# Introduction to the Vivado HLS Tool Flow

2020.1

## Abstract

This lab introduces how to perform basic actions using the Vivado® High-Level Synthesis (HLS) design flow.

This lab should take approximately 45 minutes.

## Objectives

After completing this lab, you will be able to:

* Create a new project in the Vivado HLS tool GUI
* Simulate a C design by using a self-checking test bench
* Synthesize the design
* Perform design analysis using the Analysis Perspective view
* Perform co-simulation on a generated RTL design by using a provided C test bench
* Implement the design

## Introduction

This lab provides an introduction to the major features of the Vivado® High-Level Synthesis (HLS) tool GUI flow. You will use the Vivado HLS tool in GUI mode to create a project. You will also simulate, synthesize, and implement the design provided.

In this lab, you will be using a C design to implement a discrete cosine transformation (DCT). The function implements a 2D DCT algorithm by first processing each row of the input array via a 1D DCT, then processing the columns of the resulting array through the same 1D DCT. It calls the read\_data, dct\_2d, and write\_data functions.

The read\_data function is defined at line 54 and consists of two loops: RD\_Loop\_Row and RD\_Loop\_Col. The write\_data function is defined at line 66 and consists of two loops to perform writing the result. The dct\_2d function, defined at line 23, calls the dct\_1d function and performs transpose.

Finally, the dct\_1d function, defined at line 4, uses dct\_coeff\_table and performs the required function by implementing a basic iterative form of the 1D Type-II DCT algorithm.

The following figure shows the function hierarchy on the left-hand side, the loops in the order they are executed, and the flow of data on the right-hand side.

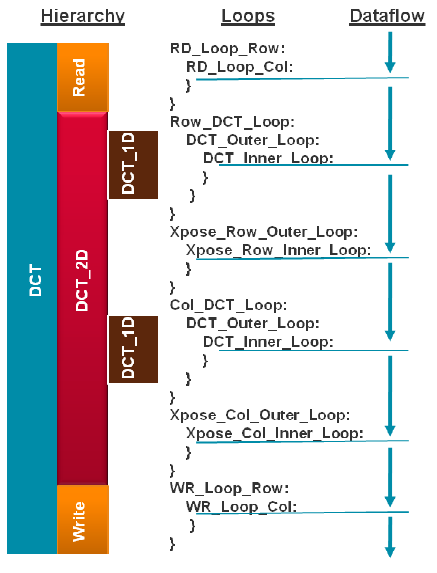


Figure 1‑1: Design Hierarchy and Dataflow

## General Flow

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step 1:  Creating a Vivado HLS Tool Project |  | Step 2:  Running  C  Simulation |  | Step 3:  Synthesizing the  Design |  | Step 4:  Using  Analysis Perspective |  | Step 5:  Performing Co- simulation |

|  |
| --- |
| Step 6:  Exporting  the RTL  as IP |

Creating a Vivado HLS Tool Project Step 1

In this step, you will launch the Vivado HLS tool GUI and create a new project for the provided C-based discrete cosine transformation (DCT) design.

There are a number of ways to launch the Vivado HLS tool. The two most popular mechanisms are shown here.

1-1. Launch the Vivado HLS tool.

1-1-1. For Windows 7: Select Start > All Programs > Xilinx Design Tools > Vivado 2020.1 > Vivado HLS 2020.1.

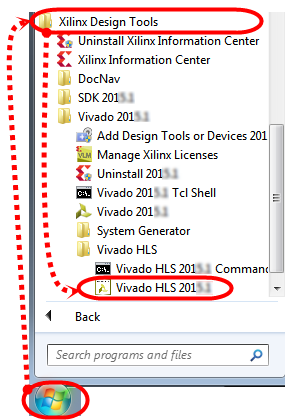


Figure 1‑2: Launching the Vivado HLS Tool

For Windows 10: Select Start > Xilinx Design Tools > Vivado HLS 2020.1.

-- OR --

Double-click the Vivado HLS shortcut icon () on the desktop.

The Vivado HLS tool opens to the Welcome window. From the Welcome window you can create a new project, open examples, and access documentation and examples.

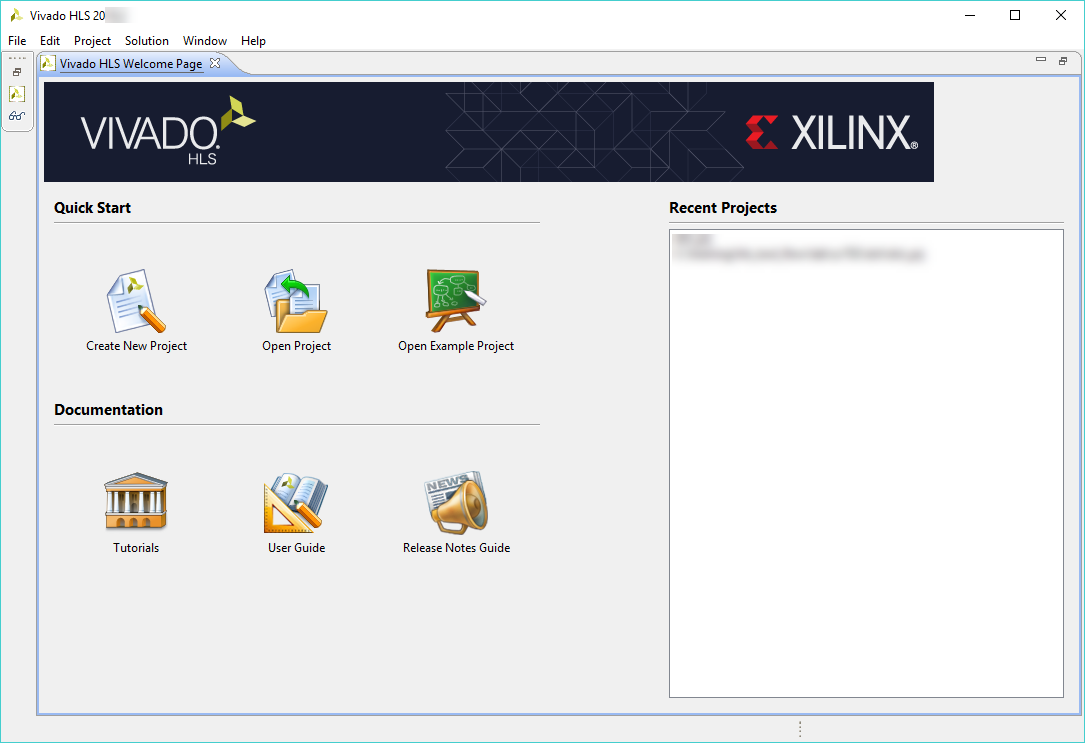


Figure 1‑3: Vivado HLS Welcome Page

Here you will learn to create a new Vivado HLS project from scratch.

1-2. Create a Vivado HLS project named dct\_prj.

1-2-1. Click Create New Project from the Welcome Page.

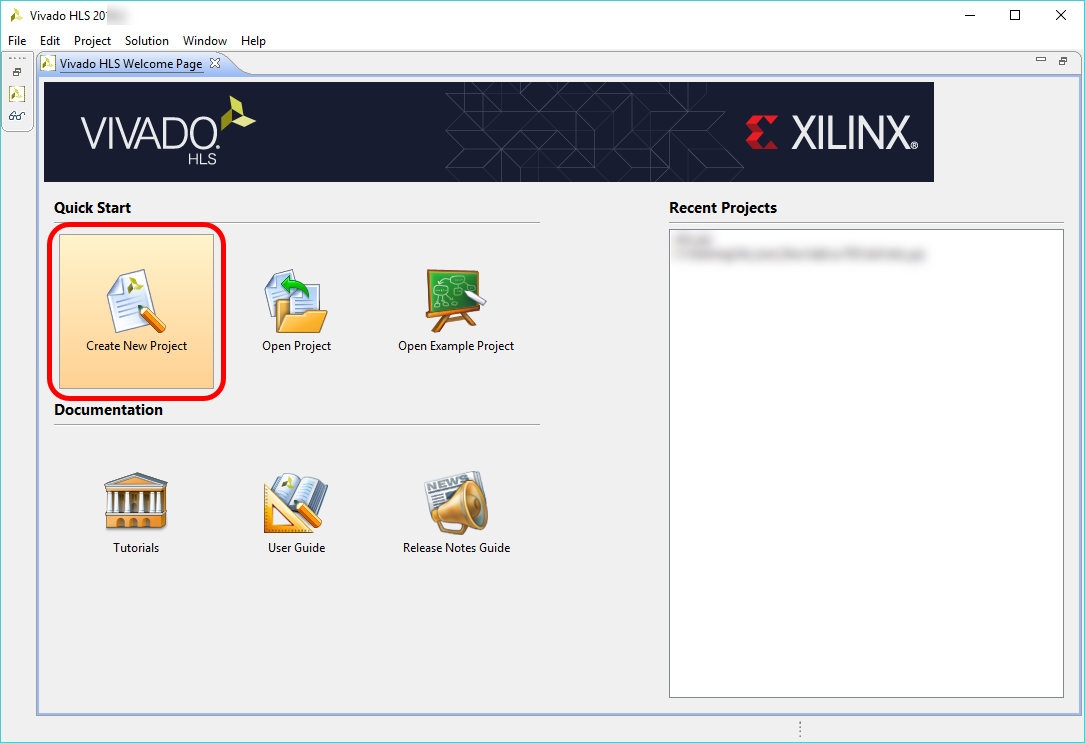


Figure 1‑4: Creating a New Vivado HLS Tool Project

1-3. The Project Configuration dialog box asks for a project name and location.

1-3-1. Enter dct\_prj in the Project name field (1).

1-3-2. Enter C:\xilinx\_trn\HLS\lab1\_hls\_tool\_flow in the Location field (2).

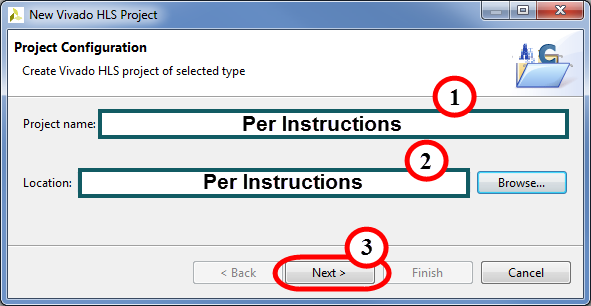


Figure 1‑5: Configuring a New HLS Project

1-3-3. Click Next (3).

1-4. The Add/Remove Files dialog box opens. Here you will be invited to add existing files or create new sources.

1-4-1. Click Add Files.

The Open dialog box opens.

Note: If do not have existing files at this moment and you want to create new ones, click New File.

1-4-2. Browse to C:\xilinx\_trn\HLS\lab1\_hls\_tool\_flow/source.

1-4-3. Select dct.c.

The Vivado HLS tool automatically adds the working directory (project directory) and any directory that contains C files added to the project to the search path. Hence, header files that reside in these directories are automatically included in the project (no need to explicitly specify them). You must specify the path to all other header files (if any) by clicking the Edit CFLAGS button.

1-4-4. Click Open to add these files.

Note that you can add compiler directives specific to each entry at this point.

1-4-5. Click Browse next to the Top Function field.

The Select Top function dialog box opens, which lists all the functions available from the specified source files.

1-4-6. Select dct (dct.c) from the list and click OK.

Note: You can also manually enter the name of the top function in the Top Function field.

In any C program, the top-level function is called main(). In the Vivado HLS design flow, you can specify any sub-function below main() as the top-level function for synthesis. You cannot synthesize the top-level function main().

The following are additional rules:

* Only one function is allowed as the top-level function for synthesis.
* Any sub-functions in the hierarchy under the top-level function for synthesis are also synthesized.
* If you want to synthesize functions that are not in the hierarchy under the top-level function for synthesis, you must merge the functions into a single top-level function for synthesis.

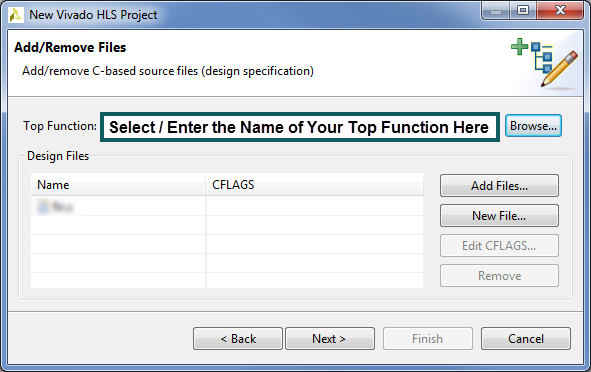


Figure 1‑6: Adding Files to a New Vivado HLS Project

1-4-7. Click Next.

1-5. Add any existing test bench files.

If you have (or want) any test bench files, they can be entered here. Sometimes the test bench is built into the synthesizable file.

1-5-1. Click Add Files.

1-5-2. Navigate to C:\xilinx\_trn\HLS\lab1\_hls\_tool\_flow/source.

1-5-3. Select dct\_test.c, in.dat, out.golden.dat.

1-5-4. Click Open to add these files.

1-5-5. Click Next.

1-6. Finally, it is time to specify some of the physical parameters of the design.

By default, solution1 is populated in the Solution Name field. No changes are required.

1-6-1. Set the clock period to 10.

You can leave the Uncertainty field blank.

1-6-2. Click the Browse button to select a part or board.

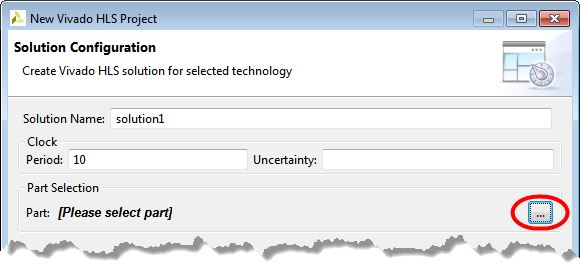


Figure 1‑7: Locating the Board Browse Button

1-6-3. Click Parts as shown below.

1-6-4. Enter xa7a12tcsg325-1q (МОЖНО ВЫБРАТЬ ЛЮБУЮ МИКРОСХЕМУ) in the Search field.

1-6-7. Click Finish.

You will see the created project in the Explorer tab.

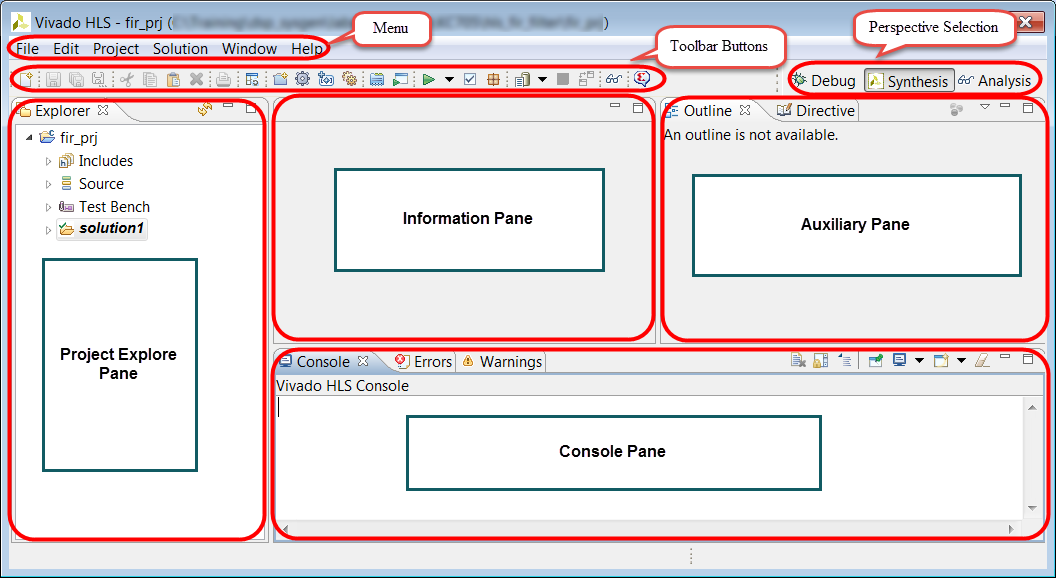


Figure 1‑9: Vivado HLS with Newly Created Project

The Vivado HLS tool GUI consists of various panes to proceed with the development work. You will see the created project in the Explorer view. Expand various sub-folders to see the entries under each sub-folder. The Source folder consists of source files associated with the project and the Test Bench folder consists of test bench files associated with the project.

Running C Simulation Step 2

After creating the project, the next step is to validate that the C function is correct before proceeding with synthesizing the design. In this step, you will validate the design by using the provided self-checking C test bench.

2-1. Become familiar with the provided source and test bench files.

2-1-1. Expand the Source folder in the Project Explorer pane.

2-1-2. Double-click dct.c to open the file.

This will open the source file in the Information pane.

2-1-3. Review the code and data structures.

2-1-4. Expand the Includes > C:\xilinx\_trn\HLS\lab1\_hls\_tool\_flow\source folder in the Project Explorer pane.

2-1-5. Double-click dct.h to open the header file.

2-1-6. Review the contents of the header file.

2-1-7. Expand the Test Bench folder in the Project Explorer pane.

2-1-8. Double-click dct\_test.c to open it in the Information pane.

This test bench is a self-checking test bench; i.e., the computed output is compared against a reference golden output and returns either pass or fail.

2-2. Simulate the Vivado HLS tool design.

2-2-1. Select Project > Run C Simulation or click the Run C Simulation icon ().

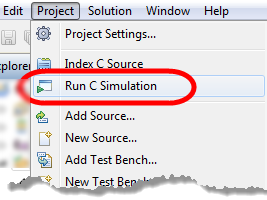


Figure 1‑10: Launching the C Simulation

The Run C Simulation dialog box opens.

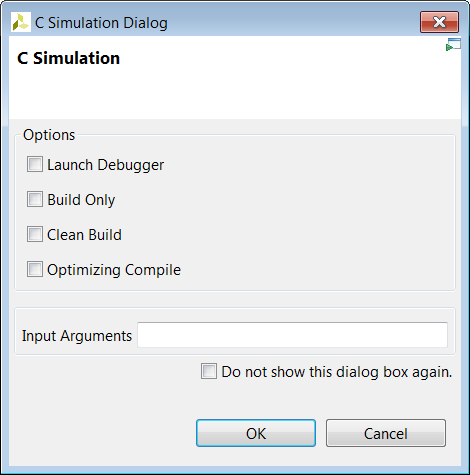


Figure 1‑11: C Simulation Dialog Box

Each of the options controls how simulation is run:

* Launch Debugger: After compilation the debug perspective automatically opens for you to step through the code.
* Build Only: Compiles the C code, but the simulation does not run.
* Clean Build: Removes any existing executable and object files before compiling the code.
* Optimizing Compile: Uses the gcc/g++ -O option (no debug info and this is mutually exclusive with debug options; this may run faster, but the difference is not substantial).

2-2-2. Select Default Options (i.e. select nothing).

2-2-3. Click OK.

The simulation log will be displayed in the editor pane.

2-3. View the simulation report.

The information generated by the Vivado HLS tool can be found in two places, both described here.

The first is the Console window, which reports not only the output produced by the code being simulated, but all of the simulation engine messages as well. The simulation log provides only a few simulation engine messages and the simulated code output.

2-3-1. Select the Console tab in the lower portion of the tool's GUI.

You may need to scroll to view all the output produced by the simulation.

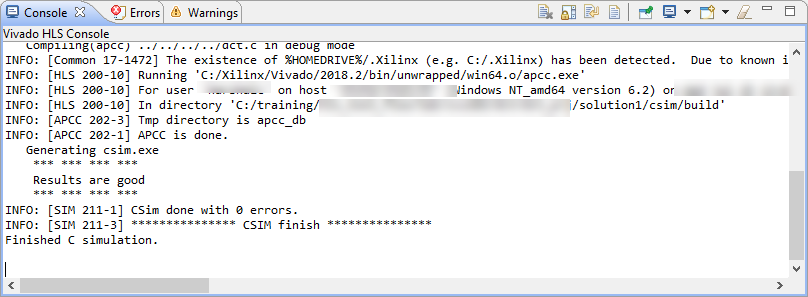


Figure 1‑12: Example Output After Simulation

The other location, described below, provides only a few simulation engine messages and the simulated code output. Typically this is opened after the simulation completes; however, if you need to access it after closing the log pane, here's how to access the simulation report.

2-3-2. Expand dct\_prj > solution1 > csim > report in the Explorer pane.

2-3-3. Double-click the log file name to open it in the editor pane.

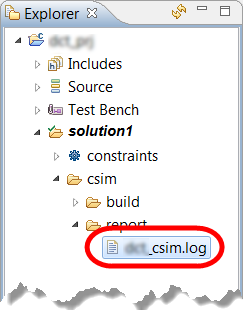


Figure 1‑13: Locating the Simulation Log File

You should see a "Results are good" message in the simulation log file and in the console area. If you do not see this message, ask for help from your instructor.

Synthesizing the Design Step 3

In this step, you will synthesize the design by using Vivado HLS tool defaults and analyze how many resources are utilized to implement the C design.

3-1. Synthesize the design.

3-1-1. Select Solution > Run C Synthesis > Active Solution or click the Run Synthesis icon in the menu bar.

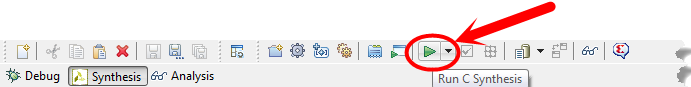


Figure 1‑14: Launching Synthesis

This option synthesizes the currently selected solution.

All solutions (or selected solutions) can be synthesized by using the drop-down menu next to the synthesis icon. You can synthesize all solutions or synthesize selected solutions in addition to the default.



Figure 1‑15: Options for What to Synthesize

When the synthesis completes, the Synthesis report will be displayed in the Information pane.

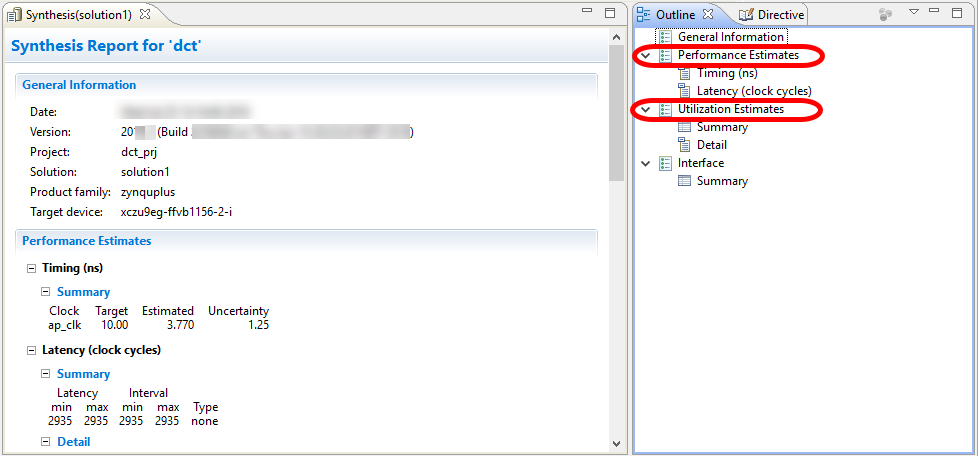


Figure 1‑16: Synthesis Report [Example]

The Synthesis report shows the performance and area estimates as well as estimated latency in the design.

3-1-2. Click Performance Estimates and Utilization Estimates in the Outline pane to answer the following question.

Question 1

Write down the following details from the Synthesis report:

* Estimated clock frequency:
* Worst case latency:
* Number of BRAM\_18K:
* Number of DSP48E used:
* Number of FFs used:
* Number of LUTs used:

3-1-3. Select Interface > Summary In the Outline pane.

The report also shows the top-level interface signals generated by the tools.

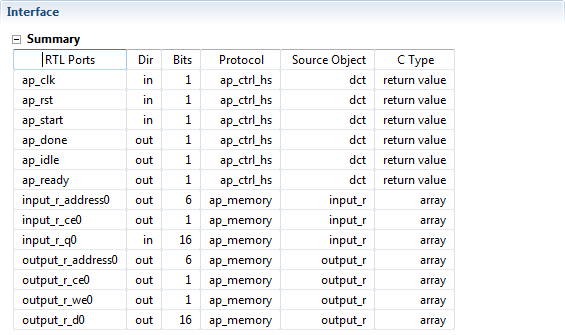


Figure 1‑17: Generated Interface Signals

You can see that ap\_clk and ap\_rst are automatically added. ap\_start, ap\_done, and ap\_idle are top-level signals used as handshaking signals to indicate when the design is able to accept the next computation command (ap\_idle), when the next computation is started (ap\_start), and when the computation is completed (ap\_done). Other signals are generated based on the design itself.

3-1-4. Select the Console tab.

The Synthesis log is available in the Vivado HLS Console.

Note that when the Solution1 > Syn folder is expanded in the Explorer view, it will show the report, systemc, verilog, and vhdl sub-folders under which the report files and generated source files (VHDL, Verilog, header, and cpp) are available. Double-clicking any of these entries will open the corresponding file in the Information pane.

Also note that the target design has hierarchical functions, and reports corresponding to lower-level functions are also created (in this example dct\_1d2\_csynth.rpt and dct\_2d\_csynth.rpt in addition to dct\_csynth.rpt). By default, the report for the top-level function is displayed in the Information pane once synthesis is completed.

Analyzing the Design Using the Analysis Perspective Step 4

The Analysis perspective is used after synthesis completes for analyzing the design in detail. This perspective provides considerable more details than the Synthesis report.

4-1. Switch to the Analysis perspective and understand the design behavior.

4-1-1. Select Solution > Open Analysis Perspective or click the Analysis perspective icon () to open the analysis viewer.

The Analysis perspective consists of five panes as shown below. Note that the module and loops hierarchies are displayed unexpanded by default.

The Module Hierarchy pane provides an overview of the entire RTL design:

* This view can navigate throughout the design hierarchy.
* The Module Hierarchy pane shows the resources and latency contribution for each block in the RTL hierarchy.

The Module Hierarchy pane shows both the performance and area information for the entire design and can be used to navigate through the hierarchy. The Performance Profile pane is visible and shows the performance details for this level of hierarchy. The information in these two panes is similar to the information reviewed earlier in the synthesis report.

The Performance view [Schedule Viewer(solution1)] is also shown in the right-hand side pane. This view shows how the operations in this particular block are scheduled into clock cycles.

* The left-hand column lists the resources.
* The top row lists the control states (C0 to C5) in the design. Control states are the internal states used by the Vivado HLS tool to schedule operations into clock cycles. There is a close correlation between the control states in the RTL FSM, but there is no one-to-one mapping.

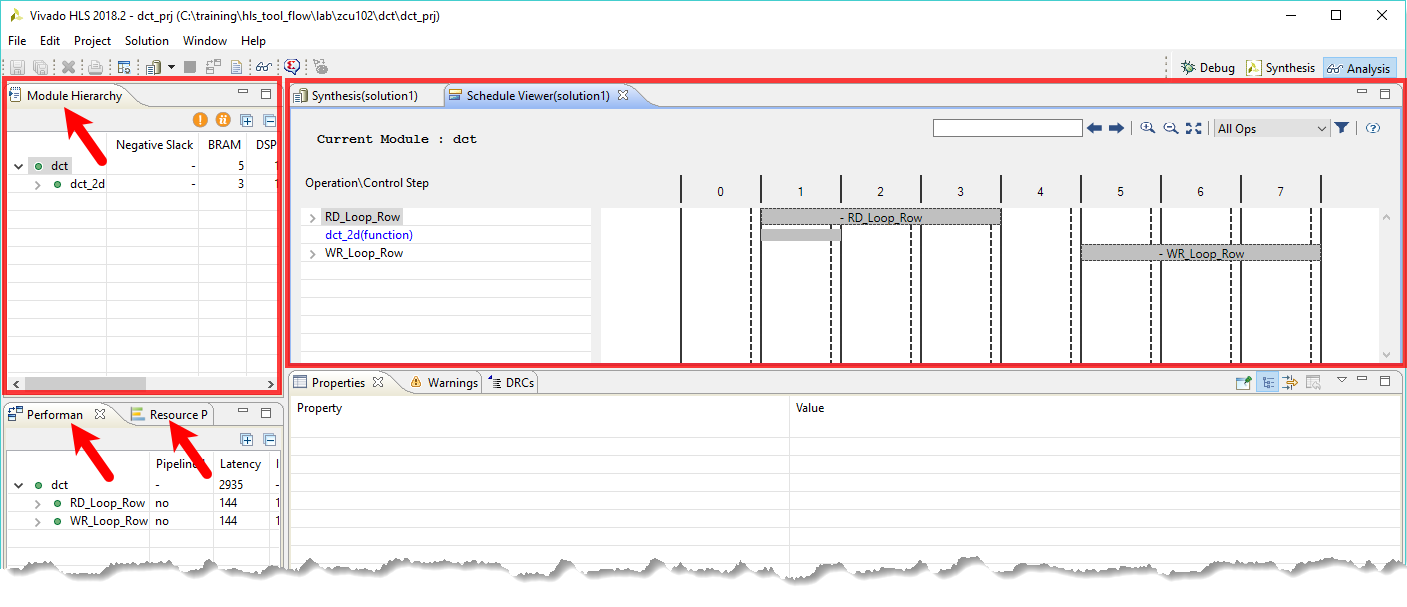


Figure 1‑18: Example of Analysis Perspective Panes

4-2. Analyze the performance of the dct module.

The dct module has three main resources:

* A loop called RD\_Loop\_Row.
* A sub-block called dct\_2d.
* A loop called WR\_Loop\_Row.

4-2-1. Click the loop RD\_Loop\_Row and click the sub-loop RD\_Loop\_Col.

4-2-2. Select tmp\_22(+) (1).

4-2-3. Right-click the highlighted box and select Goto Source (2).

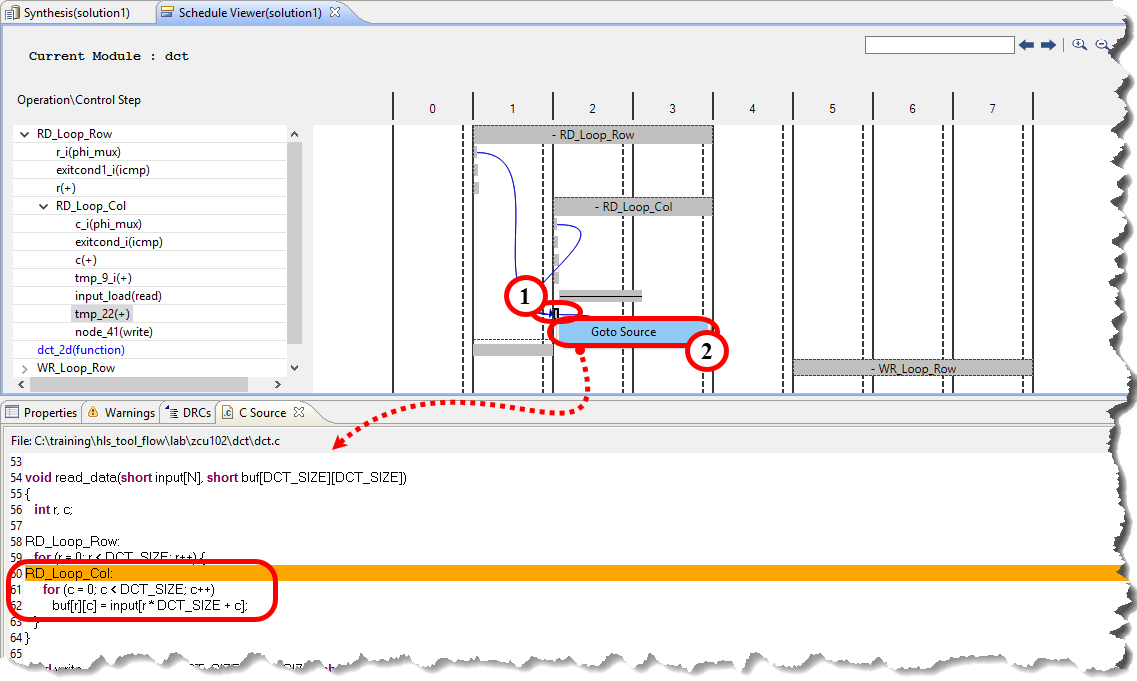


Figure 1‑19: Performance of the dct Loop Operation

The information presented in the Schedule view is explained below by reviewing the first set of resources to be executed (the RD\_Loop\_Row loop):

* The design starts in the 0 state.
* It then starts to execute the logic in the loop RD\_Loop\_Row.
* Note: In the first state of the loop, the exit condition is checked and there is an add operation.
* The loop executes over three states: 1, 2, and 3.

4-2-4. Review the Performance Profile pane information.

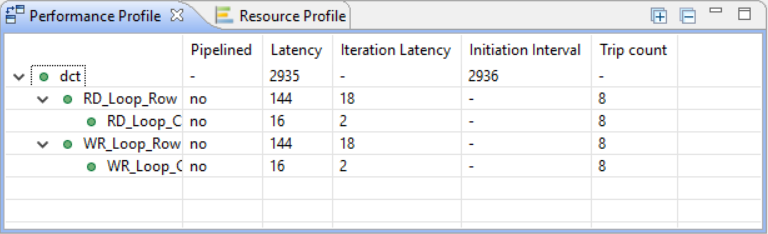


Figure 1‑20: Performance Profile Pane

The Performance Profile pane shows that this loop has a trip count of eight. It therefore iterates around these three states eight times.

The Performance Profile pane shows that the loop RD\_Loop\_Row takes 144 clock cycles to execute:

* One cycle at the start of the loop RD\_Loop\_Row.
* It takes 16 clock cycles to execute all operations of the loop RD\_Loop\_Col.
* Plus a clock cycle to return to the start of the loop RD\_Loop\_Row for a total of 18 cycles per loop iteration.
* Eight iterations of 18 cycles is why it takes 144 clock cycles to complete.

Within the loop RD\_Loop\_Col you can see there are some adders, a two-cycle read operation, and a write operation.

4-2-5. Click the loop WR\_Loop\_Row and click the sub-loop WR\_Loop\_Col.

4-2-6. Select tmp\_23(+) (1).

4-2-7. Right-click the highlighted box and select Goto Source (2).

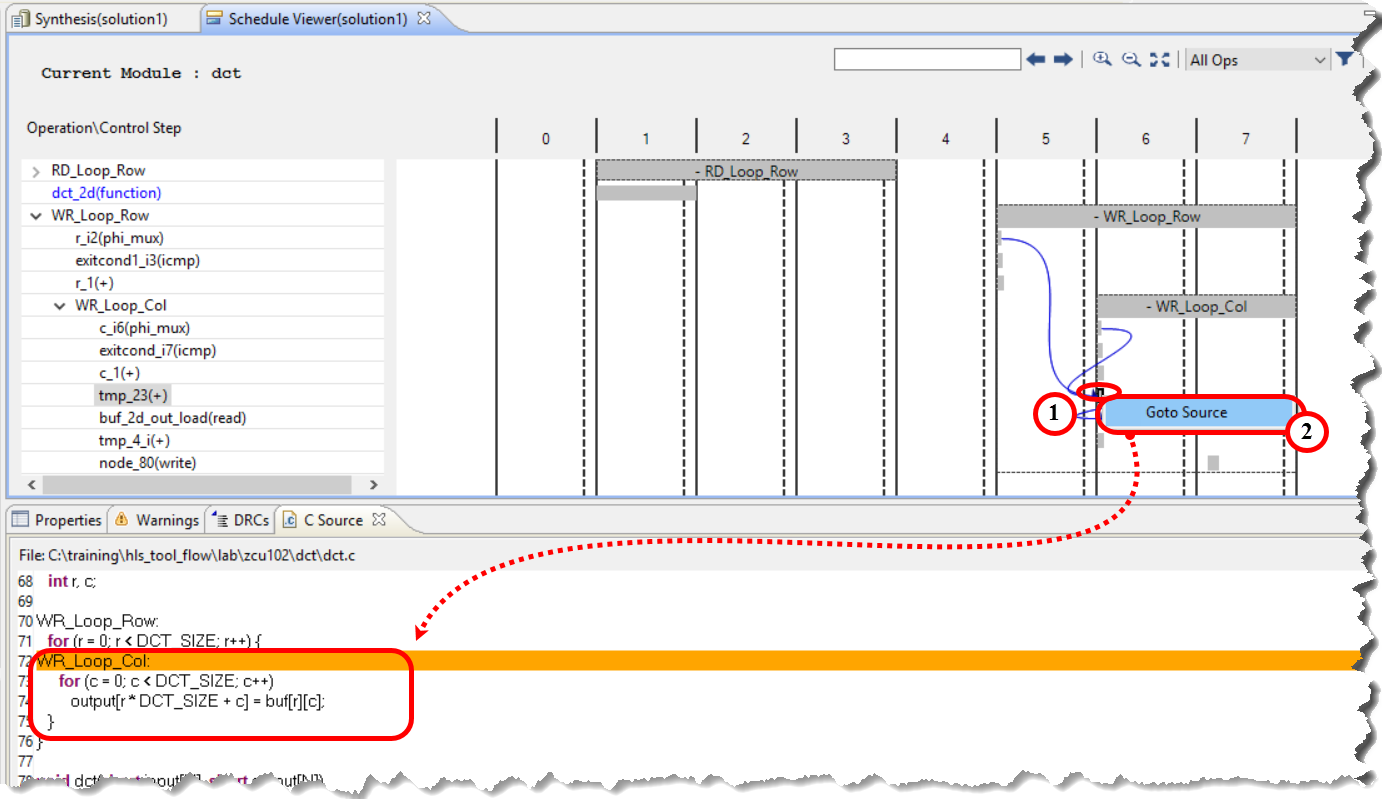


Figure 1‑21: C Source Code Correlation

You can see that the write operation is implementing the writing of data into the buf array from the input array variable.

4-3. Analyze the resource usage for the design.

4-3-1. Select the Module Hierarchy and Resource Profile tabs as shown below.

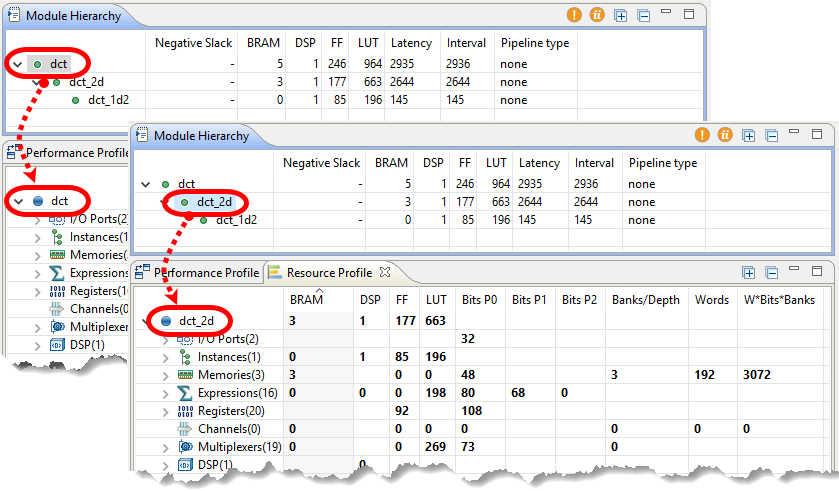


Figure 1‑22: Analysis Perspective – Resource Usage

The Resource Profile tab shows the resources used at this level of hierarchy. In this design, you can see that most of the resources are due to the instances—blocks that are instantiated inside this block. As per the function selected in the Module Hierarchy tab, you are able to view the resources used by the function in the Resource Profile tab.

4-3-2. Click the Synthesis perspective icon () to return to the Synthesis view.

Performing C/RTL Co-simulation Step 5

Now that you have performed high-level synthesis on the C design, you will perform RTL co-simulation on the generated RTL using the C test bench.

Run C/RTL co-simulation, selecting Verilog and skipping VHDL. Verify that the simulation passes.

5-1. Cosimulate the Vivado HLS tool design.

5-1-1. Select Solution > Run C/RTL Cosimulation or click the Run C/RTL Cosimulation icon ().

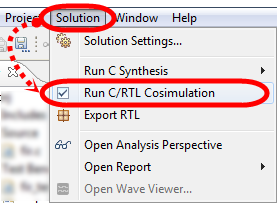


Figure 1‑23: Launching from the Menu

The Run C/RTL Co-simulation dialog box opens.

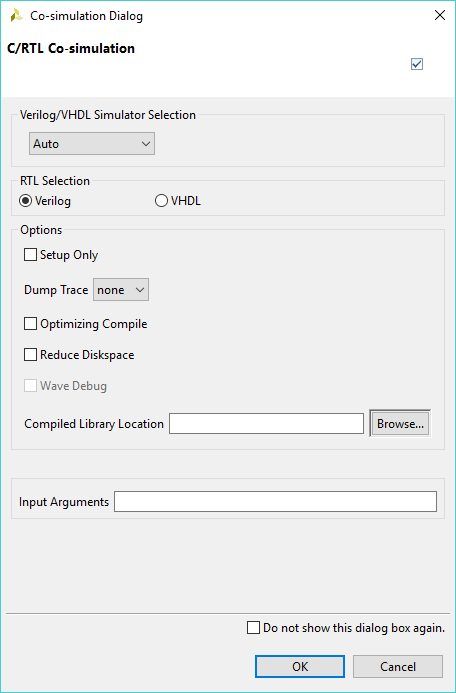


Figure 1‑24: Co-simulation Dialog Box

Each of the options controls how C/RTL co-simulation is run:

* RTL Selection: Select the RTL that is simulated (Verilog/VHDL).
* Setup Only: This creates all the files (wrappers, adapters, and scripts) required to run the simulation but does not execute the simulator.
* Dump Trace: During RTL verification, the trace files can be saved and viewed using an appropriate viewer. By selecting this option, the trace file will be saved to the <solution>/sim/<RTL> folder.
* Optimizing Compile: This ensures that a high level of optimization is used to compile the C test bench. Using this option increases the compile time but the simulation executes faster.
* Reduce Diskspace: Saves the results for all transactions before executing RTL simulation. In some cases, this can result in large data files. This option can be used to execute one transaction at a time and reduce the amount of disk space required for the file. If the function is executed N times in the C test bench, the reduce\_diskspace option ensures N separate RTL simulations are performed. This causes the simulation to run slower.
* Compiled Library Location: This specifies the location of the compiled library for a third-party RTL simulator.
* Input Arguments: This allows the specification of any arguments required by the test bench.

5-1-2. Select **the d**efault options (i.e. select nothing).

5-1-3. Click OK.

The simulation log will be displayed in the editor pane.

5-2. View the Cosimulation report.

The information generated by the Vivado HLS tool can be found in two places, both described here.

The first is the Console window, which reports not only the output produced by the code being simulated, but all of the simulation engine messages as well. The simulation log provides only a few simulation engine messages and the simulated code output.

5-2-1. Select the Console tab in the lower portion of the tool's GUI.

You may need to scroll to view all the output produced by the cosimulation.

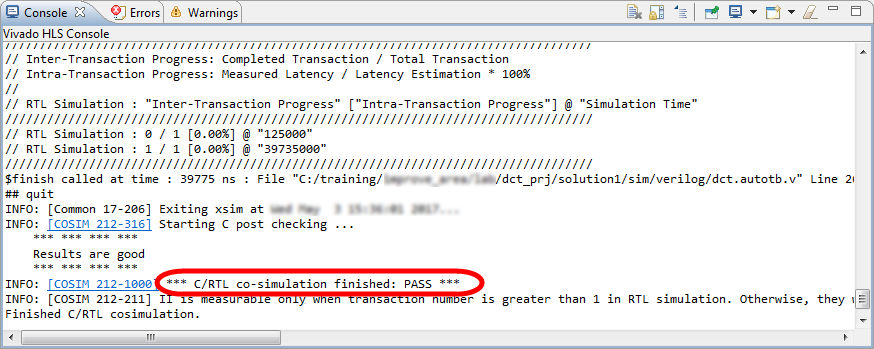


Figure 1‑25: Example Output After a C/RTL Co-simulation

The other location, described below, provides only a few simulation engine messages and the simulated code output. Typically this is opened after the simulation completes; however, if you need to access it after closing the log pane, here is how to access the simulation report.

5-2-2. Expand dct\_prj > solution1 > sim > report in the Explorer pane.

5-2-3. Double-click the log file name to open it in the editor pane.

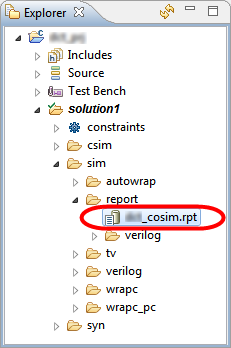


Figure 1‑26: Locating the Co-simulation Log File

The Cosimulation Report in HTML format will be displayed in the main viewing area.

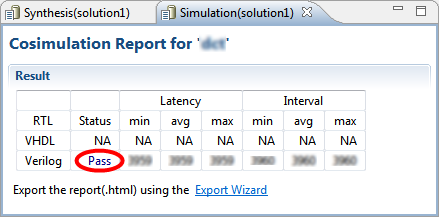


Figure 1‑27: Cosimulation Report – HTML

You can quickly verify the cosimulation status here.

This process will take a few minutes to complete. After C/RTL cosimulation has been completed, the Cosimulation report will be accessible in the Information pane, including the latency information.

Also, in the Console tab, notice the "Results are Good " message that is displayed.

Exporting the RTL as an IP Core Step 6

In this step you will export the RTL as an IP core to be used with the top-level design.

6-1. Export the RTL, selecting Verilog as the language.

6-1-1. Select Solution > Export RTL.

Alternatively, you can click the  toolbar button.

Note: If you see the Feedback Request dialog box, click Cancel.

6-1-2. Ensure that IP Catalog is selected from the Format Selection drop-down list.

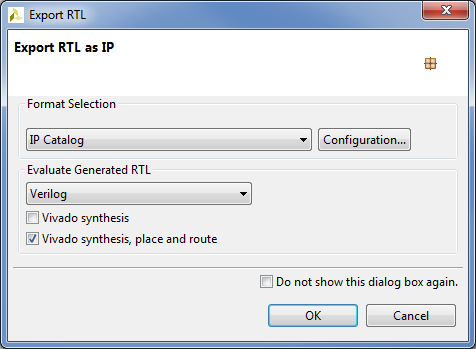


Figure 1‑28: Export RTL Dialog Box

6-1-3. Click Configuration next to the Format Selection drop-down list.

Notice that you can provide information about the IP, such as the vendor, library, version, and description in the IP Identification dialog box.

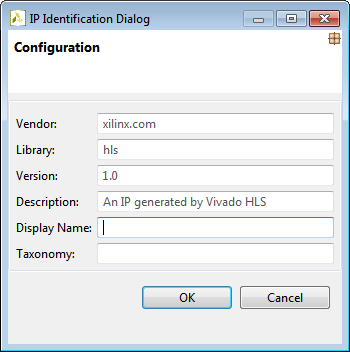


Figure 1‑29: IP Identification Dialog Box

6-1-4. Click Cancel in the IP Identification dialog box.

6-1-5. Make sure that Verilog is selected as the RTL to be evaluated.

6-1-6. Select the Vivado synthesis, place and route option.

This will perform Vivado RTL synthesis and implementation on the generated IP. Implementation is run to evaluate and provide confidence that the RTL will meet its estimated timing and area goals and that these results are not included as part of the exported package.

6-1-7. Click OK in the Export RTL dialog box.

You can observe the progress in the Console tab. When the run is complete, the Implementation Report is displayed in the Information pane.

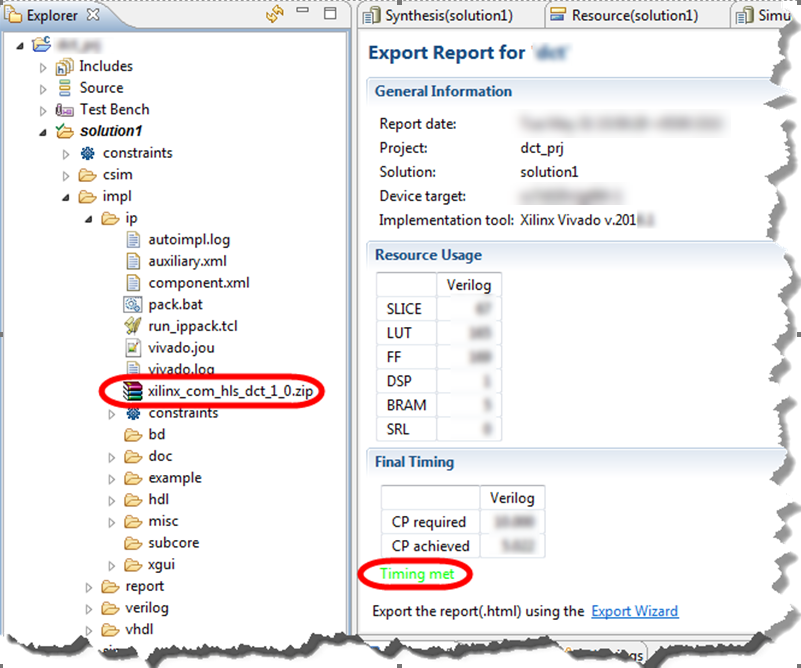


Figure 1‑30: Implementation Results in the Vivado HLS Tool

Once implementation completes, the Implementation report will open in the Information pane. The final timing of the implemented design has been achieved.

The exported IP is available in the <solution\_directory>\impl\ip folder.

6-1-8. Select File > Exit to close the Vivado HLS tool.

## Summary

In this lab, you learned how to create a new Vivado HLS tool project in the GUI and execute major steps (simulate, synthesize, co-simulate and export) in the HLS Vivado Design Suite flow.

In the following labs, you will examine some of the software reports, determine how the design was implemented, and determine whether or not design goals for area and performance were met.

## Answers

1. Write down the following details from the Synthesis report:

Estimated clock frequency: 3.77

Worst case latency: 2935

Number of BRAM\_18K: 5

Number of DSP48E used: 1

Number of FFs used: 246

Number of LUTs used: 964

# 