Vitis HLS Design Flow
Vitis HLS Project Creation, Vivado Hardware Creation, and Vitis Software Control

Joseph Cavallaro
Rice University

27 September 2022

Last Lecture

- ❒ Video available on Model Composer -> Vivado -> Vitis based on the treeadd example posted to Canvas in the Media Gallery tab
- ❒ Lab tutorial – demo on I/O from/to workspace
  - ⇒ We will use Matlab timeseries data files to load and display data for Model Composer
- ❒ Xilinx intro slides on Vitis HLS

Today

- Into to Vitis HLS and design flow
  - ⇒ Vivado HLS renamed to Vitis HLS to focus on the software instead of hardware aspects. (Some notes and slides may still say Vivado HLS.)
- Vitis HLS User Guide in Files/CAD_Tool_Documentation/Vitis_HLS
  - ⇒ ug1399-vitis-hls.pdf
- Moving away from Model Composer / Simulink HW Cosim to Vitis program control
- Demo of Vitis HLS -> Vivado -> Vitis on similar treeadd example
- AXI lite to get started for single computation
- Next will be loops on AXI lite and then AXI stream communication

Preview of Project 3: High-level Synthesis Tools - Matrix Multiplication

- Use Vitis HLS C Synthesis tool to design a matrix multiplication system which could be easily targeted to ASIC/FPGA.
- Verify your C++ code and HDL code to make sure that they function correctly. Check the scheduling result by looking at the Gantt chart.
- Repeat until a good balance between system throughput and hardware resource costs – at least 2 optimizations.
- Export your Vitis HLS code to Vivado for Vitis C++ code hardware control

Vitis HLS Flow

- ❒ Create a project after starting Vitis HLS in GUI mode
- ❒ Run C simulation
  - ⇒ to understand the design behavior
- ❒ Run the debugger
  - ⇒ to see how the top-level module works
- ❒ Synthesize the design
- ❒ Analyze the generated output using the Analysis perspective
- ❒ Run C/RTL co-simulation
  - ⇒ to perform RTL simulation
- ❒ Export RTL for use in Vivado to be integrated with Zynq ARM processor core

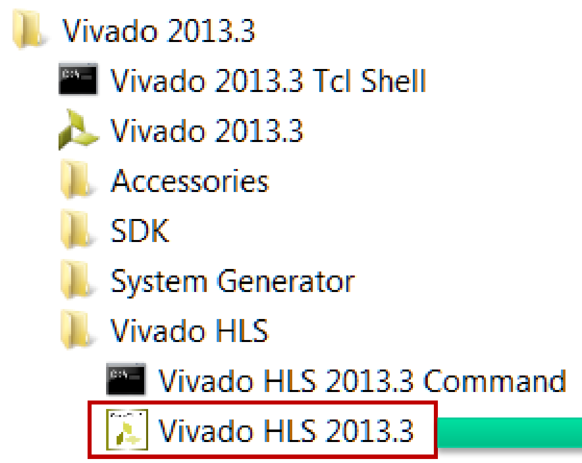
Vitis HLS Style Optimizations

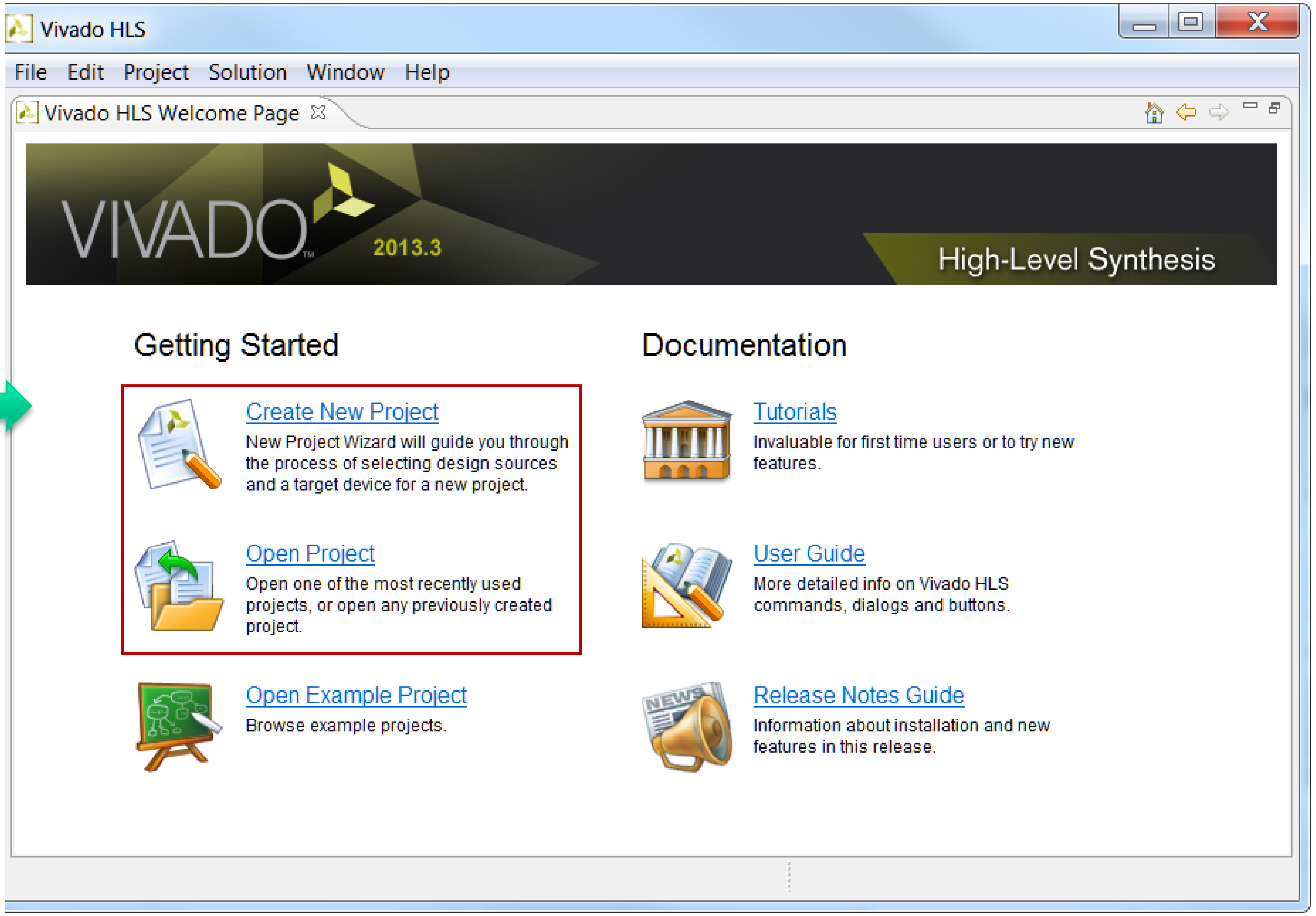
- IN-LINING
- PIPELINING
- DATAFLOW
- RESHAPE

- All these optimizations benefit from understanding Data Flow Graphs and Single Assignment Form

- Our initial demo will be a simple design

Invoke Vitis HLS from Windows Menu


- Vivado 2013.3
  - Vivado 2013.3 Tcl Shell
  - Vivado 2013.3
  - Accessories
  - SDK
  - System Generator
  - Vivado HLS
    - Vivado HLS 2013.3 Command
    - Vivado HLS 2013.3



Getting Started

Create New Project
New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Open Project
Open one of the most recently used projects, or open any previously created project.

Open Example Project
Browse example projects.

Documentation

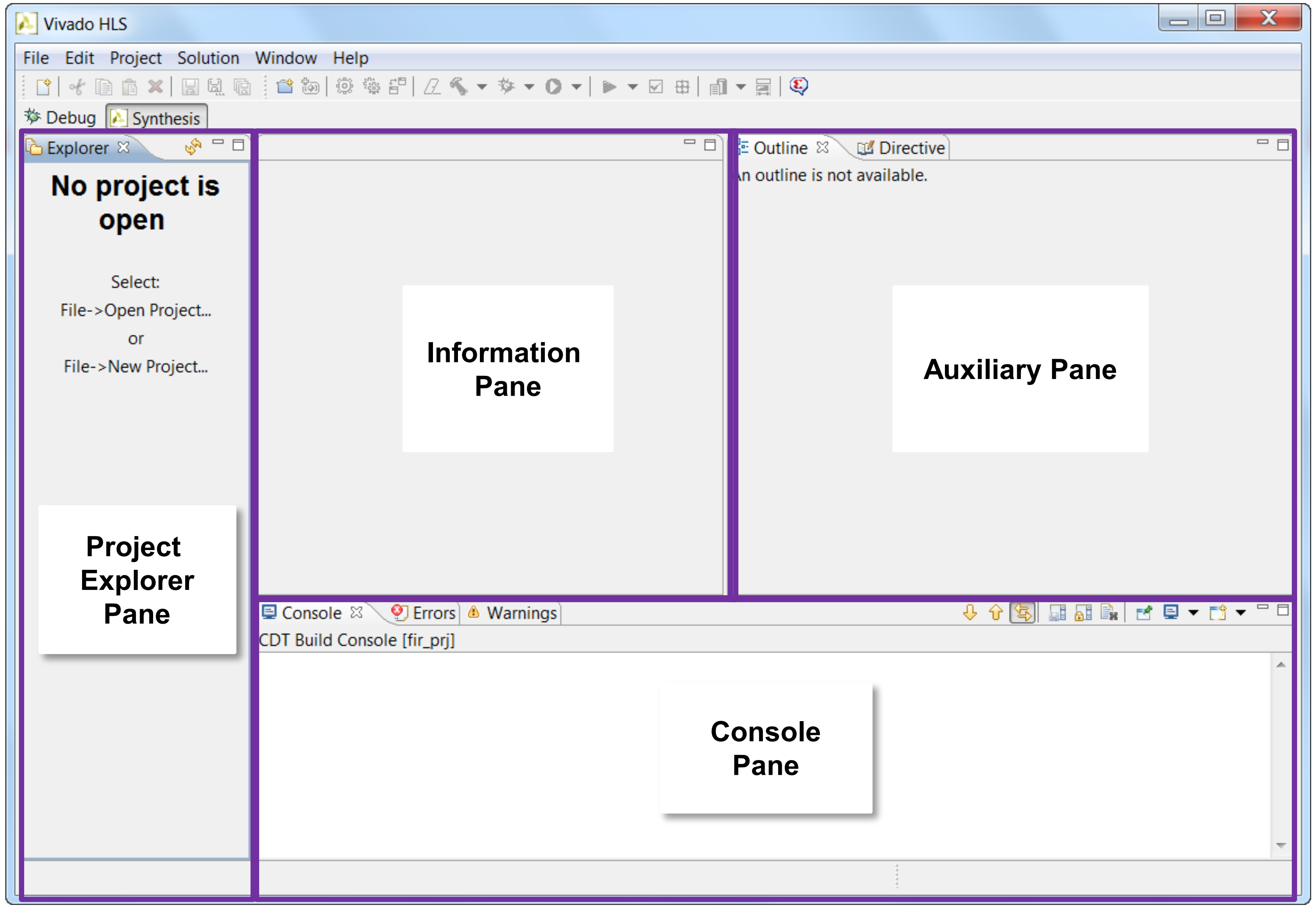
Tutorials
Invaluable for first time users or to try new features.

User Guide
More detailed info on Vivado HLS commands, dialogs and buttons.

Release Notes Guide
Information about installation and new features in this release.

The first step is to open or create a project

Vitis HLS GUI



No project is open

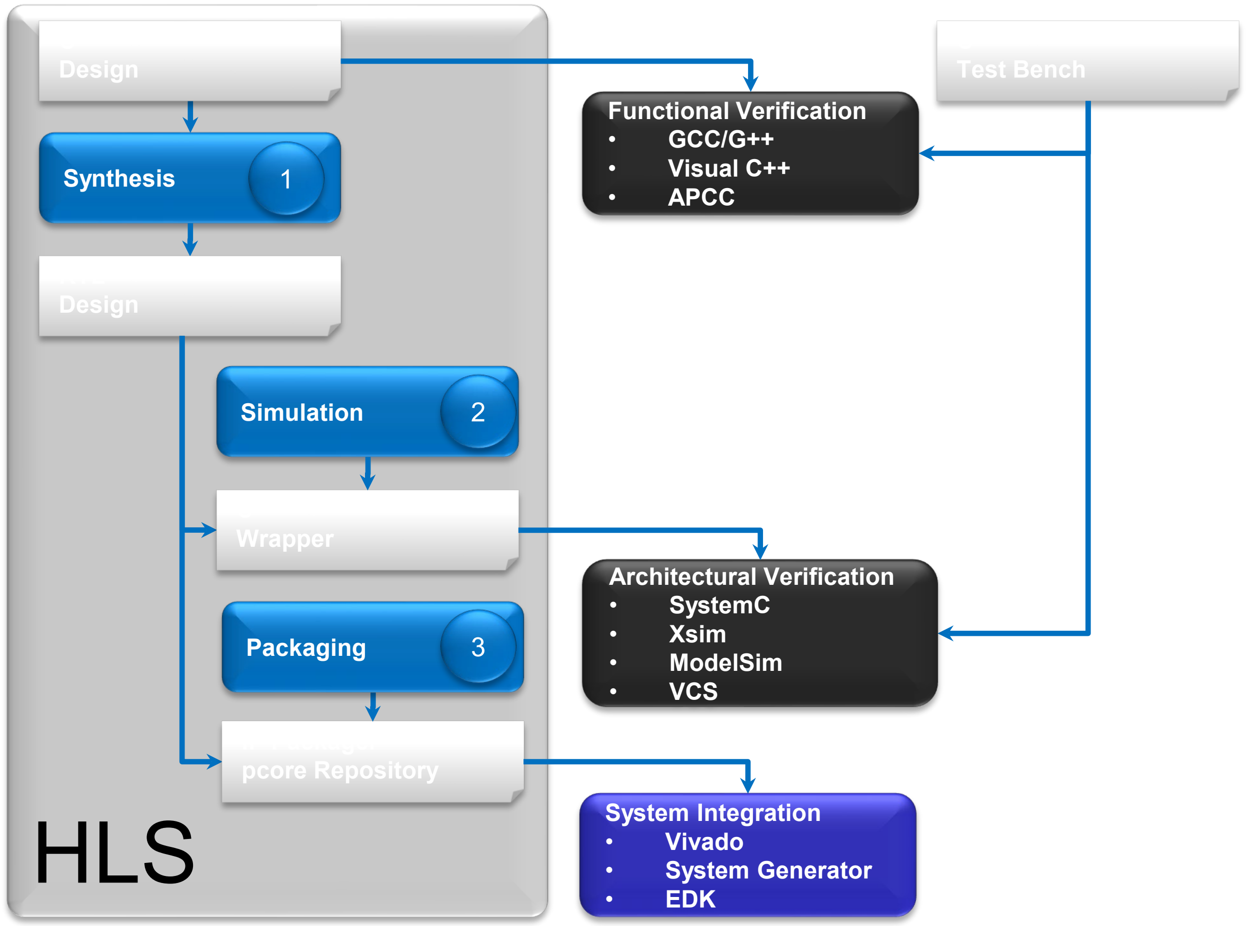
Select:
File->Open Project...
or
File->New Project...

An outline is not available.

CDT Build Console [fir_prj]

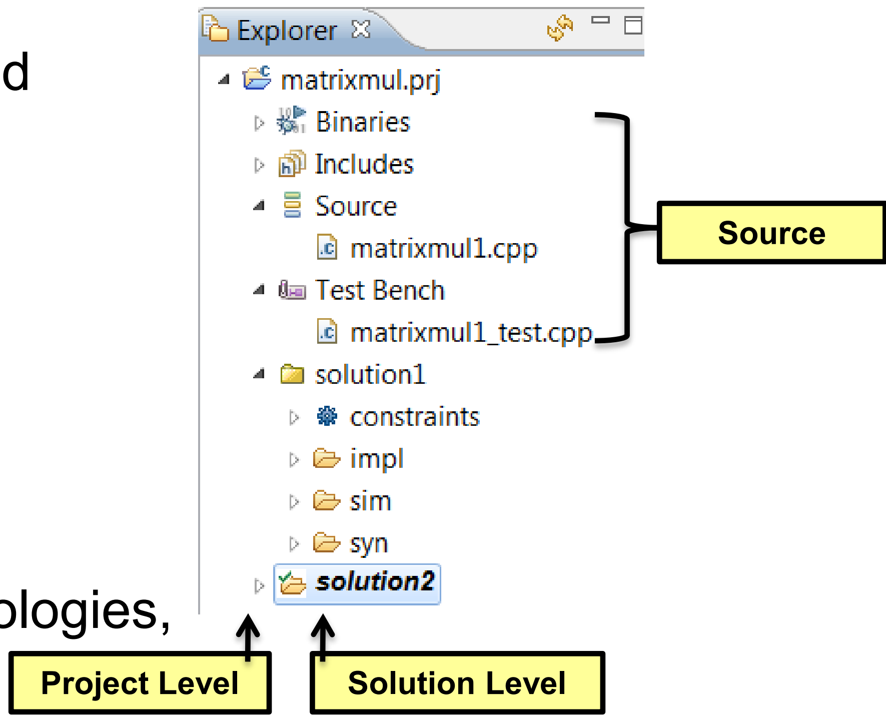
Vitis HLS Design Flow

- ❐ Starts at C (+ constraints)
  - ⇒ C
  - ⇒ C++
  - ⇒ SystemC
- ❐ Produces RTL (+ constraints)
  - ⇒ Verilog
  - ⇒ VHDL
  - ⇒ SystemC
- ❐ Automates Flow
  - ⇒ Verification
  - ⇒ Implementation



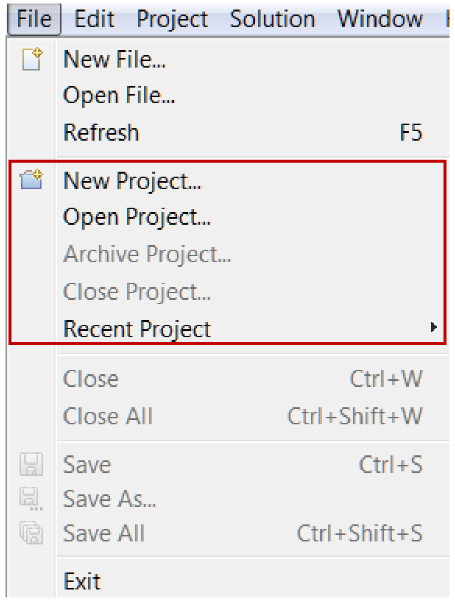
Vitis HLS Projects and Solutions

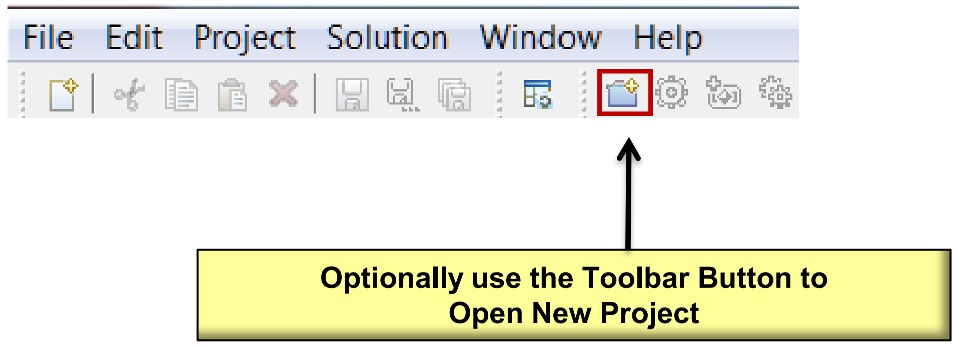
- Vitis HLS is project based
  - ⇒ A project specifies the source code which will be synthesized
  - ⇒ Each project is based on one set of source code
  - ⇒ Each project has a user specified name
- A project can contain multiple solutions
  - ⇒ Solutions are different implementations of the same code
  - ⇒ Auto-named solution1, solution2, etc.
  - ⇒ Supports user specified names
  - ⇒ Solutions can have different clock frequencies, target technologies, synthesis directives
- Projects and solutions are stored in a hierarchical directory structure
  - ⇒ Top-level is the project directory
  - ⇒ The disk directory structure is identical to the structure shown in the GUI project explorer (except for source code location)



Vitis HLS Step 1: Create or Open a project

- Start a new project
  - ⇒ The GUI will start the project wizard to guide you through all the steps

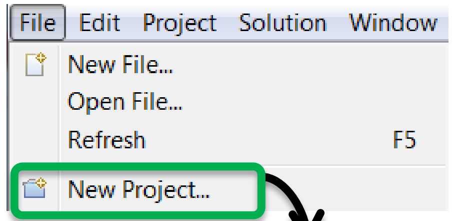



File Edit Project Solution Window Help
Optionally use the Toolbar Button to Open New Project

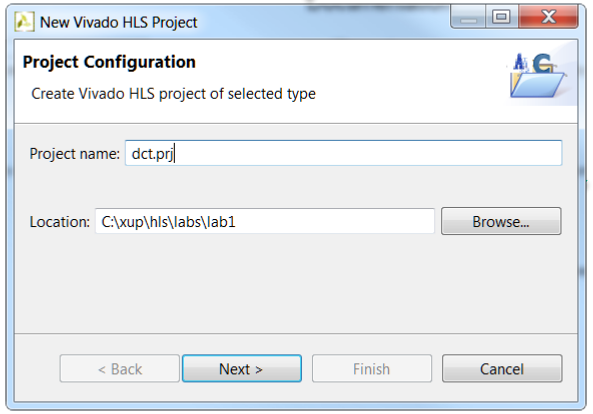
- Open an existing project
  - ⇒ All results, reports and directives are automatically saved/remembered
  - ⇒ Use “Recent Project” menu for quick access

Vitis HLS Project Wizard

- The Project Wizard guides users through the steps of opening a new project



Step-by-step guide …


New Vivado HLS Project

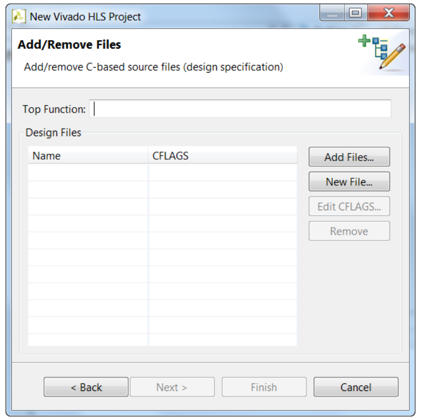
Project Configuration

Create Vivado HLS project of selected type

Project name: dct.prj

Location: C:\xup\hls\labs\lab1

Define project and directory


New Vivado HLS Project

Add/Remove Files

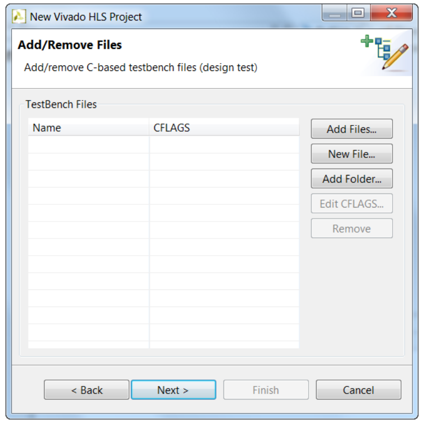
Add/remove C-based source files (design specification)

Top Function:

Design Files

Name	CFLAGS

Add design source files


New Vivado HLS Project

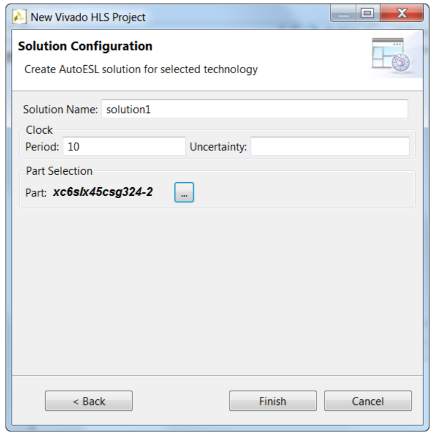
Add/Remove Files

Add/remove C-based testbench files (design test)

TestBench Files

Name	CFLAGS

Specify test bench files


New Vivado HLS Project

Solution Configuration

Create AutoESL solution for selected technology

Solution Name: solution1

Clock

Period: 10 Uncertainty:

Part Selection

Part: ***xc6slx45csg324-2***

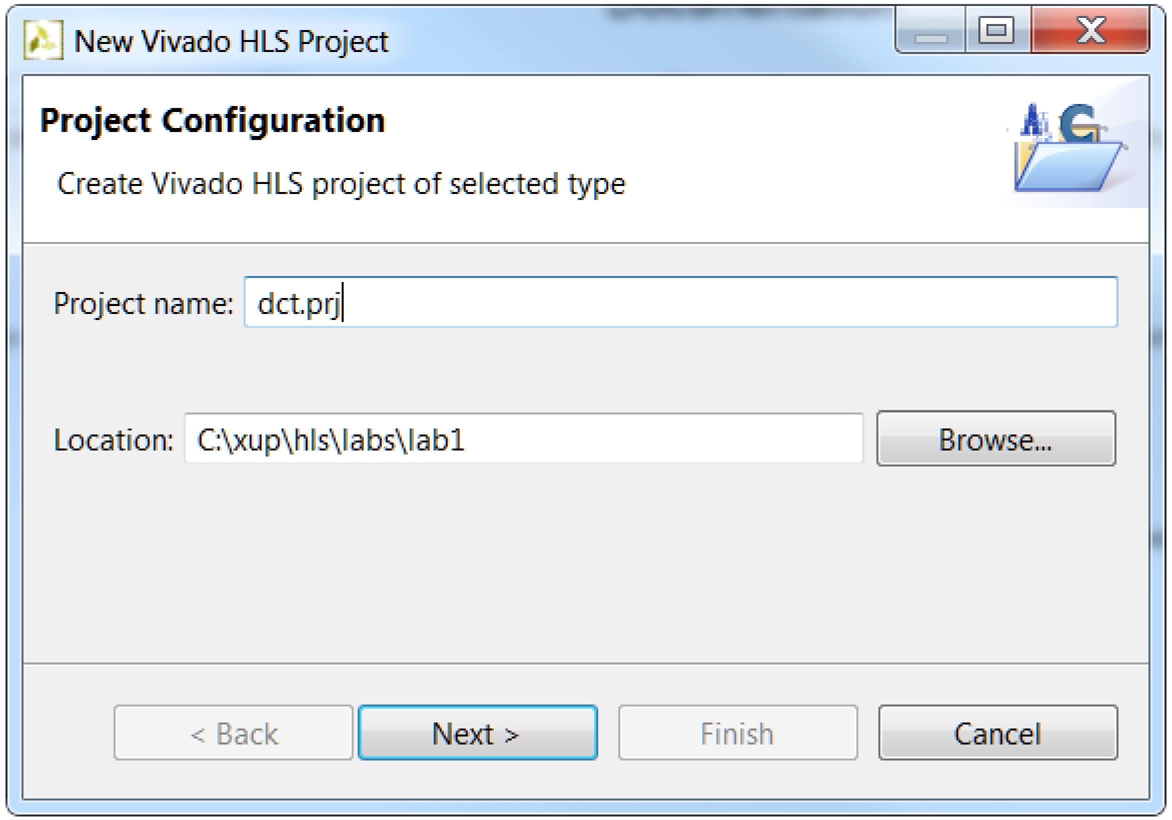
Specify clock and select part

Project Level Information

1st Solution Information

Define Project & Directory

- Define the project name
  - Note, here the project is given the extension .prj
  - A useful way of seeing it's a project (and not just another directory) when browsing
- Browse to the location of the project
  - ⇒ In this example, project directory "dct.prj" will be created inside directory "lab1"


New Vivado HLS Project

Project Configuration

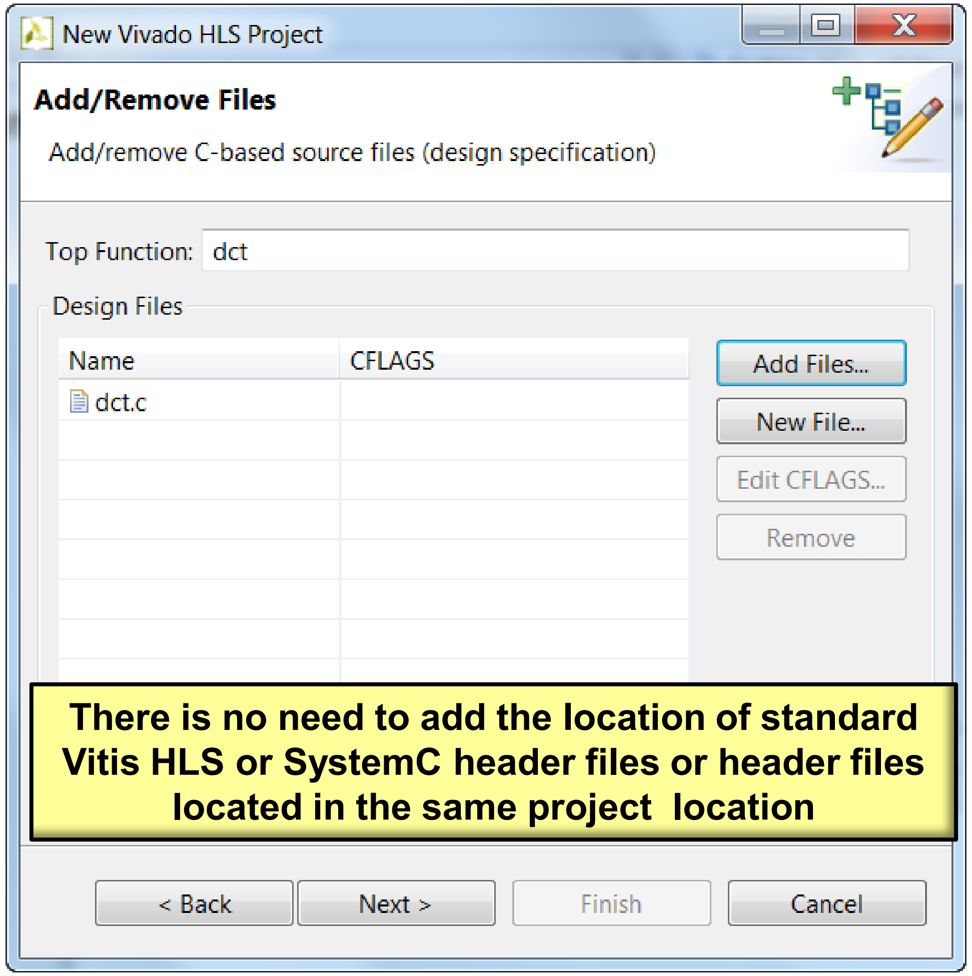
Create Vivado HLS project of selected type

Project name: dct.prj

Location: C:\xup\hls\labs\lab1

Add Design Source Files

- Add Design Source Files
  - This allows Vitis HLS to determine the top-level design for synthesis, from the test bench & associated files
- Add Files…
  - ⇒ Select the source code file(s)
  - ⇒ The CTRL and SHIFT keys can be used to add multiple files
  - ⇒ No need to include headers (.h) if they reside in the same directory
- Select File and Edit CFLAGS…
  - If required, specify C compile arguments using the "Edit CFLAGS…"
    - Define macros: -DVERSION1
    - Location of any (header) files not in the same directory as the source: -I../include



Top Function: dct

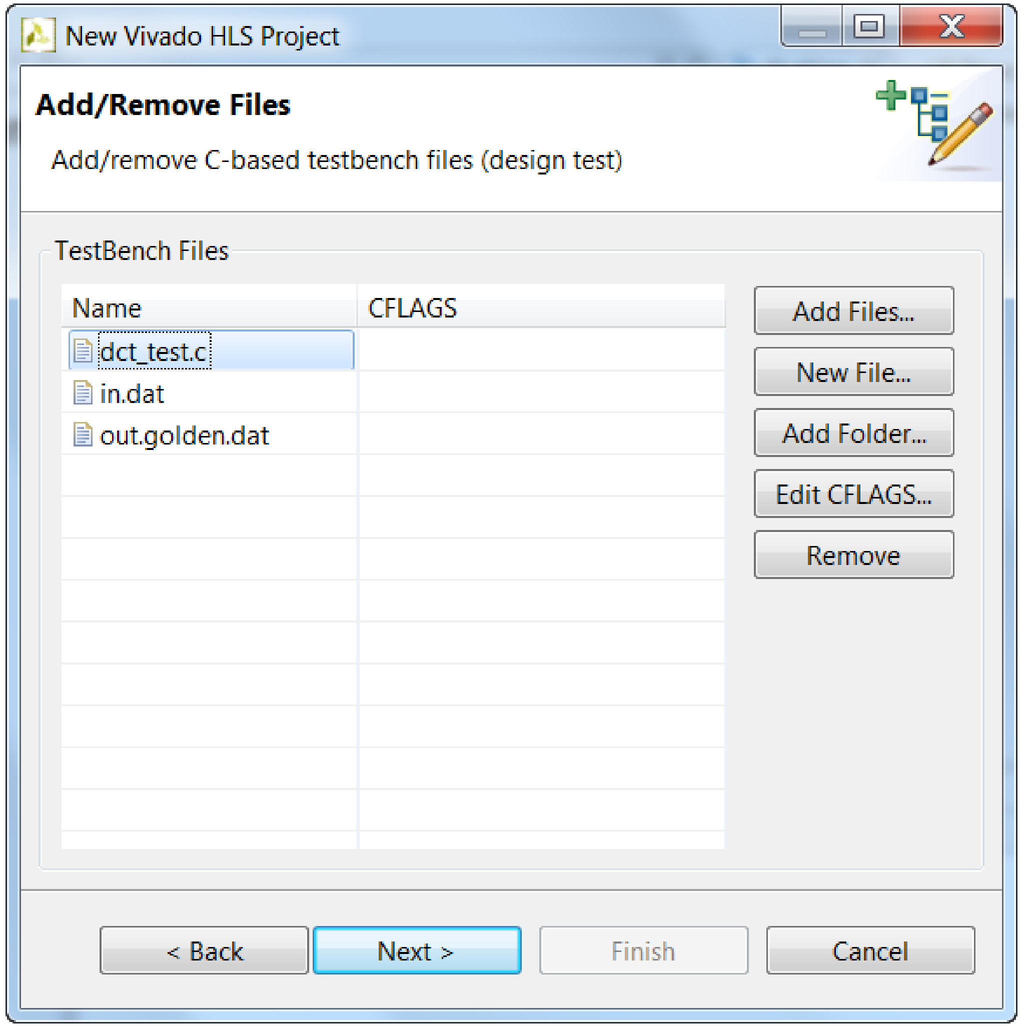
Design Files

Name	CFLAGS
dct.c	

There is no need to add the location of standard Vitis HLS or SystemC header files or header files located in the same project location

Specify Test Bench Files

- Use “Add Files” to include the test bench
  - ⇒ Vitis HLS will re-use these to verify the RTL using co-simulation
- And all files referenced by the test bench
  - ⇒ The RTL simulation will be executed in a different directory (Ensures the original results are not over-written)
  - ⇒ Vitis HLS needs to also copy any files accessed by the test bench
  - ⇒ Input data and output results (*.dat) are shown in this example
- Add Folders
  - ⇒ If the test bench uses relative paths like “sub_directory/my_file.dat” you can add “sub_directory” as a folder/directory
- Click on testbench, Use “Edit CFLAGS…”
  - ⇒ To add any C compile flags required for compilation
  - ⇒ -D HW_COSIM



Name	CFLAGS
dct_test.c	
in.dat	
out.golden.dat	

Test benches I

- The test bench should be in a separate file
- Or excluded from synthesis
  - ⇒ The Macro __SYNTHESIS__ can be used to isolate code <u>which will not be synthesized</u>
    - This macro is defined when Vitis HLS parses any code (-D__SYNTHESIS__)

```
// test.c
#include <stdio.h>
void test (int d[10]) {
  int acc = 0;
  int i;
  for (i=0;i<10;i++) {
    acc += d[i];
    d[i] = acc;
  }
}
#ifndef __SYNTHESIS__
int main () {
  int d[10], i;
  for (i=0;i<10;i++) {
    d[i]   = i;
  }
  test(d);
  for (i=0;i<10;i++) {
    printf("%d %d\n", i, d[i]);
  }
  return 0;
}
#endif
```

Design to be synthesized

<u>Test Bench</u>
Nothing in this ifndef will be read by Vitis HLS
(will be read by gcc)

Test benches II

- Ideal test bench
  - ⇒ Should be self checking
    - RTL verification will re-use the C test bench
  - ⇒ If the test bench is self-checking
    - Allows RTL Verification to be run without a requirement to check the results again
  - ⇒ RTL verification “passes” if the test bench return value is 0 (zero)
    - Actively return a 0 if the simulation passes

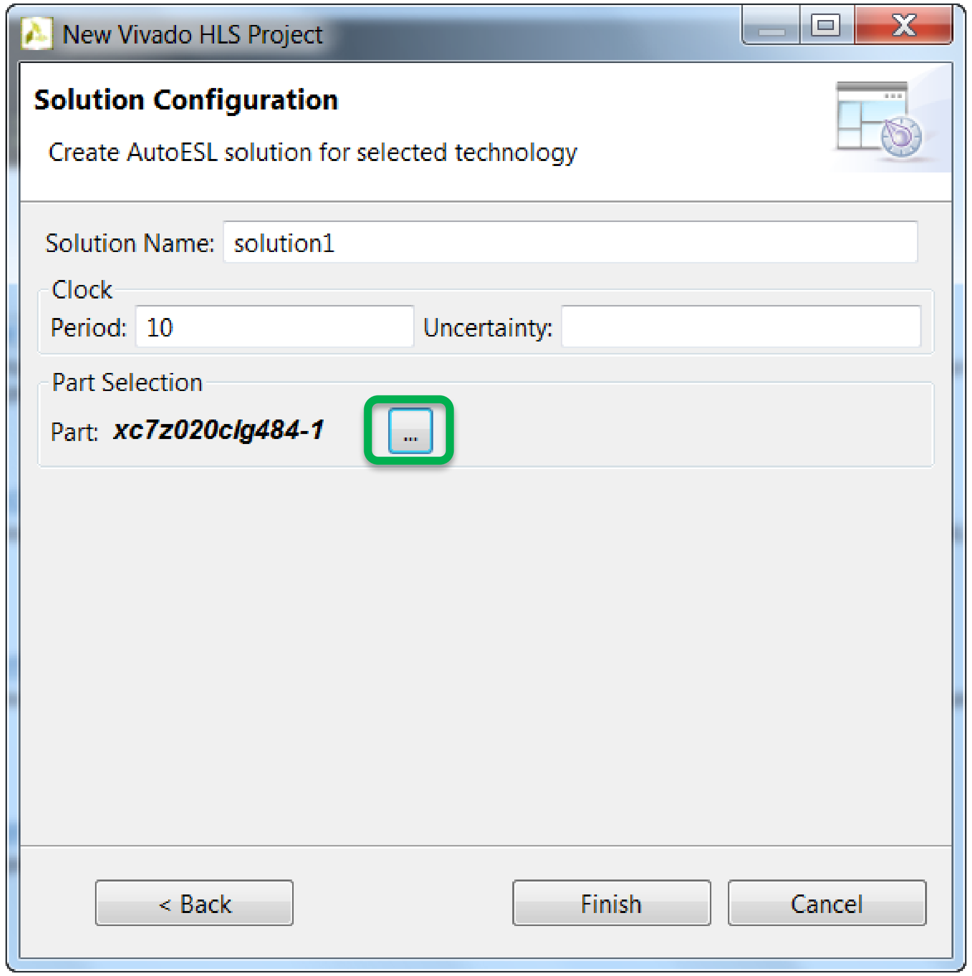
```
int main () {
  // Compare results
  int ret = system("diff --brief -w test_data/output.dat test_data/output.golden.dat");
  if (ret != 0) {
        printf("Test failed !!!\n", ret); return 1;
  } else {
        printf("Test passed !\n", ret); return 0;
  }
```

The –w option ensures the “newline” does not cause a difference between Windows and Linux files

  - ⇒ Non-synthesizable constructs may be added to a synthesize function if __SYNTHESIS__ is used

Solution Configuration

- ❐ Provide a solution name
  - ⇒ Default is solution1, then solution2 etc.
- ❐ Specify the clock
  - ⇒ The clock uncertainty is subtracted from the clock to provide an "effective clock period"
  - ⇒ Vitis HLS uses the "effective clock period" for Synthesis
  - ⇒ Provides users defined margin for downstream RTL synthesis, P&R
- ❐ Select the part or board
  - – For ELEC 522 select Zedboard


New Vivado HLS Project

Solution Configuration

Create AutoESL solution for selected technology

Solution Name: solution1

Clock

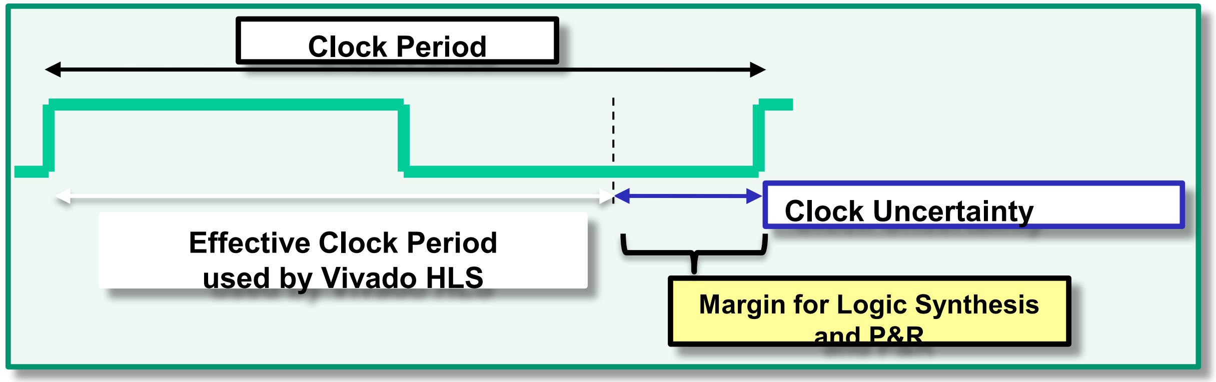
Period: 10 Uncertainty:

Part Selection

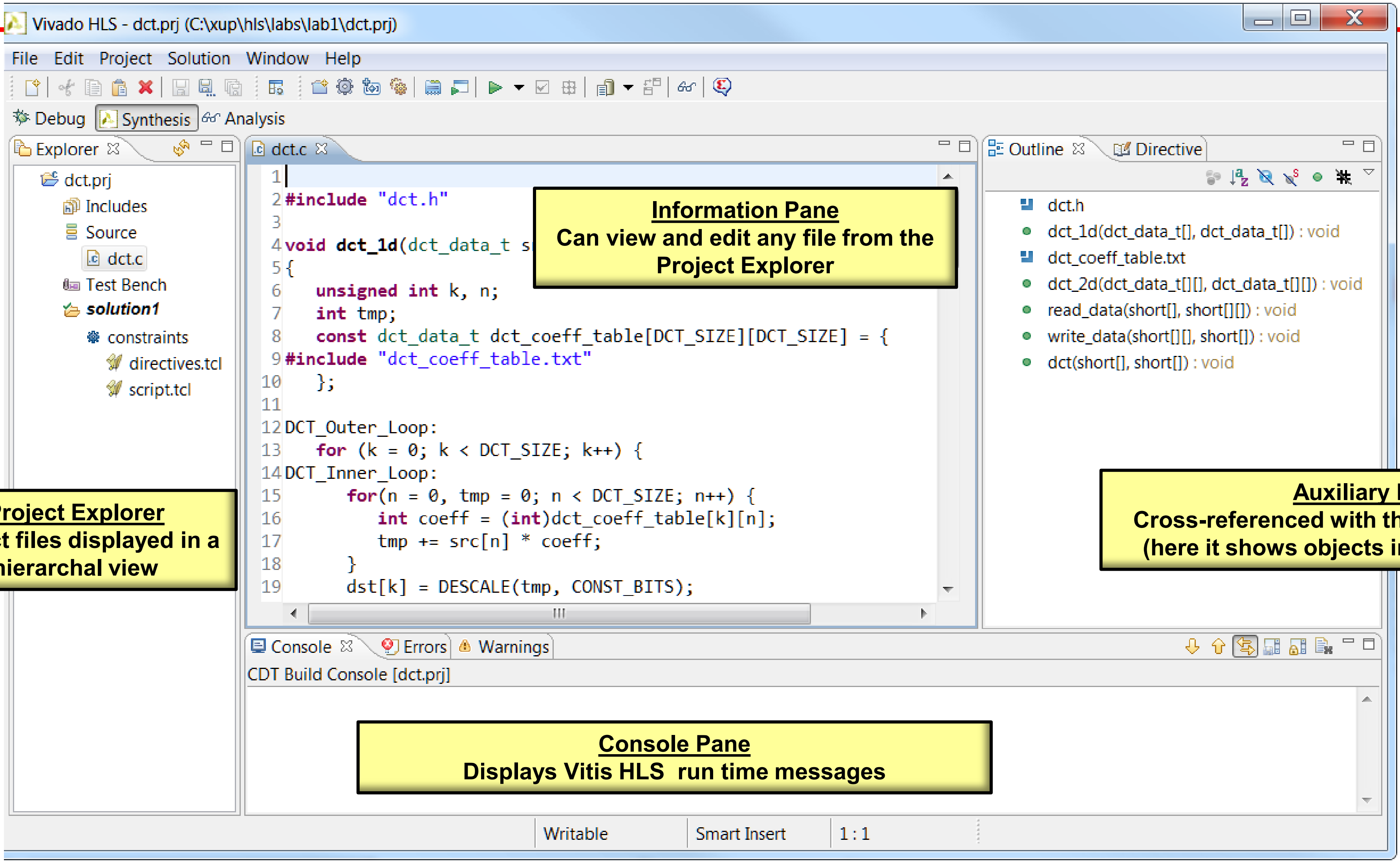
Part: ***xc7z020clg484-1***

Clock Specification

- Clock frequency must be specified
  - ⇒ Only 1 clock can be specified for C/C++ functions
  - ⇒ SystemC can define multiple clocks
- Clock uncertainty can be specified
  - ⇒ Subtracted from the clock period to give an effective clock period
  - ⇒ The effective clock period is used for synthesis
    - Should not be used as a design parameter
    - Do not vary for different results: this is your safety margin
  - ⇒ A user controllable margin to account for downstream RTL synthesis and P&R


Clock Period
Clock Uncertainty
Effective Clock Period used by Vivado HLS
Margin for Logic Synthesis and P&R

A Vitis HLS Project



```
#include "dct.h"

void dct_1d(dct_data_t s
{
   unsigned int k, n;
   int tmp;
   const dct_data_t dct_coeff_table[DCT_SIZE][DCT_SIZE] = {
#include "dct_coeff_table.txt"
   };

DCT_Outer_Loop:
   for (k = 0; k < DCT_SIZE; k++) {
DCT_Inner_Loop:
      for(n = 0, tmp = 0; n < DCT_SIZE; n++) {
         int coeff = (int)dct_coeff_table[k][n];
         tmp += src[n] * coeff;
      }
      dst[k] = DESCALE(tmp, CONST_BITS);
```

```
dct.h
dct_1d(dct_data_t[], dct_data_t[]) : void
dct_coeff_table.txt
dct_2d(dct_data_t[][], dct_data_t[][]) : void
read_data(short[], short[][]) : void
write_data(short[][], short[]) : void
dct(short[], short[]) : void
```

CDT Build Console [dct.prj]

Information Pane
Can view and edit any file from the Project Explorer

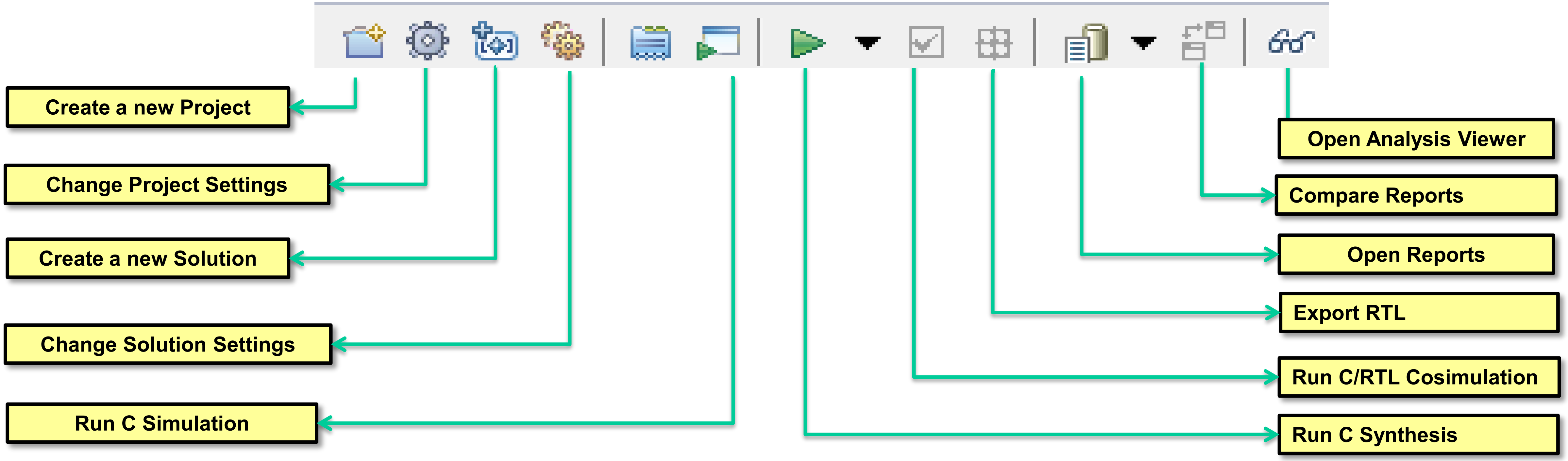
Auxiliary Pane
Cross-referenced with the Information Pane (here it shows objects in the source code)

Project Explorer
Project files displayed in a hierarchal view

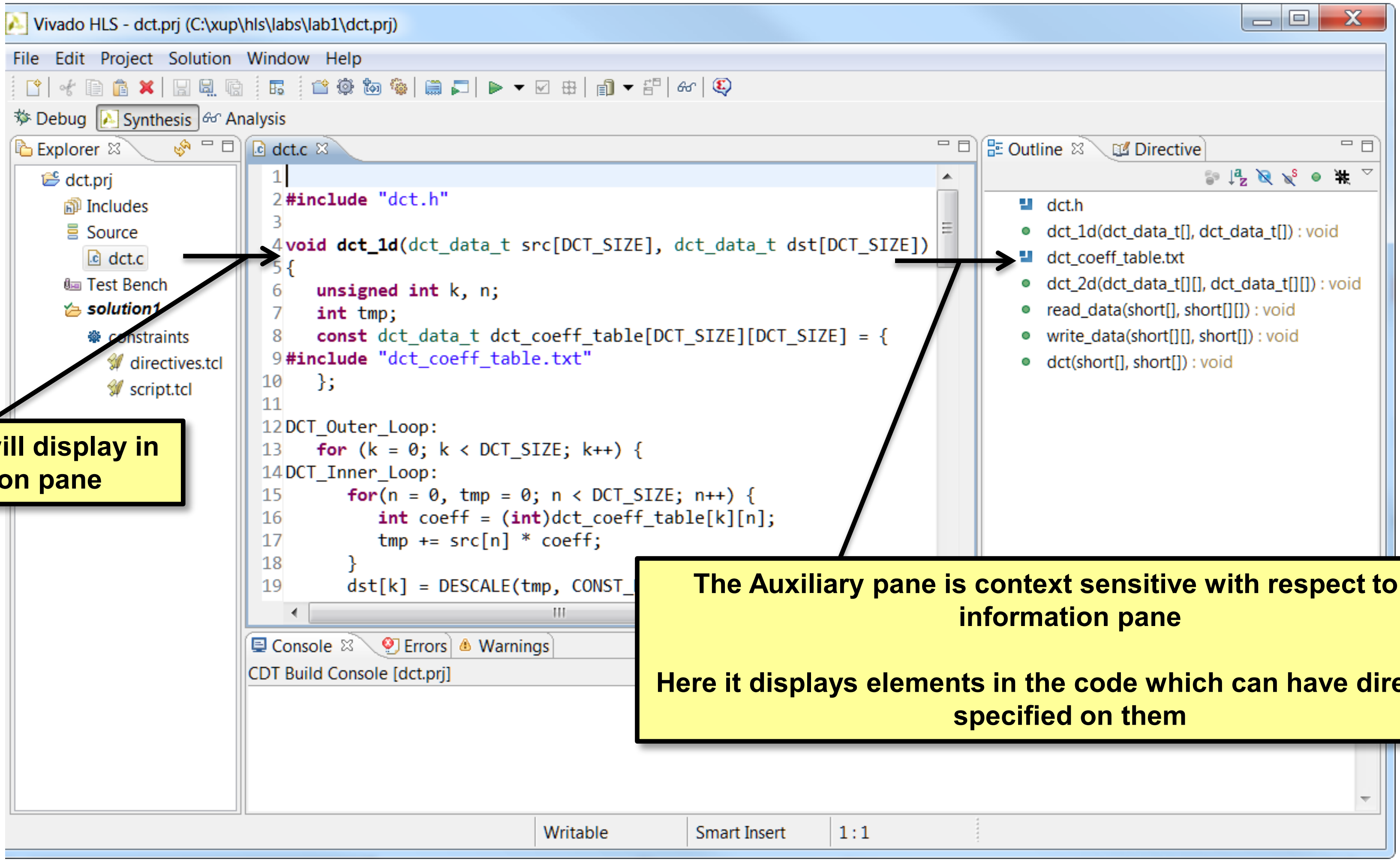
Console Pane
Displays Vitis HLS run time messages

Vitis HLS GUI Toolbar

- The primary commands have toolbar buttons
  - ⇒ Easy access for standard tasks
  - ⇒ Button highlights when the option is available
    - E.g. cannot perform C/RTL simulation before synthesis



Files: Views, Edits & Information



```c

#include "dct.h"

void dct_1d(dct_data_t src[DCT_SIZE], dct_data_t dst[DCT_SIZE])
{
   unsigned int k, n;
   int tmp;
   const dct_data_t dct_coeff_table[DCT_SIZE][DCT_SIZE] = {
#include "dct_coeff_table.txt"
   };

DCT_Outer_Loop:
   for (k = 0; k < DCT_SIZE; k++) {
DCT_Inner_Loop:
      for(n = 0, tmp = 0; n < DCT_SIZE; n++) {
         int coeff = (int)dct_coeff_table[k][n];
         tmp += src[n] * coeff;
      }
      dst[k] = DESCALE(tmp, CONST_
```

dct.h
dct_1d(dct_data_t[], dct_data_t[]) : void
dct_coeff_table.txt
dct_2d(dct_data_t[][], dct_data_t[][]) : void
read_data(short[], short[][]) : void
write_data(short[][], short[]) : void
dct(short[], short[]) : void

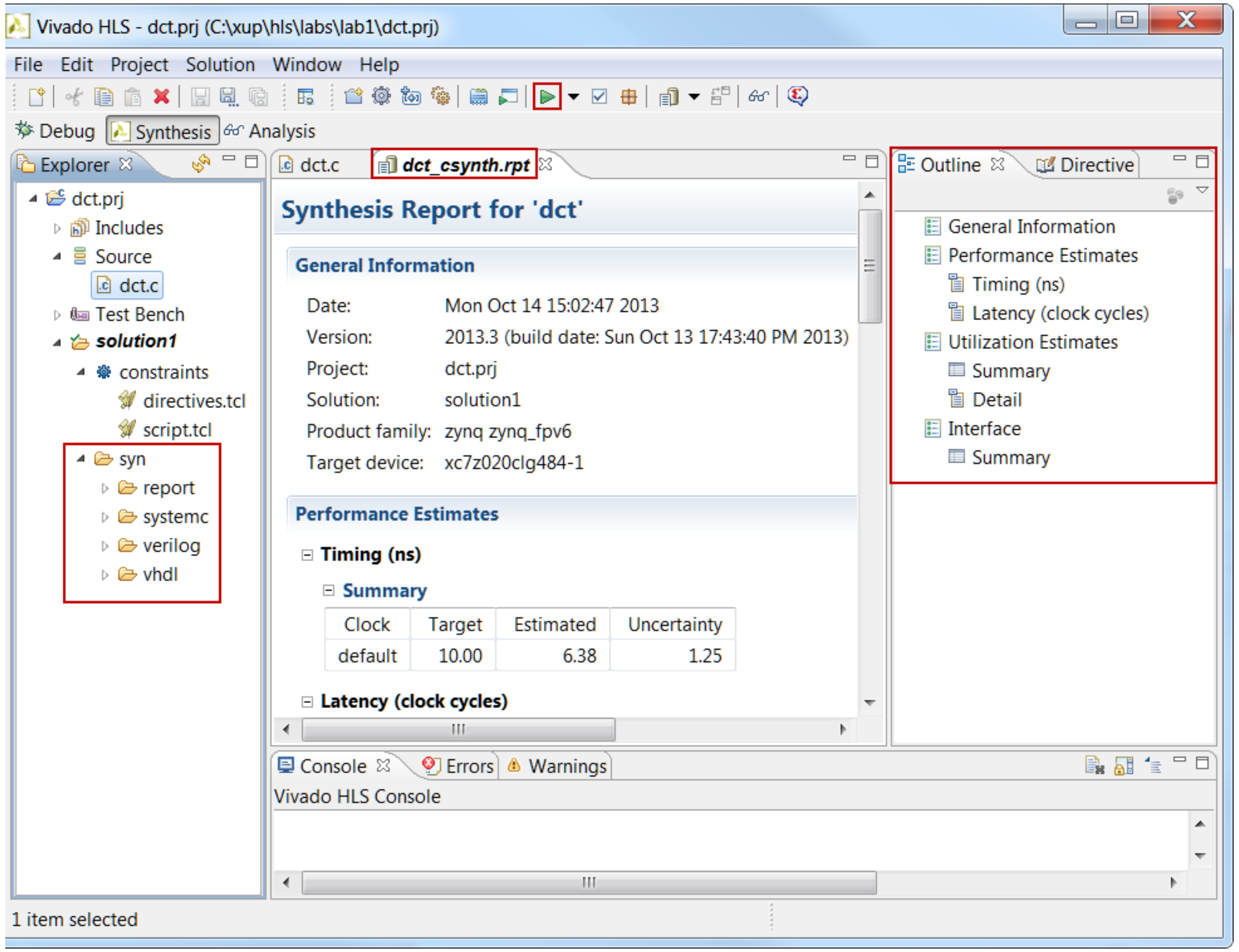
Open file and it will display in the information pane

The Auxiliary pane is context sensitive with respect to the information pane

Here it displays elements in the code which can have directives specified on them

Synthesis

- Run C Synthesis
- Console
  - ⇒ Will show run time information
  - ⇒ Examine for failed constraints
- A “syn” directory is created
  - ⇒ Verilog, VHDL & SystemC RTL
  - ⇒ Synthesis reports for all non-inlined functions
- Report opens automatically
  - ⇒ When synthesis completes
- Report is outlined in the Auxiliary pane


Vivado HLS - dct.prj (C:\xup\hls\labs\lab1\dct.prj)

File Edit Project Solution Window Help

Debug Synthesis Analysis

Explorer: dct.prj, Includes, Source, dct.c, Test Bench, solution1, constraints, directives.tcl, script.tcl, syn, report, systemc, verilog, vhdl

dct.c | dct_csynth.rpt

Synthesis Report for 'dct'

General Information

Date:	Mon Oct 14 15:02:47 2013
Version:	2013.3 (build date: Sun Oct 13 17:43:40 PM 2013)
Project:	dct.prj
Solution:	solution1
Product family:	zynq zynq_fpv6
Target device:	xc7z020clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
default	10.00	6.38	1.25

Latency (clock cycles)

Outline | Directive

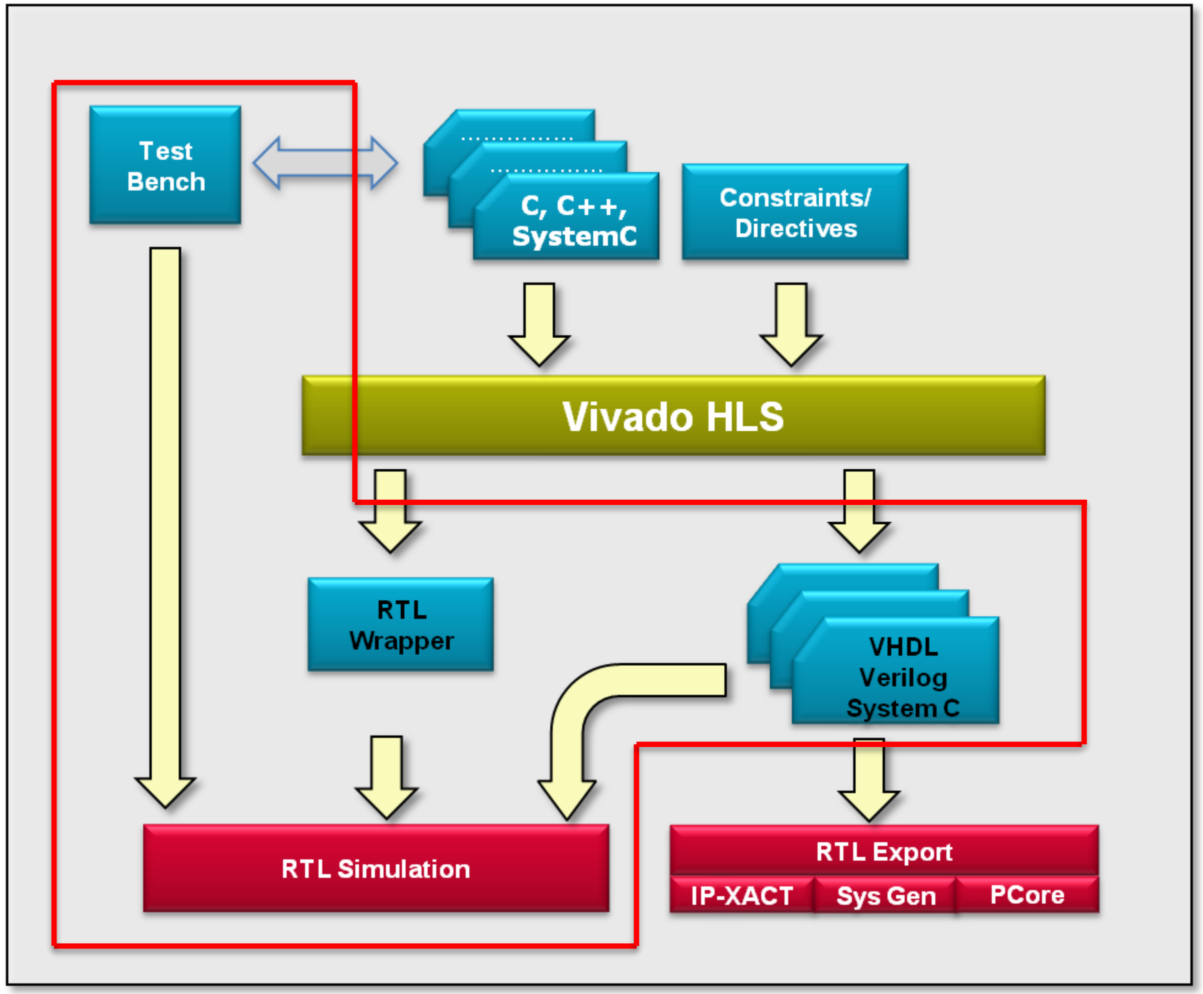
- General Information
- Performance Estimates
  - Timing (ns)
  - Latency (clock cycles)
- Utilization Estimates
  - Summary
  - Detail
- Interface
  - Summary

Console | Errors | Warnings

Vivado HLS Console

1 item selected

Vitis HLS : RTL Verification



RTL output in Verilog, VHDL and SystemC

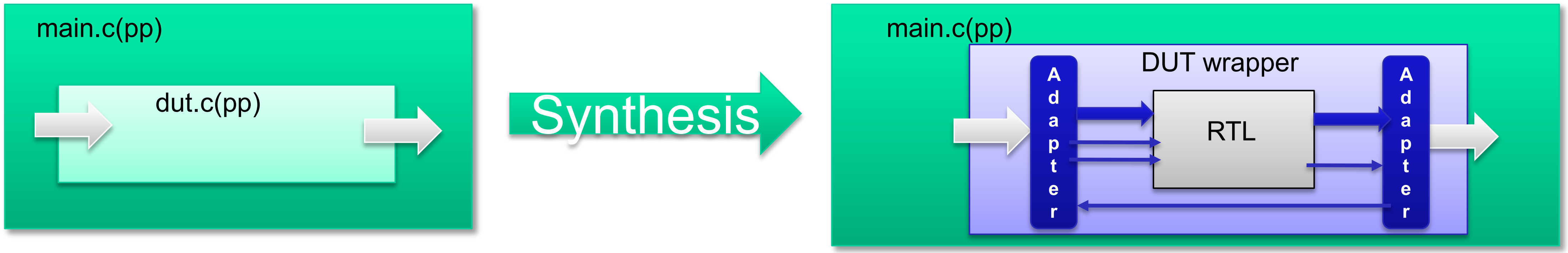
Automatic re-use of the C-level test bench

RTL verification can be executed from within Vitis HLS

Support for Xilinx simulators (XSim and ISim) and 3rd party HDL simulators in automated flow

RTL Verification: Under-the-Hood

- RTL Co-Simulation
  - ⇒ Vitis HLS provides RTL verification
  - ⇒ Creates the wrappers and adapters to re-use the C test bench



- Prior to synthesis
  - Test bench
  - Top-level C function

- After synthesis
  - Test bench
  - SystemC wrapper created by Vivado HLS
  - SystemC adapters created by Vivado HLS
  - RTL output from Vitis HLS
    - SystemC, Verilog or VHDL

There is no HDL test bench created

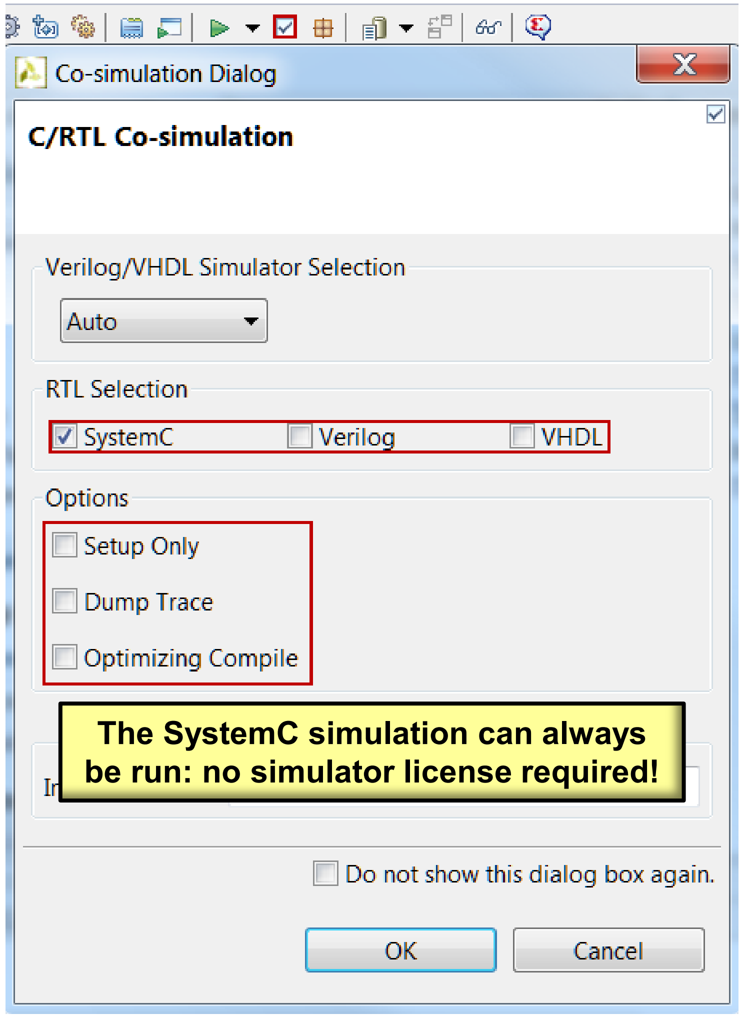
RTL Verification Support

- Vitis HLS RTL Output
  - ⇒ Vitis HLS outputs RTL in SystemC, Verilog and VHDL
    - The SystemC output is at the RT Level
    - The input is not transformed to SystemC at the ESL
- RTL Verification with SystemC
  - ⇒ The SystemC RTL output can be used to verify the design without the need for a HDL simulator and license
- HDL Simulation Support
  - ⇒ Vitis HLS supports HDL simulators on both Windows & Linux
  - ⇒ The 3rd party simulator executable must be in OS search path

Simulator	Linux	Windows
XSim (Vivado Simulator)	Supported	Supported
ISim (ISE Simulator)	Supported	Supported
Mentor Graphics ModelSim	Supported	Supported
Synopsys VCS	Supported	Not Available
NCSim	Supported	Not Available
Riviera	Supported	Supported

C/RTL Co-simulation

- Start Simulation
  - ⇒ Opens the dialog box
- Select the RTL
  - ⇒ SystemC does not require a 3rd party license
  - ⇒ Verilog and VHDL require the appropriate simulator
    - Select the desired simulator
  - ⇒ Run any or all
- Options
  - ⇒ Can output trace file (VCD format)
  - ⇒ Optimize the C compilation & specify test bench linker flags
  - ⇒ The "setup only" option will not execute the simulation
- OK will run the simulator
  - ⇒ Output files will be created in a "sim" directory


Co-simulation Dialog

C/RTL Co-simulation

Verilog/VHDL Simulator Selection: Auto

RTL Selection: [✓ SystemC] Verilog VHDL

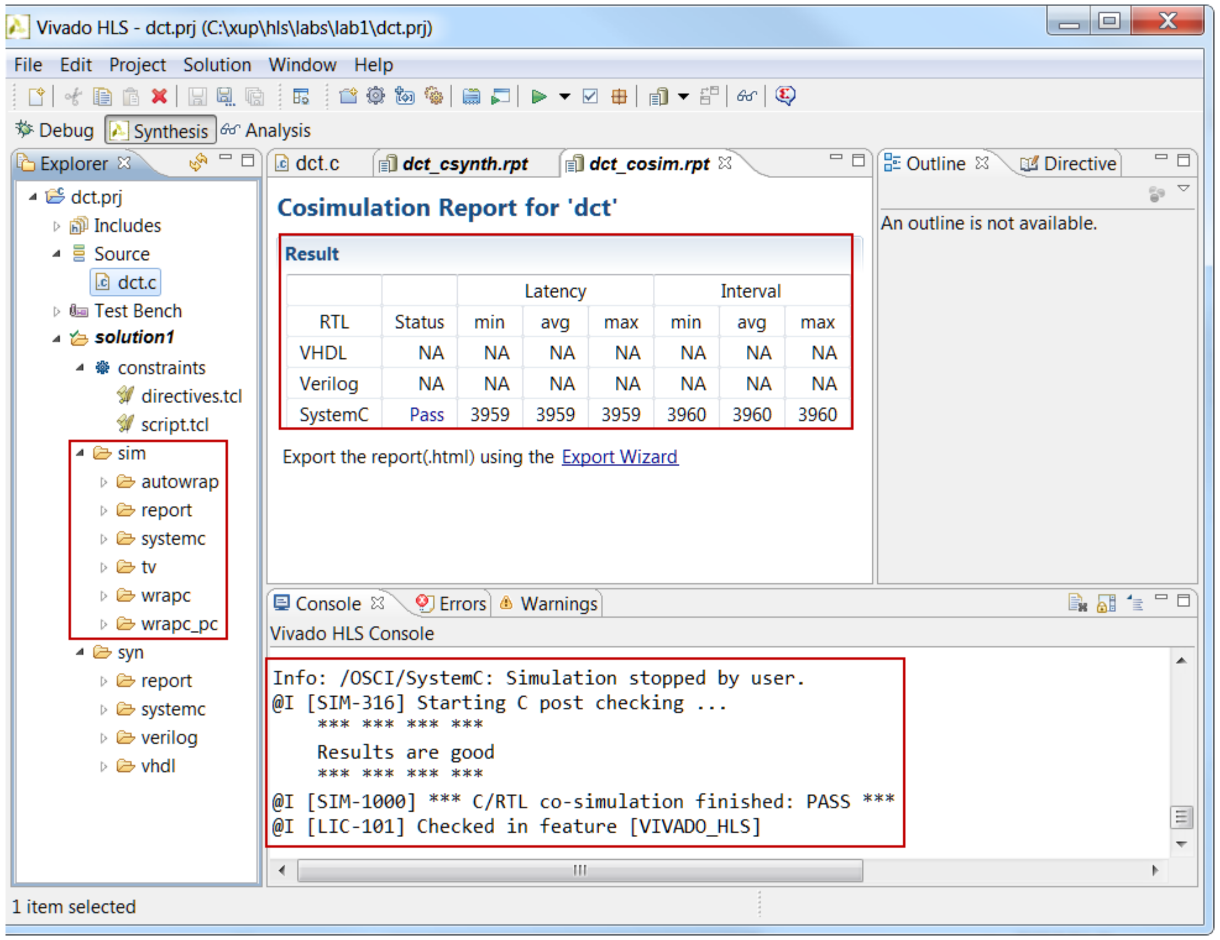
Options: Setup Only, Dump Trace, Optimizing Compile

The SystemC simulation can always be run: no simulator license required!

Do not show this dialog box again.

Simulation Results

- ❒ Simulation output is shown in the console
- ❒ Expect the same test bench response
  - ⇒ If the C test bench plots, it will with the RTL design (but slower)
- ❒ Sim Directory
  - ⇒ Will contain a sub-directory for each RTL
    which is verified
- ❒ Report
  - ⇒ A report is created and opened automatically



Cosimulation Report for 'dct'

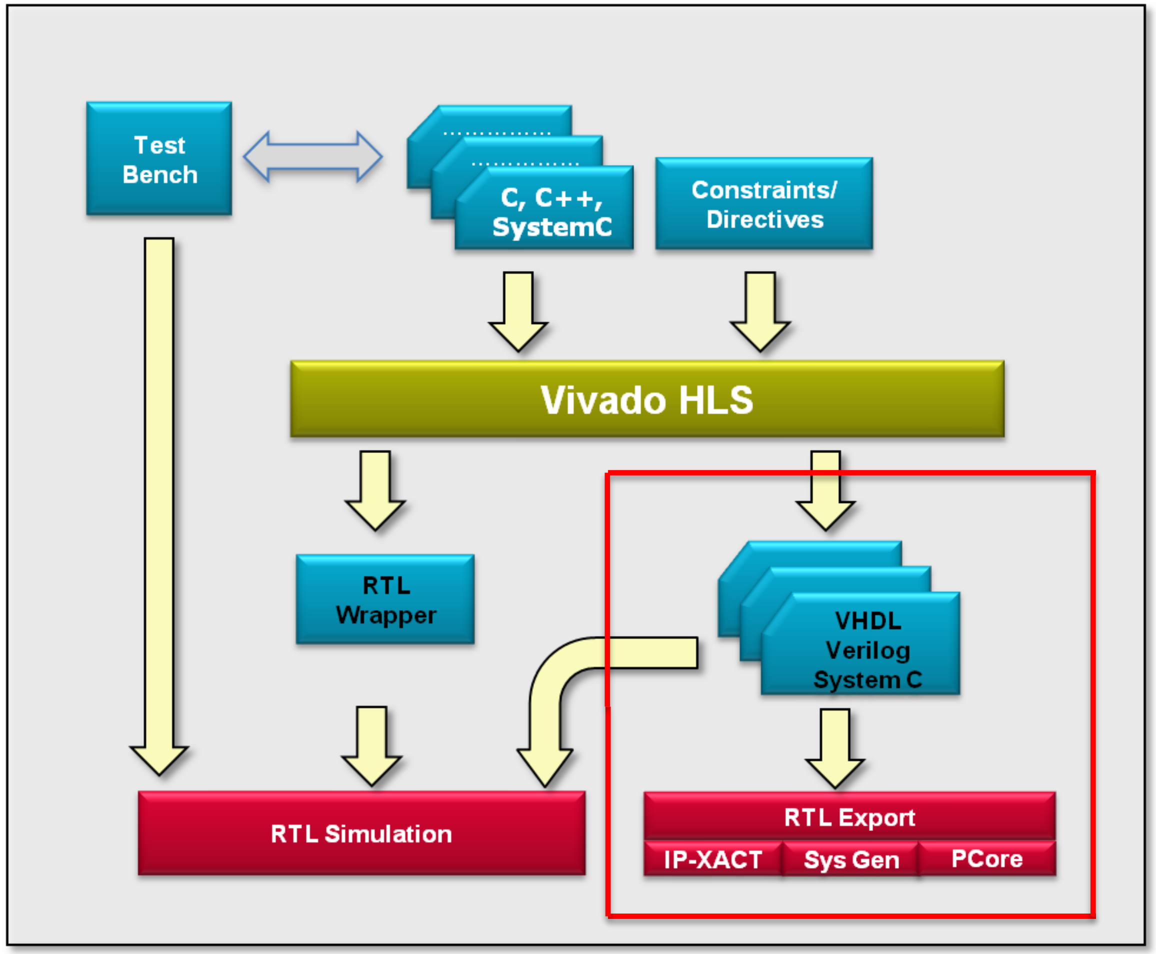
Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	NA	NA	NA	NA	NA	NA	NA
SystemC	Pass	3959	3959	3959	3960	3960	3960

Export the report(.html) using the Export Wizard

```
Info: /OSCI/SystemC: Simulation stopped by user.
@I [SIM-316] Starting C post checking ...
    *** *** *** ***
    Results are good
    *** *** *** ***
@I [SIM-1000] *** C/RTL co-simulation finished: PASS ***
@I [LIC-101] Checked in feature [VIVADO_HLS]
```

Vitis HLS : RTL Export



RTL output in Verilog, VHDL and SystemC

Scripts created for RTL synthesis tools

RTL Export to IP-Catalog, SysGen, and Pcore formats

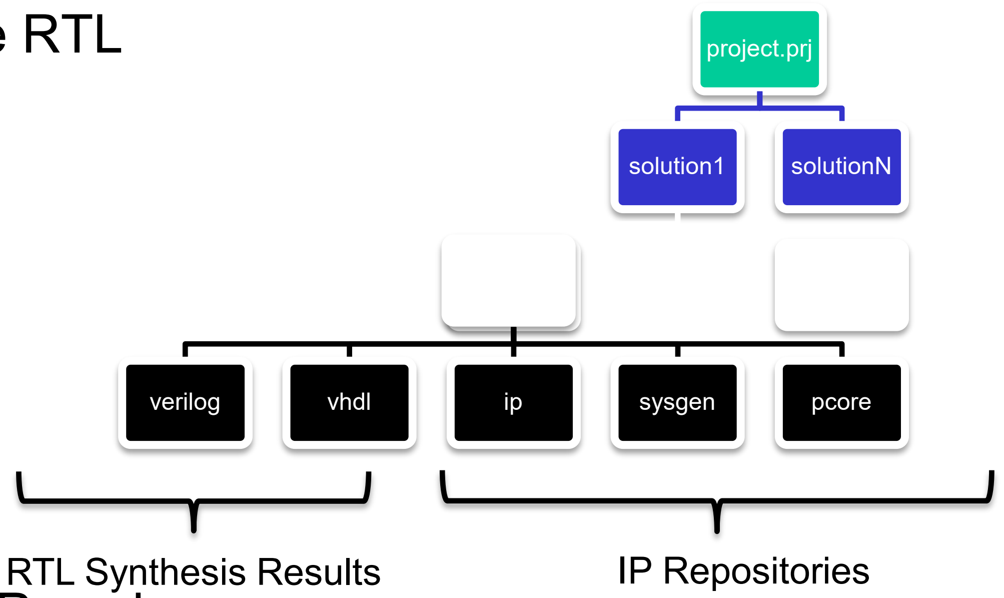
IP-Catalog and SysGen => Vitis HLS for 7 Series and Zynq families

RTL Export Support

- RTL Export
  - ⇒ Can be exported to one of the three types
    - IP-Catalog formatted IP for use with Vivado System Edition
      - 7 Series and Zynq families only; supported by HLS license
    - A System Generator IP block
      - 7 Series and Zynq families only; supported by HLS license
- Generation in both Verilog and VHDL for non-bus or non-interface based designs
- Logic synthesis will automatically be performed
  - ⇒ HLS license will use Vivado RTL Synthesis

RTL Export: Synthesis

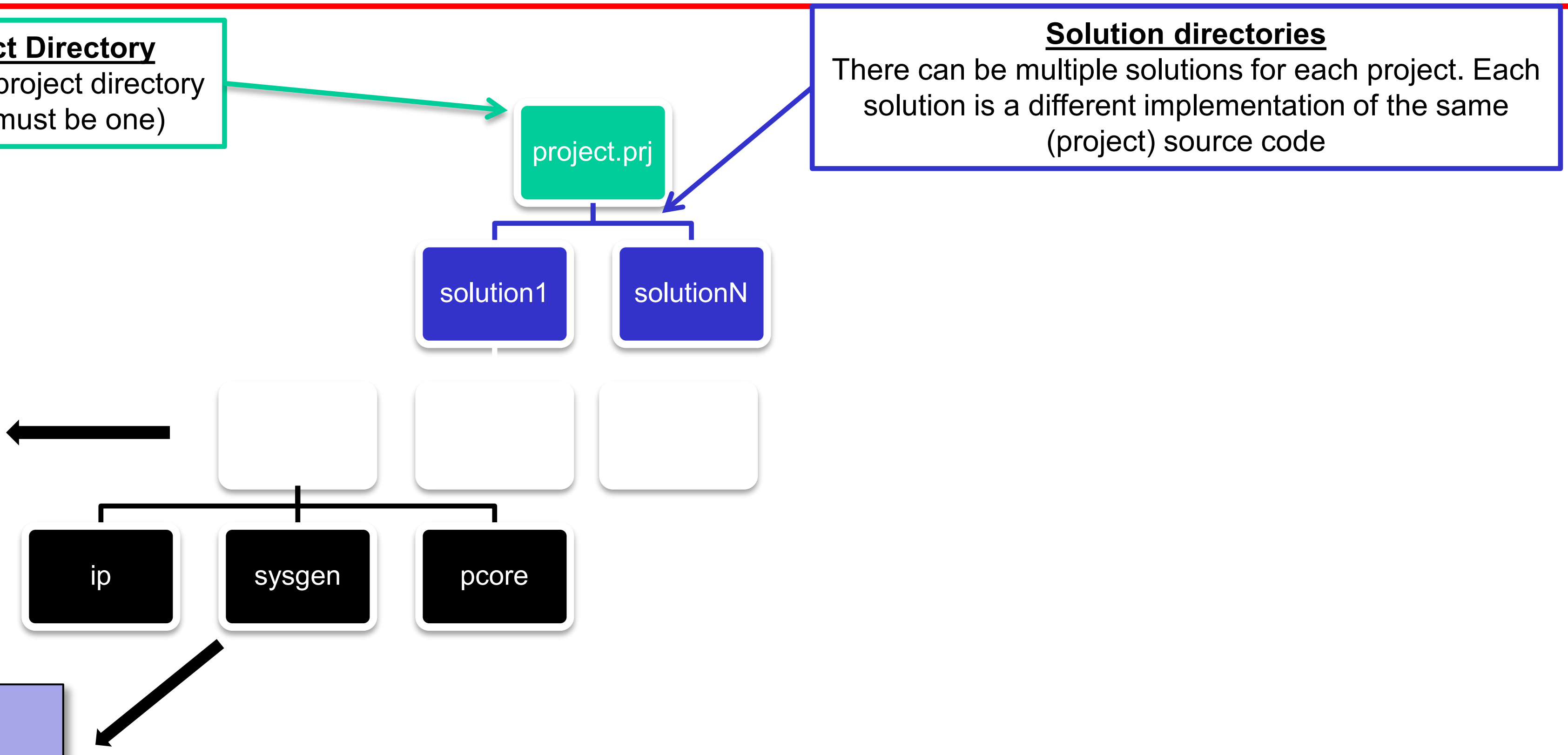
- RTL Synthesis can be performed to evaluate the RTL
  - ⇒ IP-Catalog and System Generator formats:
  - ⇒ Vivado synthesis performed



- RTL synthesis results are not included with the IP package
  - ⇒ Evaluate step is provided to give confidence
    - Timing will be as estimate (or better)
    - Area will be as estimated (or better)
  - ⇒ Final RTL IP is synthesized with the rest of the RTL design
    - RTL Synthesis results from the Vitis HLS evaluation are not used

RTL Export: IP Repositories

- IP can be imported into other Xilinx tools



Project Directory
Top-level project directory (there must be one)

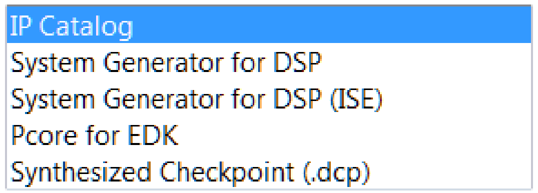
Solution directories
There can be multiple solutions for each project. Each solution is a different implementation of the same (project) source code

In Vivado :
1. Project Manager > IP Catalog
2. Add IP to import this block
3. Browse to the zip file inside "ip"

In System Generator :
1. Use XilinxBlockAdd
2. Select Vitis_HLS block type
3. Browse to the solution directory

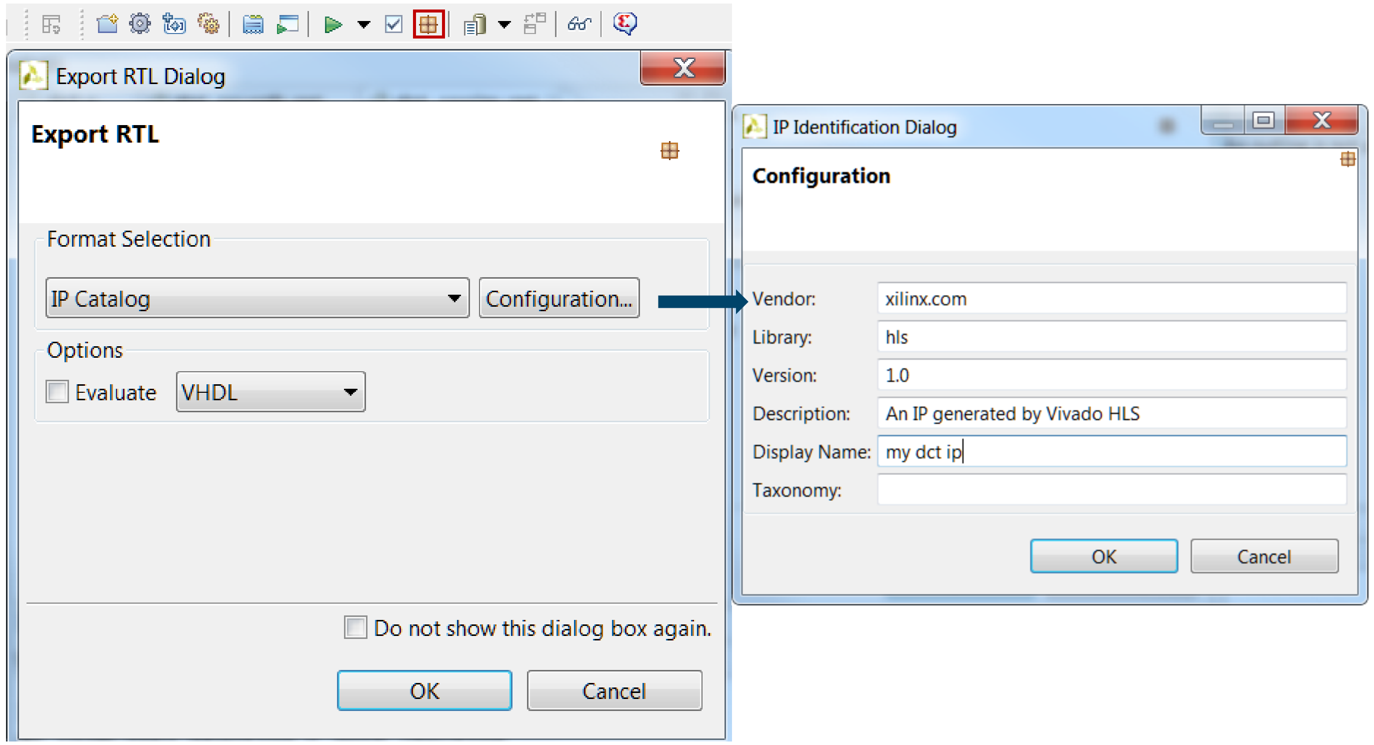
RTL Export for Implementation

- Click on Export RTL
  - ⇒ Export RTL Dialog opens
- Select the desired output format



IP Catalog
System Generator for DSP
System Generator for DSP (ISE)
Pcore for EDK
Synthesized Checkpoint (.dcp)

- Optionally, configure the output
- Select the desired language
- Optionally, click on Evaluate button for invoking implementation tools from within Vitis HLS
- Click OK to start the implementation



Export RTL Dialog

Export RTL

Format Selection

IP Catalog — Configuration...

Options

☐ Evaluate — VHDL

☐ Do not show this dialog box again.

OK — Cancel

IP Identification Dialog

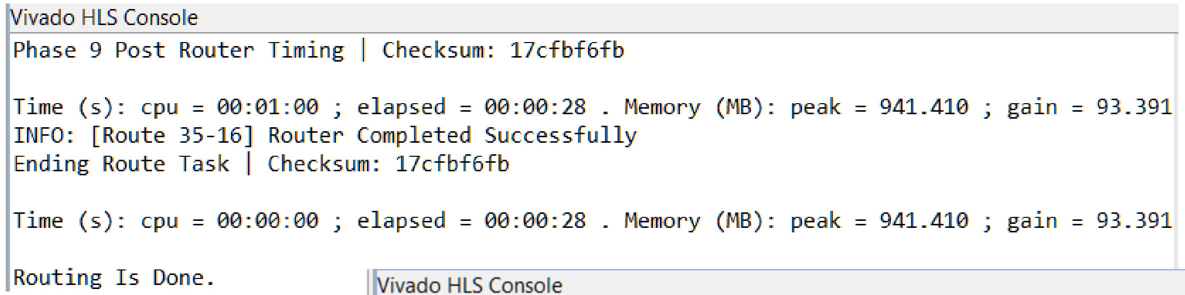
Configuration

Vendor:	xilinx.com
Library:	hls
Version:	1.0
Description:	An IP generated by Vivado HLS
Display Name:	my dct ip
Taxonomy:	

OK — Cancel

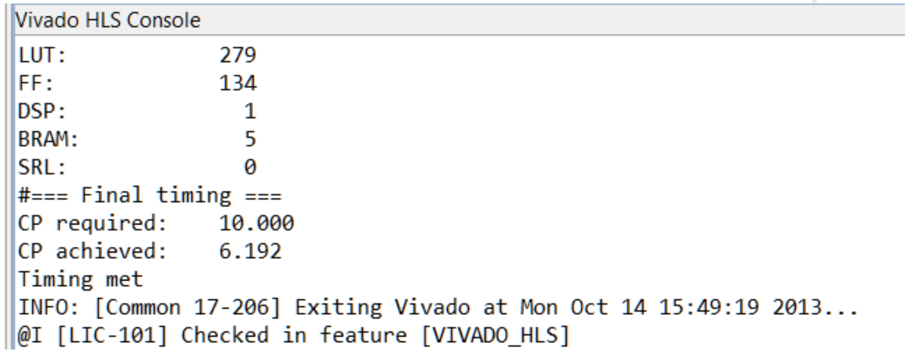
RTL Export (Evaluate Option) Results

- Impl directory created
  - ⇒ Will contain a sub-directory for each RTL which is synthesized
- Report
  - ⇒ A report is created and opened automatically



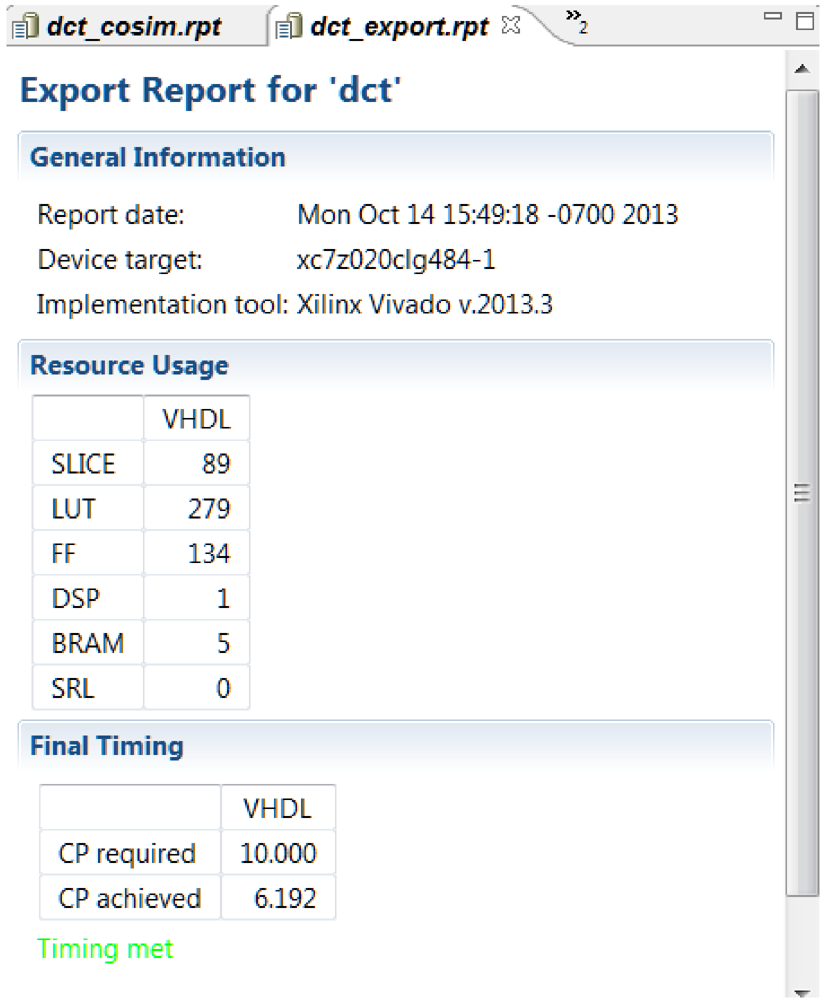
Vivado HLS Console

```
Phase 9 Post Router Timing | Checksum: 17cfbf6fb

Time (s): cpu = 00:01:00 ; elapsed = 00:00:28 . Memory (MB): peak = 941.410 ; gain = 93.391
INFO: [Route 35-16] Router Completed Successfully
Ending Route Task | Checksum: 17cfbf6fb

Time (s): cpu = 00:00:00 ; elapsed = 00:00:28 . Memory (MB): peak = 941.410 ; gain = 93.391

Routing Is Done.
```



Vivado HLS Console

```
LUT:            279
FF:             134
DSP:              1
BRAM:             5
SRL:              0
#=== Final timing ===
CP required:    10.000
CP achieved:    6.192
Timing met
INFO: [Common 17-206] Exiting Vivado at Mon Oct 14 15:49:19 2013...
@I [LIC-101] Checked in feature [VIVADO_HLS]
```



Export Report for 'dct'

General Information

Report date:	Mon Oct 14 15:49:18 -0700 2013
Device target:	xc7z020clg484-1
Implementation tool:	Xilinx Vivado v.2013.3

Resource Usage

	VHDL
SLICE	89
LUT	279
FF	134
DSP	1
BRAM	5
SRL	0

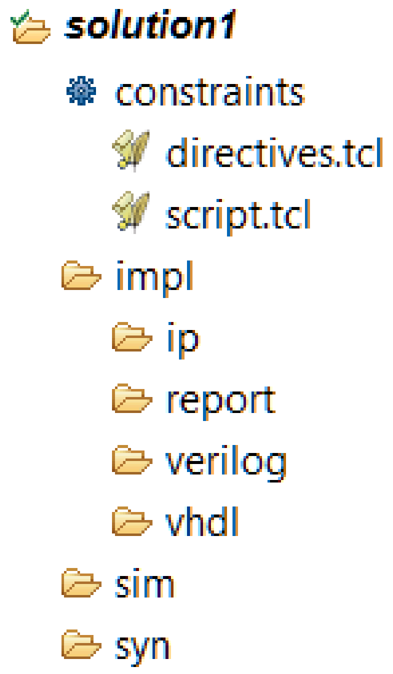
Final Timing

	VHDL
CP required	10.000
CP achieved	6.192

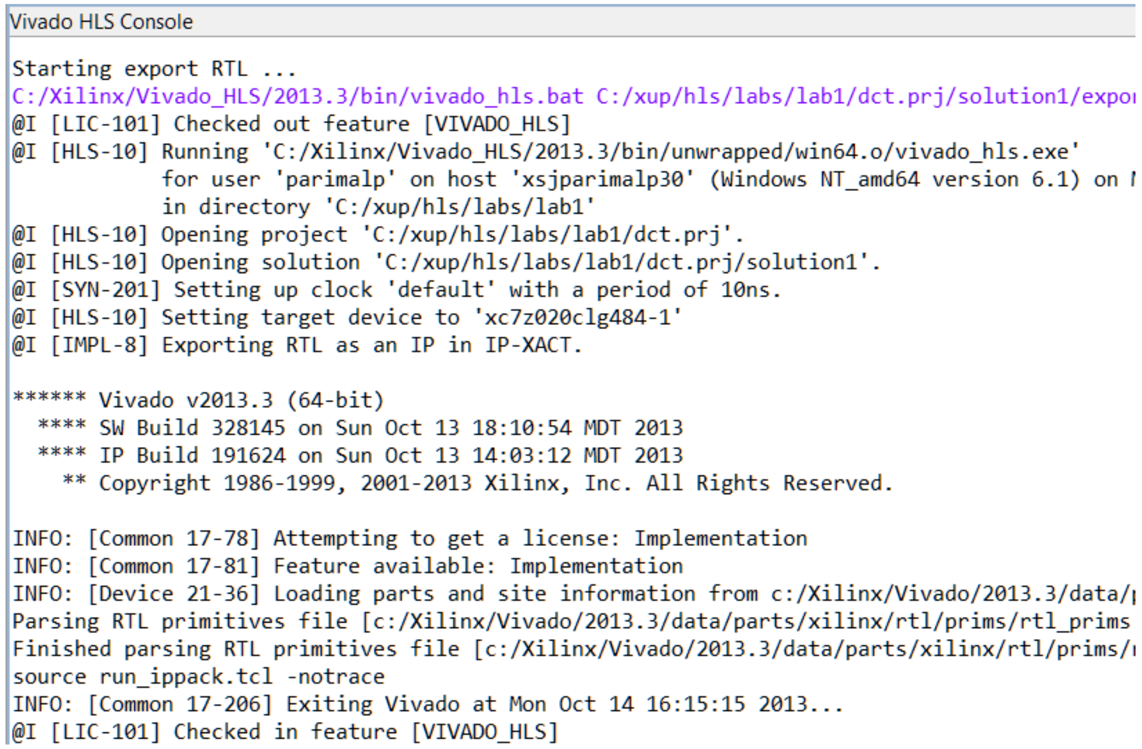
Timing met

RTL Export Results (Evaluate Option Unchecked)

- Impl directory created
  - ⇒ Will contain a sub-directory for both VHDL and Verilog along with the ip directory
- No report will be created
- Observe the console
  - ⇒ No packing, routing phases



```
solution1
    constraints
        directives.tcl
        script.tcl
    impl
        ip
        report
        verilog
        vhdl
    sim
    syn
```

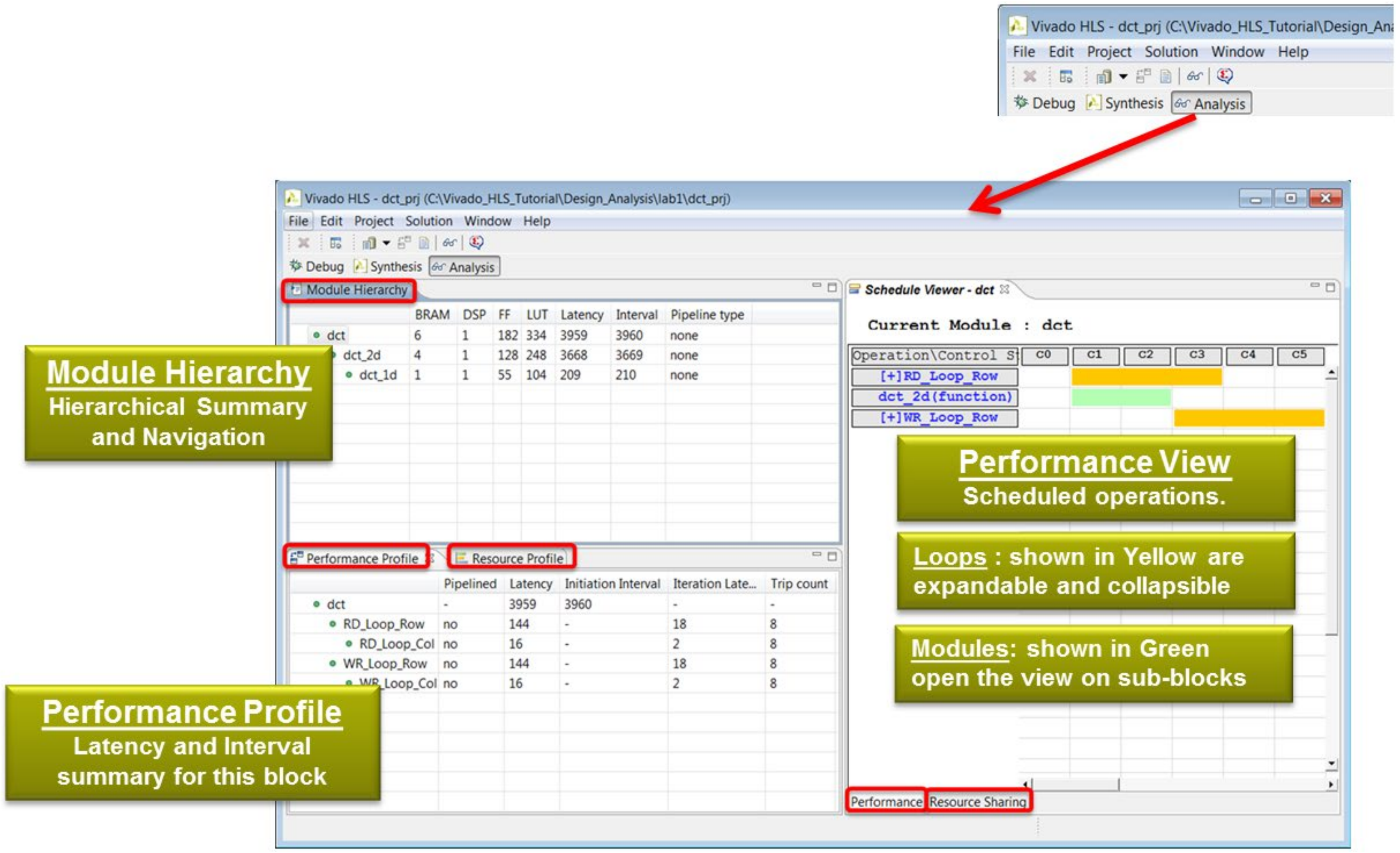


Vivado HLS Console

```
Starting export RTL ...
C:/Xilinx/Vivado_HLS/2013.3/bin/vivado_hls.bat C:/xup/hls/labs/lab1/dct.prj/solution1/expo
@I [LIC-101] Checked out feature [VIVADO_HLS]
@I [HLS-10] Running 'C:/Xilinx/Vivado_HLS/2013.3/bin/unwrapped/win64.o/vivado_hls.exe'
            for user 'parimalp' on host 'xsjparimalp30' (Windows NT_amd64 version 6.1) on
            in directory 'C:/xup/hls/labs/lab1'
@I [HLS-10] Opening project 'C:/xup/hls/labs/lab1/dct.prj'.
@I [HLS-10] Opening solution 'C:/xup/hls/labs/lab1/dct.prj/solution1'.
@I [SYN-201] Setting up clock 'default' with a period of 10ns.
@I [HLS-10] Setting target device to 'xc7z020clg484-1'
@I [IMPL-8] Exporting RTL as an IP in IP-XACT.

****** Vivado v2013.3 (64-bit)
  **** SW Build 328145 on Sun Oct 13 18:10:54 MDT 2013
  **** IP Build 191624 on Sun Oct 13 14:03:12 MDT 2013
    ** Copyright 1986-1999, 2001-2013 Xilinx, Inc. All Rights Reserved.

INFO: [Common 17-78] Attempting to get a license: Implementation
INFO: [Common 17-81] Feature available: Implementation
INFO: [Device 21-36] Loading parts and site information from c:/Xilinx/Vivado/2013.3/data/
Parsing RTL primitives file [c:/Xilinx/Vivado/2013.3/data/parts/xilinx/rtl/prims/rtl_prims
Finished parsing RTL primitives file [c:/Xilinx/Vivado/2013.3/data/parts/xilinx/rtl/prims/
source run_ippack.tcl -notrace
INFO: [Common 17-206] Exiting Vivado at Mon Oct 14 16:15:15 2013...
@I [LIC-101] Checked in feature [VIVADO_HLS]
```

Analysis Perspective

- Perspective for design analysis
  - ⇒ Allows interactive analysis



Vivado HLS - dct_prj (C:\Vivado_HLS_Tutorial\Design_Ana
File Edit Project Solution Window Help
Debug Synthesis Analysis
Vivado HLS - dct_prj (C:\Vivado_HLS_Tutorial\Design_Analysis\lab1\dct_prj)
Module Hierarchy
Schedule Viewer - dct
Performance Profile
Resource Profile
Performance
Resource Sharing

Module Hierarchy
Hierarchical Summary and Navigation

	BRAM	DSP	FF	LUT	Latency	Interval	Pipeline type
dct	6	1	182	334	3959	3960	none
dct_2d	4	1	128	248	3668	3669	none
dct_1d	1	1	55	104	209	210	none

Current Module : dct

Operation\Control St	C0	C1	C2	C3	C4	C5
[+]RD_Loop_Row		🟨	🟨	🟨		
dct_2d(function)		🟩	🟩			
[+]WR_Loop_Row				🟨	🟨	🟨

Performance View
Scheduled operations.

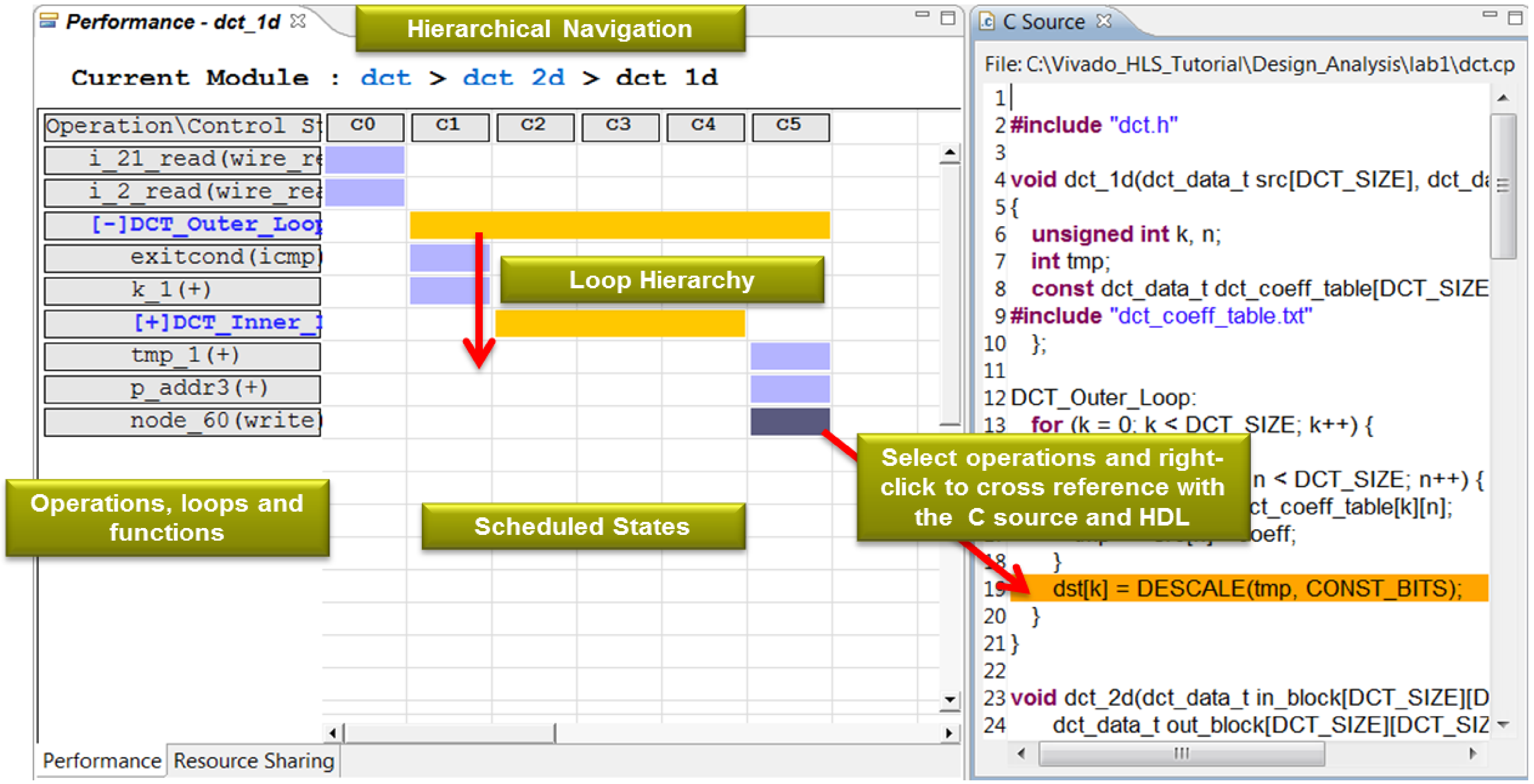
Loops : shown in Yellow are expandable and collapsible

Modules: shown in Green open the view on sub-blocks

	Pipelined	Latency	Initiation Interval	Iteration Late...	Trip count
dct	-	3959	3960	-	-
RD_Loop_Row	no	144	-	18	8
RD_Loop_Col	no	16	-	2	8
WR_Loop_Row	no	144	-	18	8
WR_Loop_Col	no	16	-	2	8

Performance Profile
Latency and Interval summary for this block

Performance Analysis


Performance - dct_1d
Hierarchical Navigation
Loop Hierarchy
Operations, loops and functions
Scheduled States
Select operations and right-click to cross reference with the C source and HDL
C Source
Performance
Resource Sharing

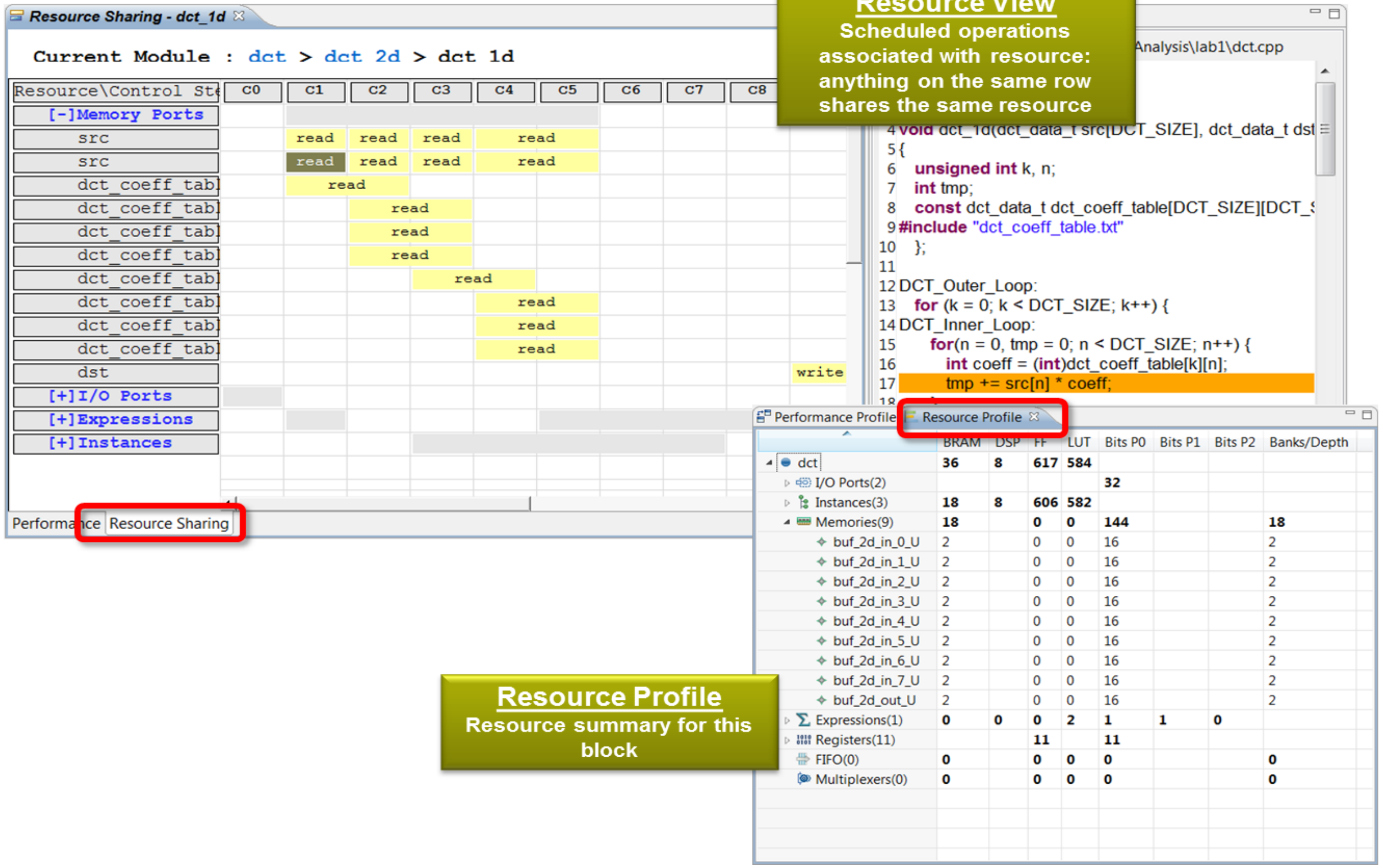
Current Module : dct > dct_2d > dct_1d

Operation\Control St	C0	C1	C2	C3	C4	C5
i_21_read(wire_re						
i_2_read(wire_rea						
[-]DCT_Outer_Loop						
exitcond(icmp)						
k_1(+)						
[+]DCT_Inner_L						
tmp_1(+)						
p_addr3(+)						
node_60(write)						

File: C:\Vivado_HLS_Tutorial\Design_Analysis\lab1\dct.cp

```

#include "dct.h"

void dct_1d(dct_data_t src[DCT_SIZE], dct_da
{
   unsigned int k, n;
   int tmp;
   const dct_data_t dct_coeff_table[DCT_SIZE
#include "dct_coeff_table.txt"
   };

DCT_Outer_Loop:
   for (k = 0; k < DCT_SIZE; k++) {
                                     n < DCT_SIZE; n++) {
                                     ct_coeff_table[k][n];
                                     oeff;
      }
      dst[k] = DESCALE(tmp, CONST_BITS);
   }
}

void dct_2d(dct_data_t in_block[DCT_SIZE][D
      dct_data_t out_block[DCT_SIZE][DCT_SIZ
```

Resource Analysis



Resource Sharing - dct_1d

Current Module : dct > dct 2d > dct 1d

Resource\Control Ste	C0	C1	C2	C3	C4	C5	C6	C7	C8	
[-]Memory Ports										
src		read	read	read	read					
src		read	read	read	read					
dct_coeff_tabl		read								
dct_coeff_tabl			read							
dct_coeff_tabl			read							
dct_coeff_tabl			read							
dct_coeff_tabl				read						
dct_coeff_tabl					read					
dct_coeff_tabl					read					
dct_coeff_tabl					read					
dst										write
[+]I/O Ports										
[+]Expressions										
[+]Instances										

Performance Resource Sharing

Resource View
Scheduled operations associated with resource: anything on the same row shares the same resource

Analysis\lab1\dct.cpp

```
void dct_1d(dct_data_t src[DCT_SIZE], dct_data_t dst
{
   unsigned int k, n;
   int tmp;
   const dct_data_t dct_coeff_table[DCT_SIZE][DCT_
#include "dct_coeff_table.txt"
   };

DCT_Outer_Loop:
   for (k = 0; k < DCT_SIZE; k++) {
DCT_Inner_Loop:
      for(n = 0, tmp = 0; n < DCT_SIZE; n++) {
         int coeff = (int)dct_coeff_table[k][n];
         tmp += src[n] * coeff;
```

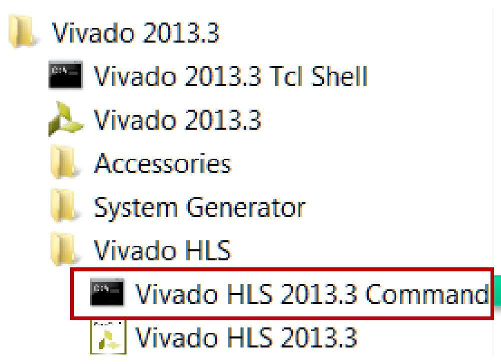
Performance Profile | Resource Profile

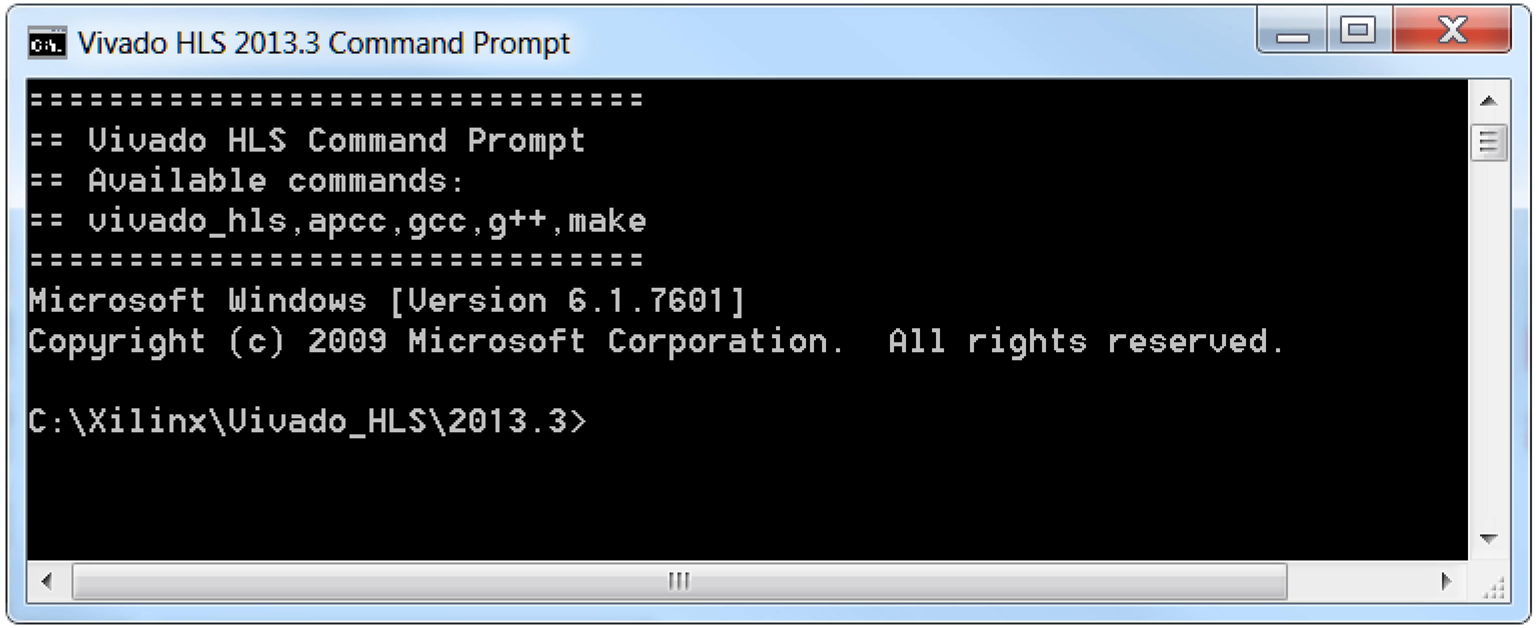
	BRAM	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth
dct	36	8	617	584				
I/O Ports(2)					32			
Instances(3)	18	8	606	582				
Memories(9)	18		0	0	144			18
buf_2d_in_0_U	2		0	0	16			2
buf_2d_in_1_U	2		0	0	16			2
buf_2d_in_2_U	2		0	0	16			2
buf_2d_in_3_U	2		0	0	16			2
buf_2d_in_4_U	2		0	0	16			2
buf_2d_in_5_U	2		0	0	16			2
buf_2d_in_6_U	2		0	0	16			2
buf_2d_in_7_U	2		0	0	16			2
buf_2d_out_U	2		0	0	16			2
Expressions(1)	0	0	0	2	1	1	0	
Registers(11)			11		11			
FIFO(0)	0		0	0	0			0
Multiplexers(0)	0		0	0	0			0

Resource Profile
Resource summary for this block

Command Line Interface: Batch Mode

- Vitis HLS can also be run in batch mode
  - ⇒ Opening the Command Line Interface (CLI) will give a shell



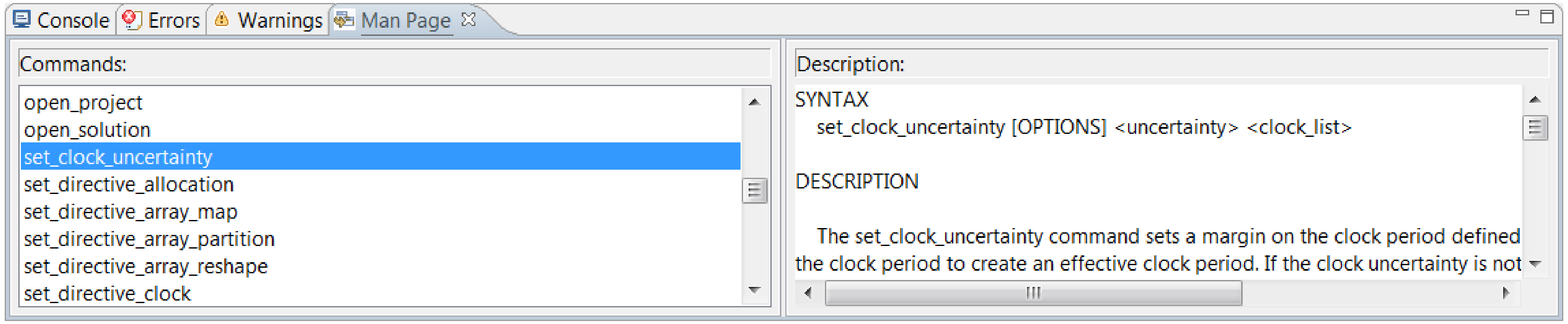


```
===============================
== Vivado HLS Command Prompt
== Available commands:
== vivado_hls,apcc,gcc,g++,make
===============================
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Xilinx\Vivado_HLS\2013.3>
```

  - ⇒ Supports the commands required to run Vitis HLS & pre-synthesis verification (gcc, g++, apcc, make)

Help

- Help is always available
  - ⇒ The Help Menu
  - ⇒ Opens User Guide, Reference Guide and Man Pages



Commands:

```
open_project
open_solution
set_clock_uncertainty
set_directive_allocation
set_directive_array_map
set_directive_array_partition
set_directive_array_reshape
set_directive_clock
```

Description:

```
SYNTAX
   set_clock_uncertainty [OPTIONS] <uncertainty> <clock_list>

DESCRIPTION

   The set_clock_uncertainty command sets a margin on the clock period defined
the clock period to create an effective clock period. If the clock uncertainty is not
```

- In interactive mode
  - ⇒ The help command lists the man page for all commands

```
Vivado_hls> help add_files

SYNOPSIS
    add_files  [OPTIONS] <src_files>
Etc…
```

Auto-Complete all commands using the tab key

Vitis HLS Summary

- Vitis HLS can be run under Windows and Linux
- Vitis HLS can be invoked through GUI and command line in Windows OS, and command line in Linux
- Vitis HLS project creation wizard involves
  - ⇒ Defining project name and location
  - ⇒ Adding design files
  - ⇒ Specifying testbench files
  - ⇒ Selecting clock and technology
- The top-level module in testbench is main() whereas top-level module in the design is the function to be synthesized

Tree_Int_HLS_Vitis

- Demo to be built and run now.
- Files on Canvas in CAD_Tool_Example/Tree_Int_HLS_Vitis
- arm_vitis folder contains C++ code for ARM core host code
- fpga_hls folder contains C++ code and testbench for Vitis HLS
- Create C:\ELEC522\<your netid> and create separate folders of:
  - ⇒ arm_vitis, flga_hls_ Vitis_HLS, Vivado, Vitis

- Download file to C:\ELEC522\<your netid> in the appropriate folders.
- Start Vitis HLS and use the C:\ELEC522\<your netid>\Tree_Int_HLS_Vitis\Vitis_HLS folder

Next Class

- Project 2 short < 5 minute individual presentations
- Please email slides to cavallar@rice.edu before 12 noon on September 29