

OK | Cancel

IP Catalog Example: DSP48 Macro

- DSP unit provides various functionalities.



Customize IP

DSP48 Macro (3.0)

Documentation IP Location Switch to Defaults

IP Symbol | Instruction summary

Show disabled ports

Component Name xbip_dsp48_macro_0

Instructions | Pipeline Options | Implementation

0 A*B+C

A*B+C
A*B+C+CARRYIN
A*B+CARRYIN
A*B+P
A*B+P+CARRYIN
A*B+P>>17
A*B+P>>17+CARRYIN
A*B+PCIN

1
2
3
4
5
6
7 #
8-63 #

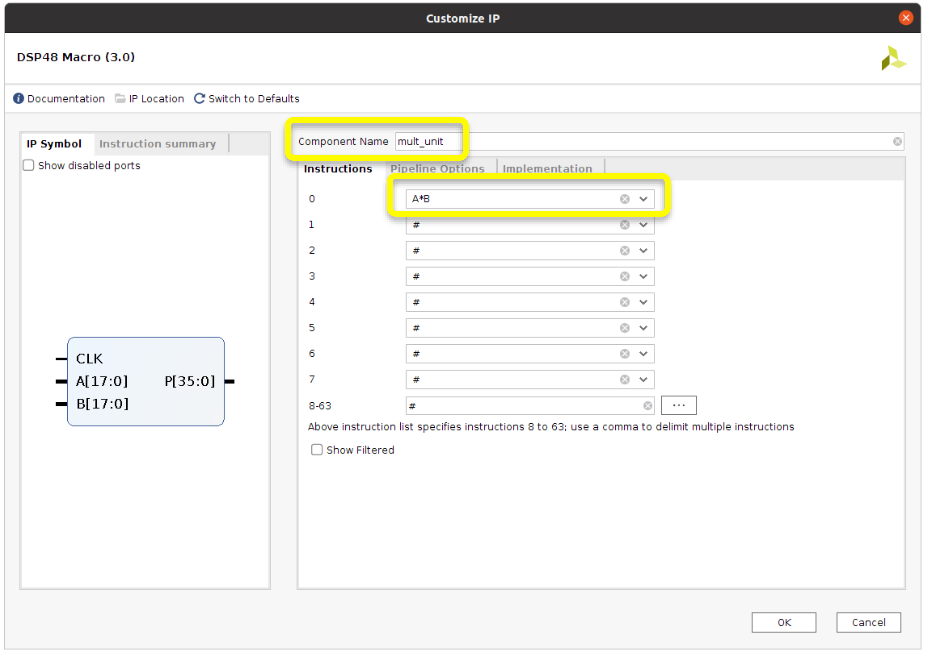
Above instruction list specifies instructions 8 to 63; use a comma to delimit multiple instructions

Show Filtered

OK Cancel

IP Catalog Example: DSP48 Macro

- As an example, we select A*B as functionality and set the component name as *mult_unit*.
  - You will use the component name for instantiating the DSP unit in your design



Customize IP

DSP48 Macro (3.0)

Documentation IP Location Switch to Defaults

IP Symbol | Instruction summary

Show disabled ports

Component Name: mult_unit

Instructions | Pipeline Options | Implementation

0	A*B
1	#
2	#
3	#
4	#
5	#
6	#
7	#
8-63	#

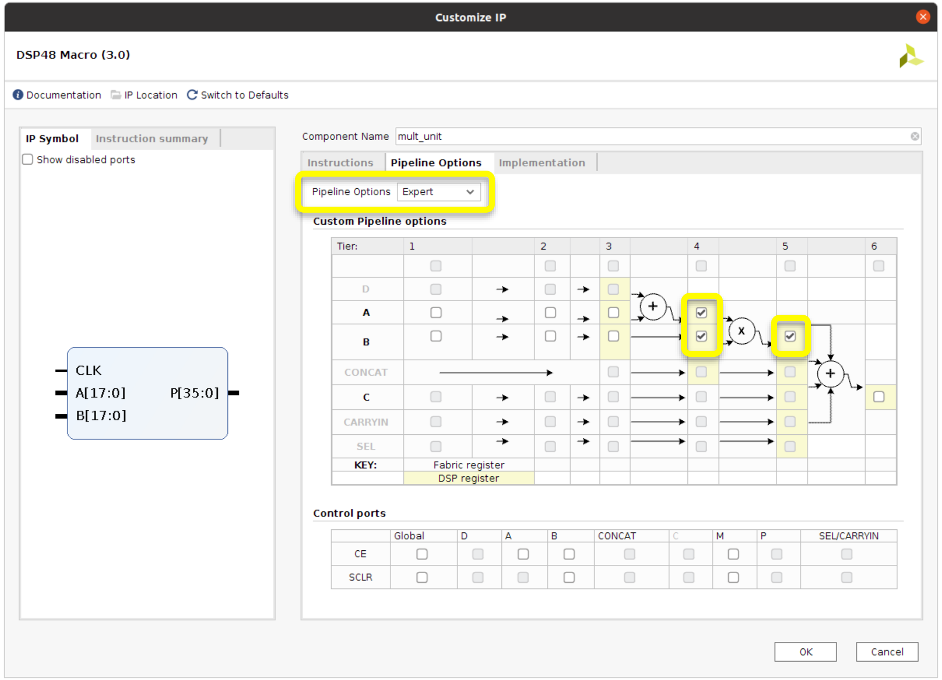
Above instruction list specifies instructions 8 to 63; use a comma to delimit multiple instructions

Show Filtered

OK Cancel

IP Catalog Example: DSP48 Macro

- You can set the pipeline options.
  - As an example, we use two level of pipeline registers



Customize IP

DSP48 Macro (3.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Instruction summary

☐ Show disabled ports

Component Name: mult_unit

Instructions | Pipeline Options | Implementation

Pipeline Options: Expert

Custom Pipeline options

Tier:	1	2	3	4	5	6
	☐	☐	☐	☐	☐	☐
D	☐	☐	☐			
A	☐	☐	☐	✓		
B	☐	☐	☐	✓	✓	
CONCAT			☐	☐	☐	
C	☐	☐	☐	☐	☐	☐
CARRYIN	☐	☐	☐	☐	☐	
SEL	☐	☐	☐	☐	☐	
KEY:	Fabric register					
	DSP register					

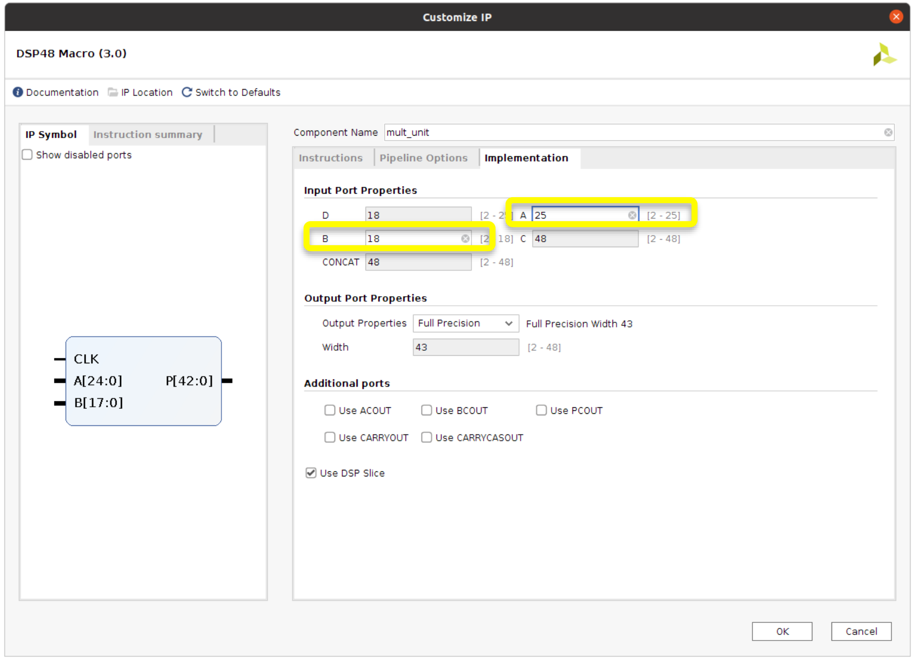
Control ports

	Global	D	A	B	CONCAT	C	M	P	SEL/CARRYIN
CE	☐	☐	☐	☐	☐	☐	☐	☐	☐
SCLR	☐	☐	☐	☐	☐	☐	☐	☐	☐

OK | Cancel

IP Catalog Example: DSP48 Macro

- You can set input/output port widths.
  - We set A and B width as 25 and 18 (an signed 25-bit x 18-bit multiplier)
  - Finally, click on *OK* and then *Generate* to create the DSP.



Customize IP

DSP48 Macro (3.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Instruction summary

Show disabled ports

Component Name: mult_unit

Instructions | Pipeline Options | Implementation

Input Port Properties

D	18	[2 - 2]	A	25	[2 - 25]
B	18	[2 - 18]	C	48	[2 - 48]
CONCAT	48	[2 - 48]			

Output Port Properties

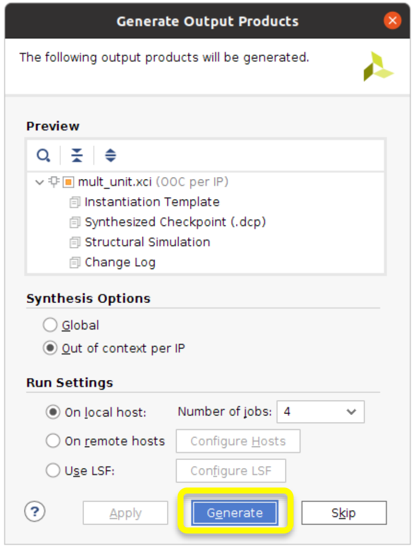
Output Properties: Full Precision — Full Precision Width 43

Width: 43 [2 - 48]

Additional ports

- [ ] Use ACOUT
- [ ] Use BCOUT
- [ ] Use PCOUT
- [ ] Use CARRYOUT
- [ ] Use CARRYCASOUT
- [x] Use DSP Slice

OK | Cancel



Generate Output Products

The following output products will be generated.

Preview

- mult_unit.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Structural Simulation
  - Change Log

Synthesis Options

- ○ Global
- ● Out of context per IP

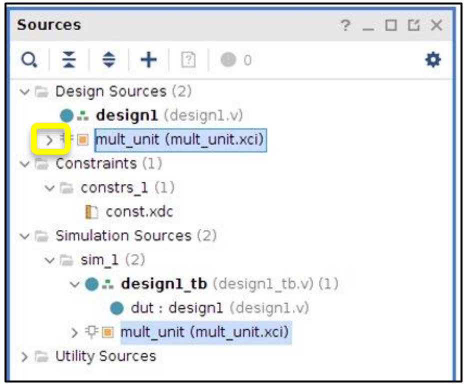
Run Settings

- ● On local host: Number of jobs: 4
- ○ On remote hosts — Configure Hosts
- ○ Use LSF: — Configure LSF

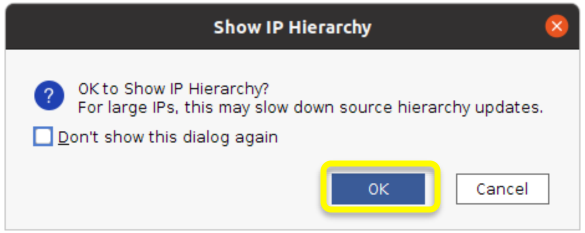
Apply | Generate | Skip

IP Catalog Example: DSP48 Macro

- You will see the DSP IP under Design Sources
  - In order to see the IP ports, expand the IP hierarchy and click on OK.



```
Design Sources (2)
  design1 (design1.v)
  mult_unit (mult_unit.xci)
Constraints (1)
  constrs_1 (1)
    const.xdc
Simulation Sources (2)
  sim_1 (2)
    design1_tb (design1_tb.v) (1)
      dut : design1 (design1.v)
      mult_unit (mult_unit.xci)
Utility Sources
```

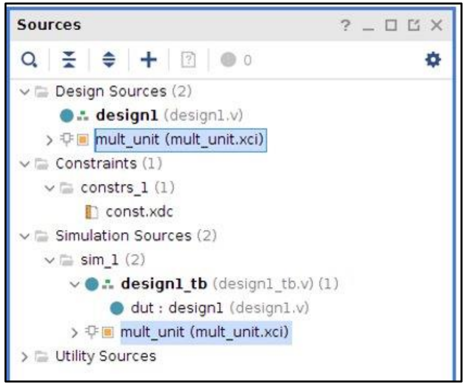

Show IP Hierarchy

OK to Show IP Hierarchy?
For large IPs, this may slow down source hierarchy updates.

☐ Don't show this dialog again

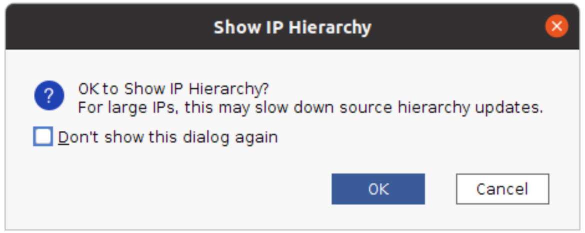
IP Catalog Example: DSP48 Macro

- You will see the DSP IP under Design Sources
  - In order to see the IP ports, expand the IP hierarchy and click on OK.
  - Double click on *mult_unit.vhd* to see the module ports.



Sources

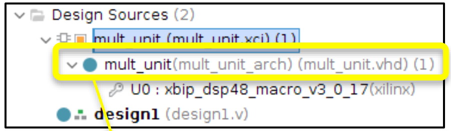
- Design Sources (2)
  - **design1** (design1.v)
  - mult_unit (mult_unit.xci)
- Constraints (1)
  - constrs_1 (1)
    - const.xdc
- Simulation Sources (2)
  - sim_1 (2)
    - **design1_tb** (design1_tb.v) (1)
      - dut : design1 (design1.v)
    - mult_unit (mult_unit.xci)
- Utility Sources



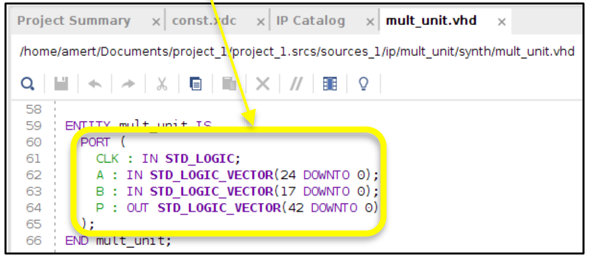
Show IP Hierarchy

OK to Show IP Hierarchy?
For large IPs, this may slow down source hierarchy updates.

☐ Don't show this dialog again



- Design Sources (2)
  - mult_unit (mult_unit.xci) (1)
    - mult_unit(mult_unit_arch) (mult_unit.vhd) (1)
      - U0 : xbip_dsp48_macro_v3_0_17(xilinx)
  - **design1** (design1.v)



/home/amert/Documents/project_1/project_1.srcs/sources_1/ip/mult_unit/synth/mult_unit.vhd

```
ENTITY mult_unit IS
  PORT (
    CLK : IN STD_LOGIC;
    A : IN STD_LOGIC_VECTOR(24 DOWNTO 0);
    B : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
    P : OUT STD_LOGIC_VECTOR(42 DOWNTO 0)
  );
END mult_unit;
```

IP Catalog Example: DSP48 Macro

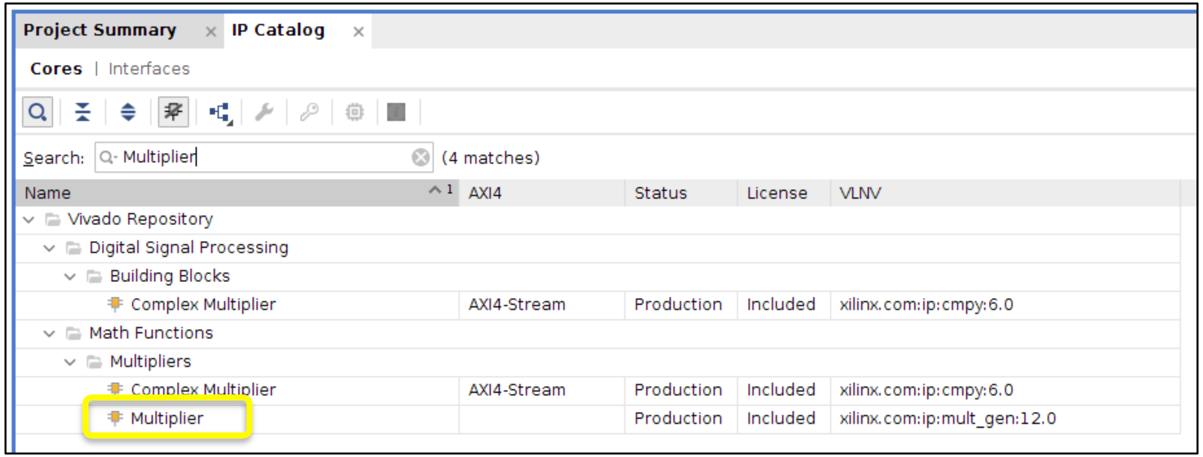
- You will see the DSP IP under Design Sources
  - In order to see the IP ports, expand the IP hierarchy and click on OK.
  - Double click on *mult_unit.vhd* to see the module ports.
    - You can instantiate the module (mult_unit) with module ports.

```
…
wire [24:0] mult_unit_A;
wire [17:0] mult_unit_B;
wire [42:0] mult_unit_P;
…
mult_unit MU(clk,
             mult_unit_A,
             mult_unit_B,
             mult_unit_P);
…
```

Adding Xilinx IPs to your Design (Example: Multiplier)

IP Catalog Example: Multiplier

- Xilinx FPGAs provide a multiplier core. 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.
  - 7 Series DSP48E1 Slice User Guide https://docs.xilinx.com/v/u/en-US/pg108-mult-gen
  - Search for *Multiplier* and double click on it.

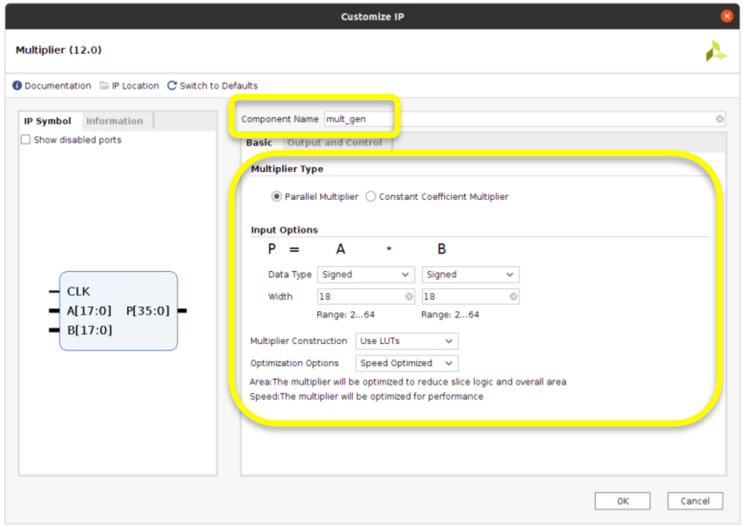


Search: Multiplier (4 matches)

Name	AXI4	Status	License	VLNV
Vivado Repository				
Digital Signal Processing				
Building Blocks				
Complex Multiplier	AXI4-Stream	Production	Included	xilinx.com:ip:cmpy:6.0
Math Functions				
Multipliers				
Complex Multiplier	AXI4-Stream	Production	Included	xilinx.com:ip:cmpy:6.0
Multiplier		Production	Included	xilinx.com:ip:mult_gen:12.0

IP Catalog Example: Multiplier

- Xilinx FPGAs provide a multiplier core.
  - Set the component name and select one of multiplier types
    - Parallel Multiplier or Constant Coefficient Multiplier
  - Set input options
    - Data type: Signed or Unsigned
    - Bit width: 2 to 64
    - Multiplier construction: Use LUT or Use Mults (DSPs)
    - Optimization options: Speed optimized or Area optimized



Component Name: mult_gen

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

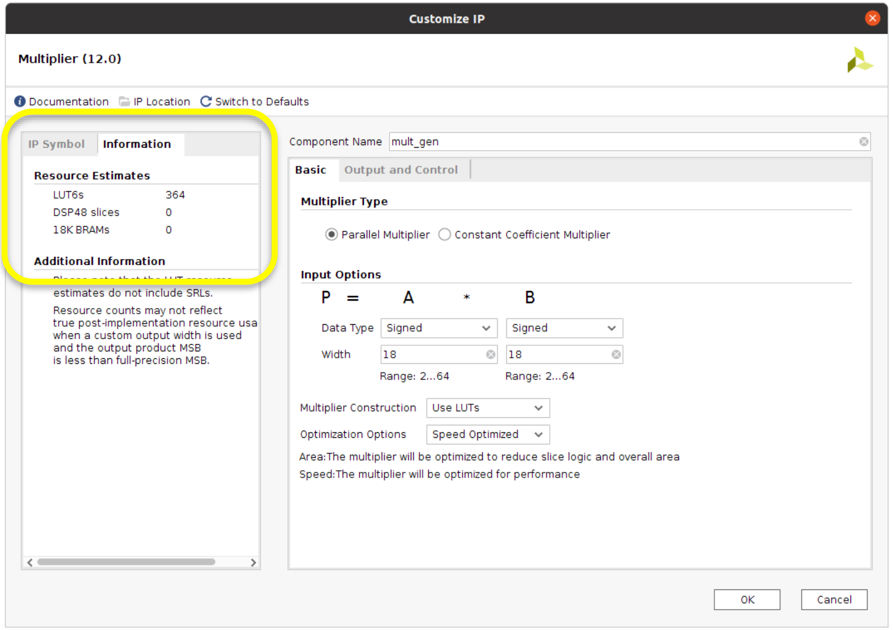
Optimization Options: Speed Optimized

Area:The multiplier will be optimized to reduce slice logic and overall area

Speed:The multiplier will be optimized for performance

IP Catalog Example: Multiplier

- Xilinx FPGAs provide a multiplier core.
  - Information tab shows resource estimate (LUT6s, DSP48 slices, 18K BRAMs) based on the Multiplier type and Input options.



Customize IP

Multiplier (12.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | **Information**

Resource Estimates

LUT6s	364
DSP48 slices	0
18K BRAMs	0

Additional Information

estimates do not include SRLs.
Resource counts may not reflect true post-implementation resource usa when a custom output width is used and the output product MSB is less than full-precision MSB.

Component Name: mult_gen

Basic | Output and Control

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

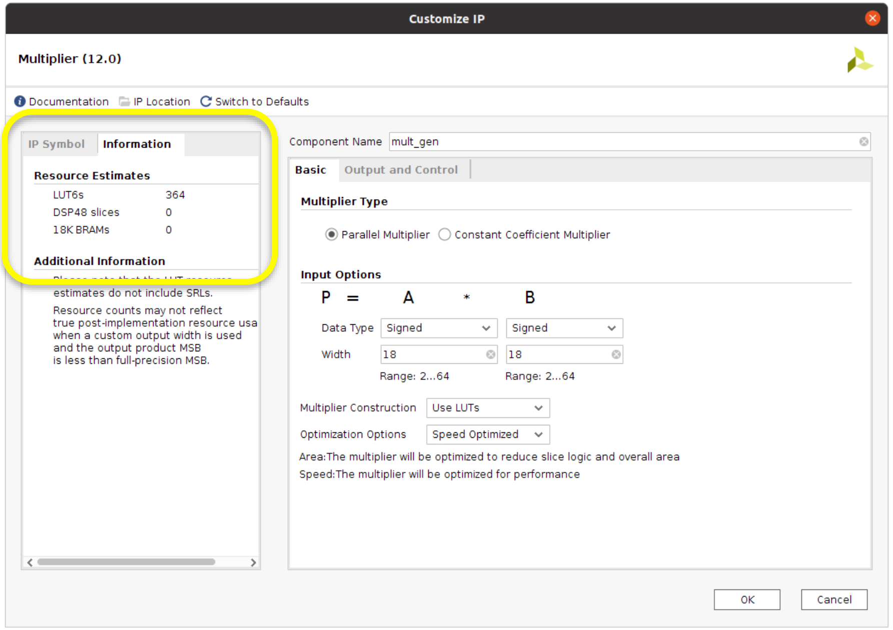
Optimization Options: Speed Optimized

Area:The multiplier will be optimized to reduce slice logic and overall area
Speed:The multiplier will be optimized for performance

OK | Cancel

IP Catalog Example: Multiplier

- Xilinx FPGAs provide a multiplier core.
  - Information tab shows resource estimate (LUT6s, DSP48 slices, 18K BRAMs) based on the Multiplier type and Input options.



Resource Estimates

LUT6s	364
DSP48 slices	0
18K BRAMs	0

Additional Information

Please note that the LUT resource
estimates do not include SRLs.

Resource counts may not reflect
true post-implementation resource usa
when a custom output width is used
and the output product MSB
is less than full-precision MSB.

Component Name: mult_gen

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

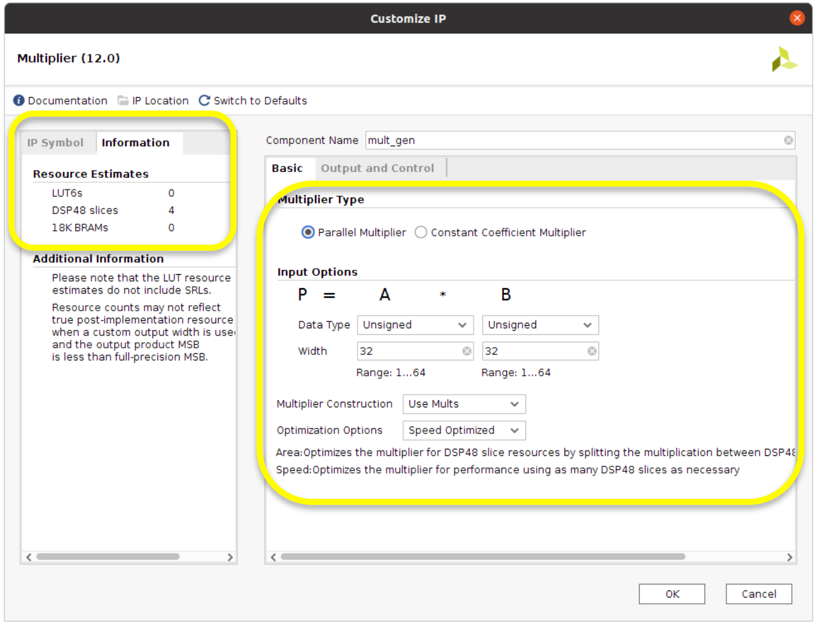
Optimization Options: Speed Optimized

Area:The multiplier will be optimized to reduce slice logic and overall area

Speed:The multiplier will be optimized for performance

IP Catalog Example: Multiplier

- Xilinx FPGAs provide a multiplier core.
  - *Parallel Multiplier* example
    - In *Output and Control* tab, you can adjust the number of pipeline stages (optimum pipeline stages are proposed by the core based on multiplier type and input options)
    - Finally click on OK and generate to create the IP block.



Resource Estimates

LUT6s	0
DSP48 slices	4
18K BRAMs	0

Additional Information

Please note that the LUT resource estimates do not include SRLs.

Resource counts may not reflect true post-implementation resource when a custom output width is used and the output product MSB is less than full-precision MSB.

Component Name: mult_gen

Multiplier Type

[✓ Parallel Multiplier] Constant Coefficient Multiplier

Input Options

$P = A * B$

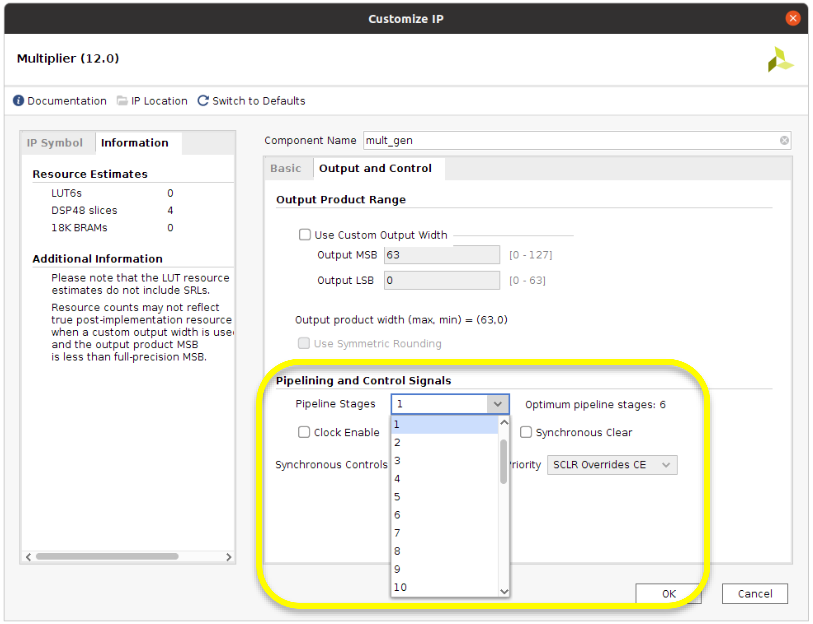
	A	B
Data Type	Unsigned	Unsigned
Width	32	32
	Range: 1...64	Range: 1...64

Multiplier Construction: Use Mults

Optimization Options: Speed Optimized

Area:Optimizes the multiplier for DSP48 slice resources by splitting the multiplication between DSP48

Speed:Optimizes the multiplier for performance using as many DSP48 slices as necessary



Resource Estimates

LUT6s	0
DSP48 slices	4
18K BRAMs	0

Additional Information

Please note that the LUT resource estimates do not include SRLs.

Resource counts may not reflect true post-implementation resource when a custom output width is used and the output product MSB is less than full-precision MSB.

Component Name: mult_gen

Output Product Range

☐ Use Custom Output Width

Output MSB: 63 [0 - 127]

Output LSB: 0 [0 - 63]

Output product width (max, min) = (63,0)

☐ Use Symmetric Rounding

Pipelining and Control Signals

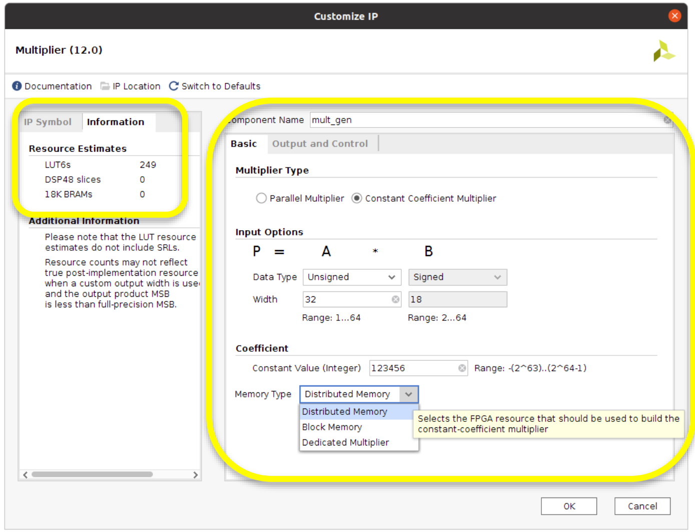
Pipeline Stages: 1 Optimum pipeline stages: 6

☐ Clock Enable ☐ Synchronous Clear

Synchronous Controls ...riority: SCLR Overrides CE

IP Catalog Example: Multiplier

- Xilinx FPGAs provide a multiplier core.
  - *Constant Coefficient Multiplier* example
    - In *Basic* tab you can set the constant integer (second operand) and memory type to implement constant multiplier
    - In *Output and Control* tab, you can adjust the number of pipeline stages (optimum pipeline stages are proposed by the core based on multiplier type and input options)
    - Finally click on OK and generate to create the IP block.



Customize IP

Multiplier (12.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Information

Resource Estimates

LUT6s	249
DSP48 slices	0
18K BRAMs	0

Additional Information

Please note that the LUT resource estimates do not include SRLs.

Resource counts may not reflect true post-implementation resource when a custom output width is use and the output product MSB is less than full-precision MSB.

omponent Name: mult_gen

Basic | Output and Control

Multiplier Type

Parallel Multiplier [✓ Constant Coefficient Multiplier]

Input Options

P = A * B

	A	B
Data Type	Unsigned	Signed
Width	32	18
	Range: 1...64	Range: 2...64

Coefficient

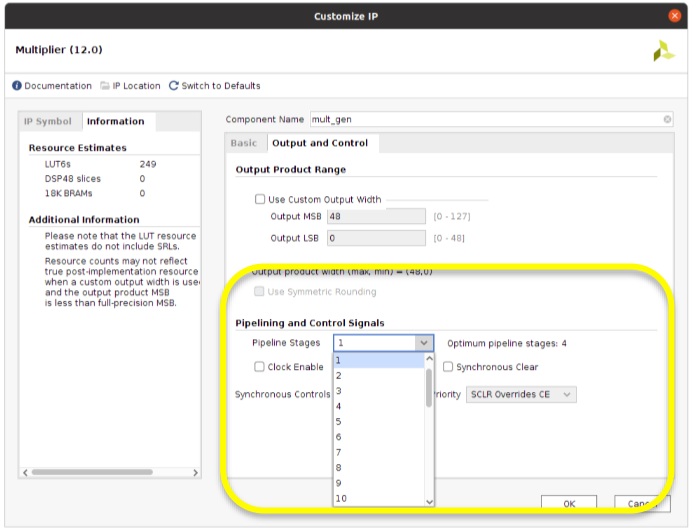
Constant Value (Integer): 123456 Range: -(2^63)..(2^64-1)

Memory Type: Distributed Memory

- Distributed Memory
- Block Memory
- Dedicated Multiplier

Selects the FPGA resource that should be used to build the constant-coefficient multiplier

OK | Cancel



Customize IP

Multiplier (12.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Information

Resource Estimates

LUT6s	249
DSP48 slices	0
18K BRAMs	0

Additional Information

Please note that the LUT resource estimates do not include SRLs.

Resource counts may not reflect true post-implementation resource when a custom output width is use and the output product MSB is less than full-precision MSB.

Component Name: mult_gen

Basic | Output and Control

Output Product Range

☐ Use Custom Output Width

Output MSB: 48 [0 - 127]

Output LSB: 0 [0 - 48]

Output product width (max, min) = (48,0)

☐ Use Symmetric Rounding

Pipelining and Control Signals

Pipeline Stages: 1 Optimum pipeline stages: 4

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

☐ Clock Enable ☐ Synchronous Clear

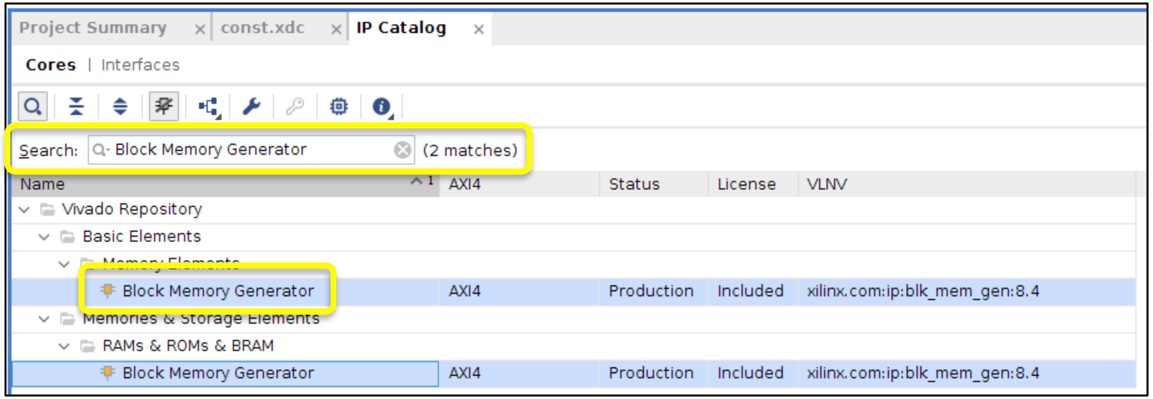
Synchronous Controls riority: SCLR Overrides CE

OK | Canc

Adding Xilinx IPs to your Design
(Example: Block Memory Generator)

IP Catalog Example: Block Memory Generator

- Xilinx FPGAs have dedicated block memory primitives
  - Block Memory Generator Product Guide
    - https://docs.xilinx.com/v/u/en-US/pg058-blk-mem-gen
  - Search for *Block Memory Generator* and double click on it.



Search: Block Memory Generator (2 matches)

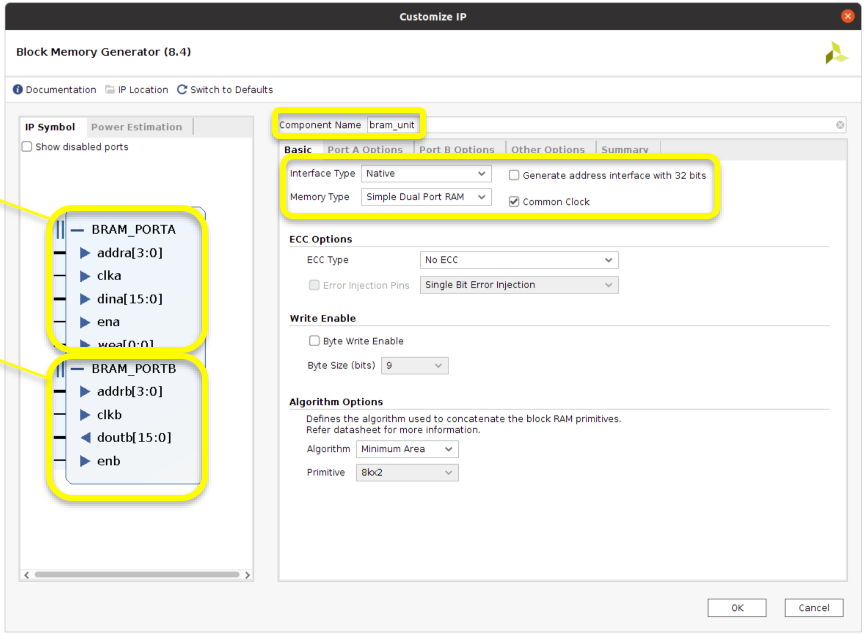
Name	AXI4	Status	License	VLNV
Vivado Repository				
Basic Elements				
Memory Elements				
Block Memory Generator	AXI4	Production	Included	xilinx.com:ip:blk_mem_gen:8.4
Memories & Storage Elements				
RAMs & ROMs & BRAM				
Block Memory Generator	AXI4	Production	Included	xilinx.com:ip:blk_mem_gen:8.4

IP Catalog Example: Block Memory Generator

- Xilinx FPGAs have dedicated block memory primitives
  - Set the component name and select one of available memory types
  - As an example, we select *Simple Dual Port RAM* and name the component as *bram_unit*

Port A is used for WRITE

Port B is used for READ



Block Memory Generator (8.4)

Component Name: bram_unit

Interface Type: Native

Memory Type: Simple Dual Port RAM

☐ Generate address interface with 32 bits

☑ Common Clock

ECC Options

ECC Type: No ECC

☐ Error Injection Pins: Single Bit Error Injection

Write Enable

☐ Byte Write Enable

Byte Size (bits): 9

Algorithm Options

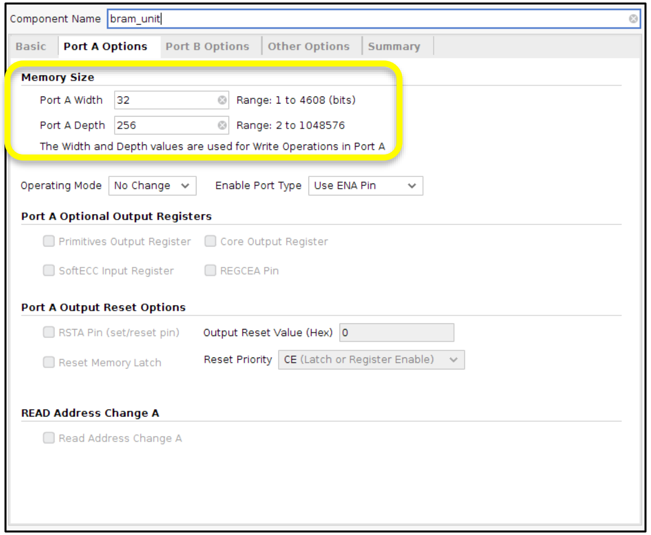
Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: 8kx2

IP Catalog Example: Block Memory Generator

- You can set the data width and depth of Port A and Port B. We set width and depth as 32 and 256 respectively. This memory unit can store 256 of 32-bit words.
  - You can enable optional output register of Port B
    (If you enable it, read latency will be 2 cycles instead of 1)



Component Name: bram_unit

Basic | **Port A Options** | Port B Options | Other Options | Summary

Memory Size

Port A Width	32	Range: 1 to 4608 (bits)
Port A Depth	256	Range: 2 to 1048576

The Width and Depth values are used for Write Operations in Port A

Operating Mode: No Change   Enable Port Type: Use ENA Pin

Port A Optional Output Registers

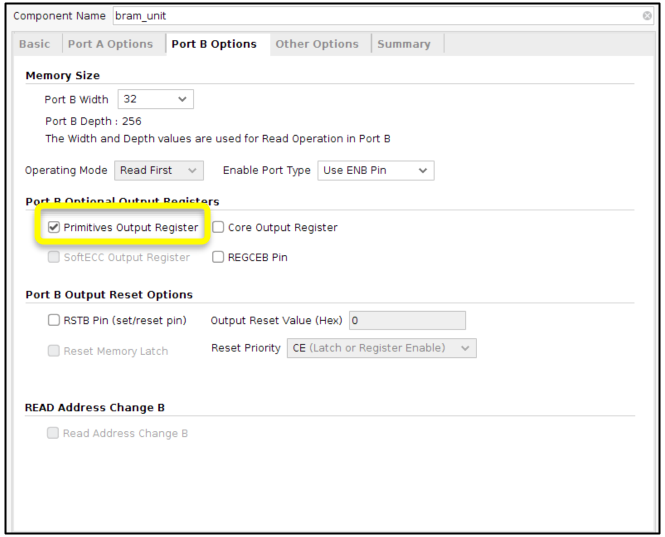
☐ Primitives Output Register   ☐ Core Output Register
☐ SoftECC Input Register   ☐ REGCEA Pin

Port A Output Reset Options

☐ RSTA Pin (set/reset pin)   Output Reset Value (Hex): 0
☐ Reset Memory Latch   Reset Priority: CE (Latch or Register Enable)

READ Address Change A

☐ Read Address Change A



Component Name: bram_unit

Basic | Port A Options | **Port B Options** | Other Options | Summary

Memory Size

Port B Width: 32
Port B Depth : 256
The Width and Depth values are used for Read Operation in Port B

Operating Mode: Read First   Enable Port Type: Use ENB Pin

Port B Optional Output Registers

☑ Primitives Output Register   ☐ Core Output Register
☐ SoftECC Output Register   ☐ REGCEB Pin

Port B Output Reset Options

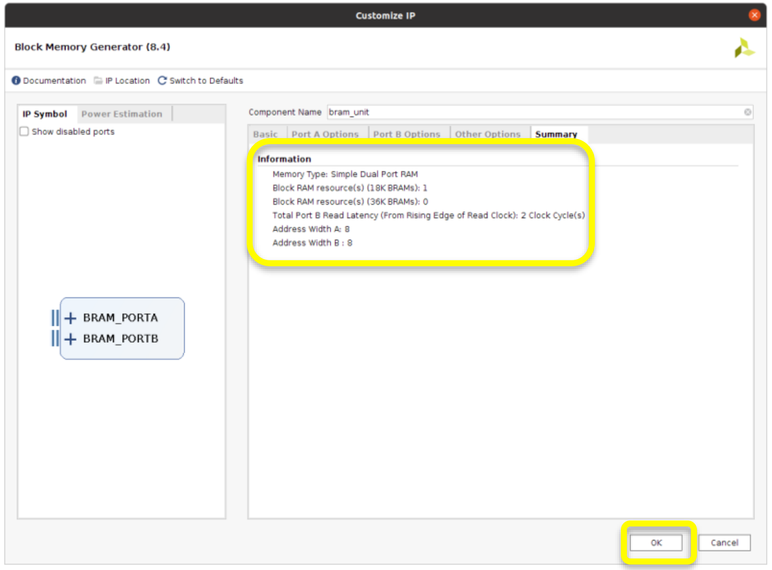
☐ RSTB Pin (set/reset pin)   Output Reset Value (Hex): 0
☐ Reset Memory Latch   Reset Priority: CE (Latch or Register Enable)

READ Address Change B

☐ Read Address Change B

IP Catalog Example: Block Memory Generator

- You can see the summary of memory unit that you generated.
  - Bit-width of ports, read latency …
- Finally click on OK and generate to create the IP block.


Customize IP

Block Memory Generator (8.4)

Documentation | IP Location | Switch to Defaults

IP Symbol | Power Estimation

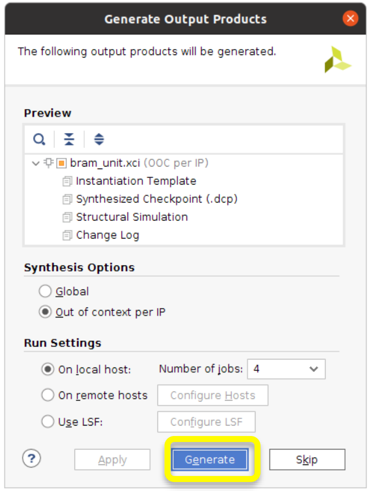
☐ Show disabled ports

Component Name: bram_unit

Basic | Port A Options | Port B Options | Other Options | Summary

Information

Memory Type: Simple Dual Port RAM
Block RAM resource(s) (18K BRAMs): 1
Block RAM resource(s) (36K BRAMs): 0
Total Port B Read Latency (From Rising Edge of Read Clock): 2 Clock Cycle(s)
Address Width A: 8
Address Width B : 8


Generate Output Products

The following output products will be generated.

Preview

bram_unit.xci (OOC per IP)
- Instantiation Template
- Synthesized Checkpoint (.dcp)
- Structural Simulation
- Change Log

Synthesis Options

○ Global
● Out of context per IP

Run Settings

● On local host: Number of jobs: 4
○ On remote hosts
○ Use LSF: