



# Vitis HLS Design Flow

**Field Application Engineer**

Adaptive and Embedded Computing Group (AECG)

# Revision History

Date	Version	Description
11/08/23	1.0	Initial version for flow introduction.

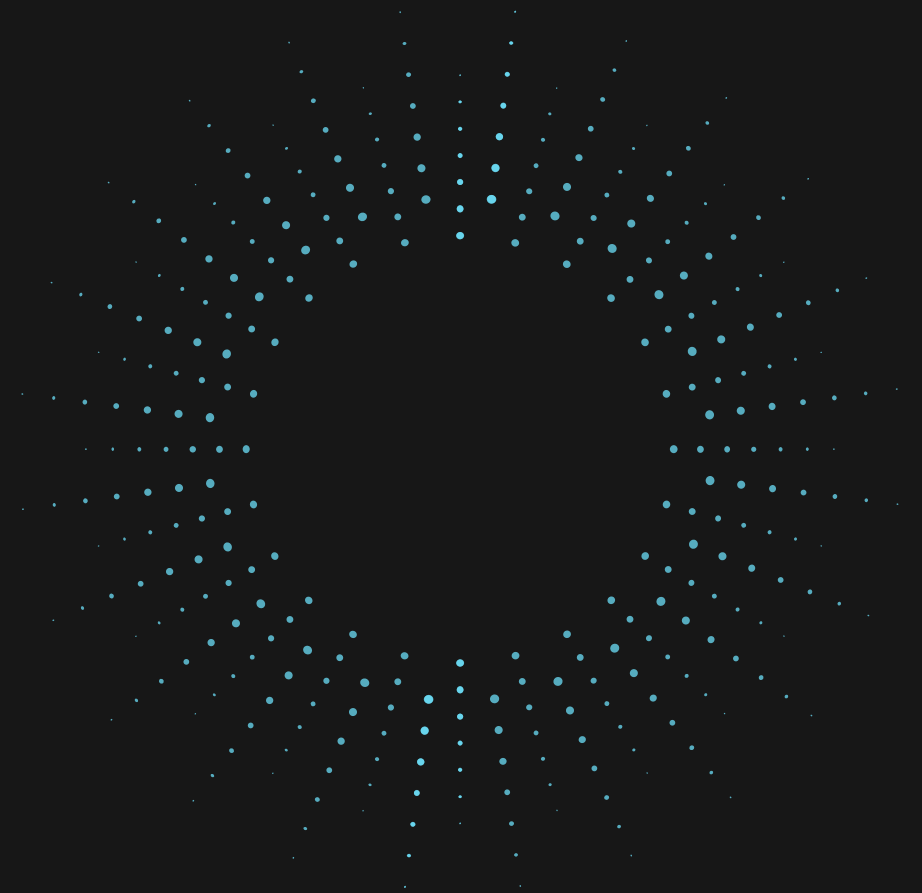
© Copyright 2021 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

NOTICE OF DISCLAIMER: The information disclosed to you hereunder (the "Information") is provided "AS-IS" with no warranty of any kind, express or implied. Xilinx does not assume any liability arising from your use of the Information. You are responsible for obtaining any rights you may require for your use of this Information. Xilinx reserves the right to make changes, at any time, to the Information without notice and at its sole discretion. Xilinx assumes no obligation to correct any errors contained in the Information or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE INFORMATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

---

# Vitis HLS 2021.1 Part

---

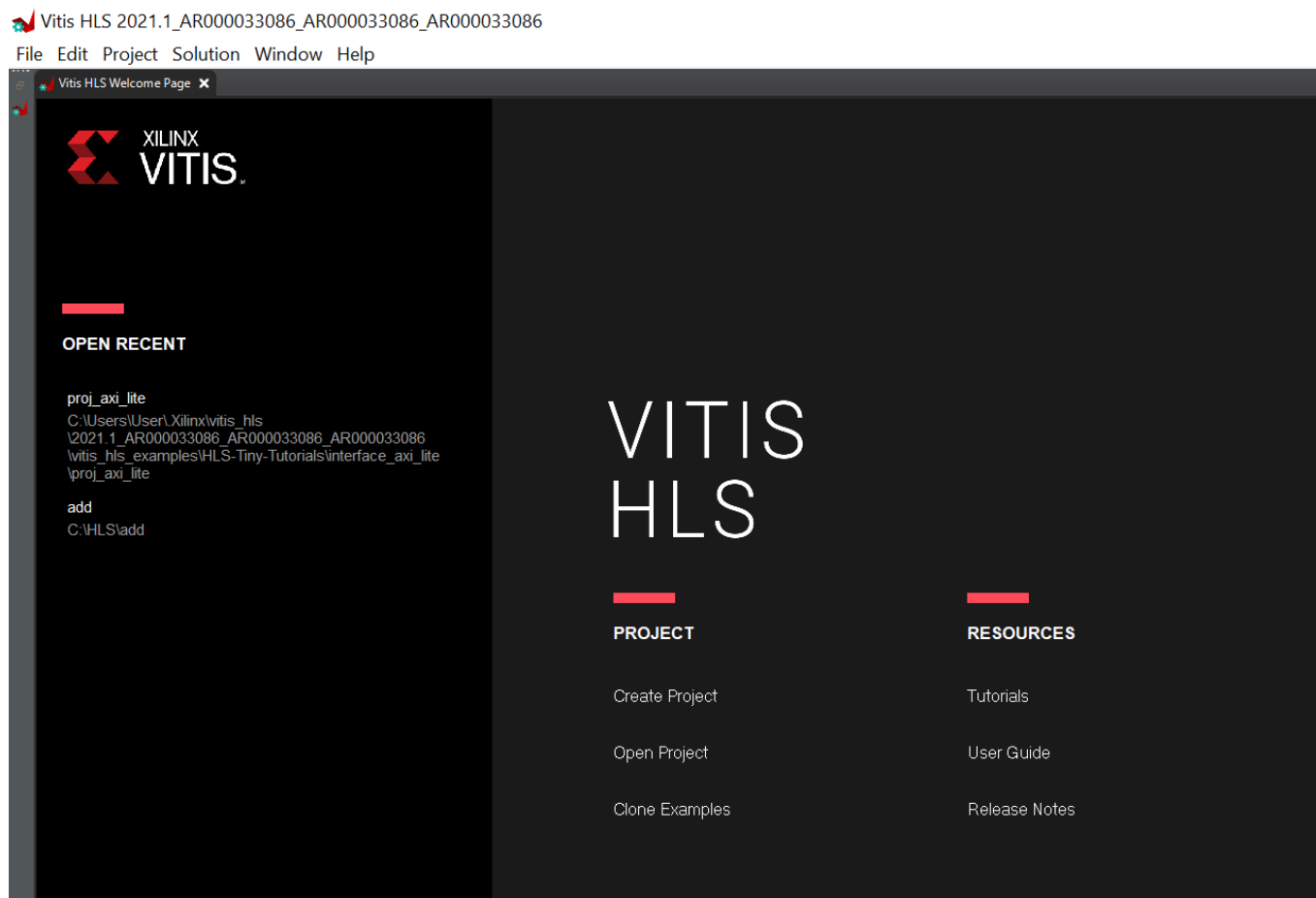


# Vitis HLS Design Flow

Open Vitis HLS 2021.1



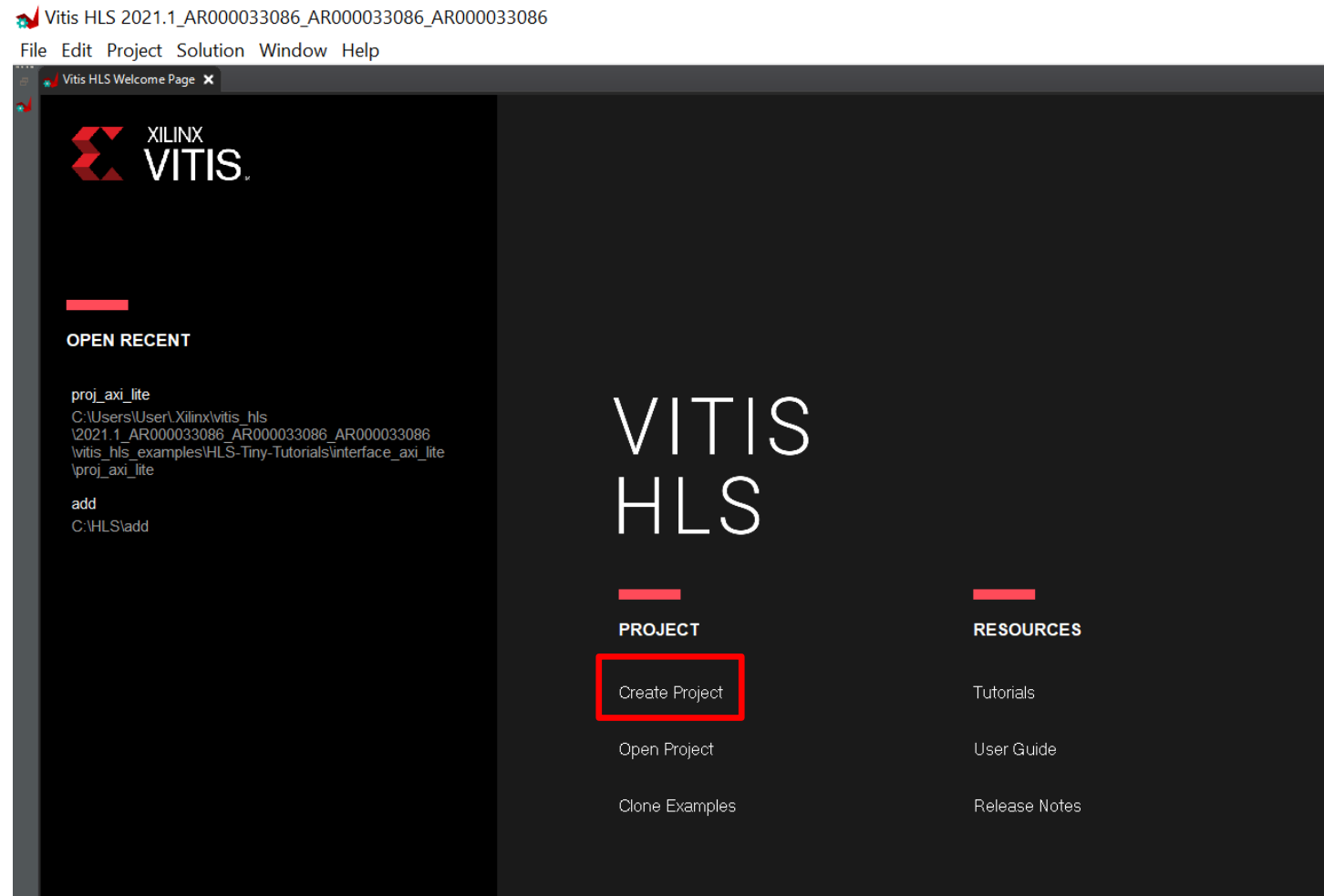
The GUI will be like the right image



# Vitis HLS Design Flow

We can create a custom project or download example code from Xilinx Vitis HLS github

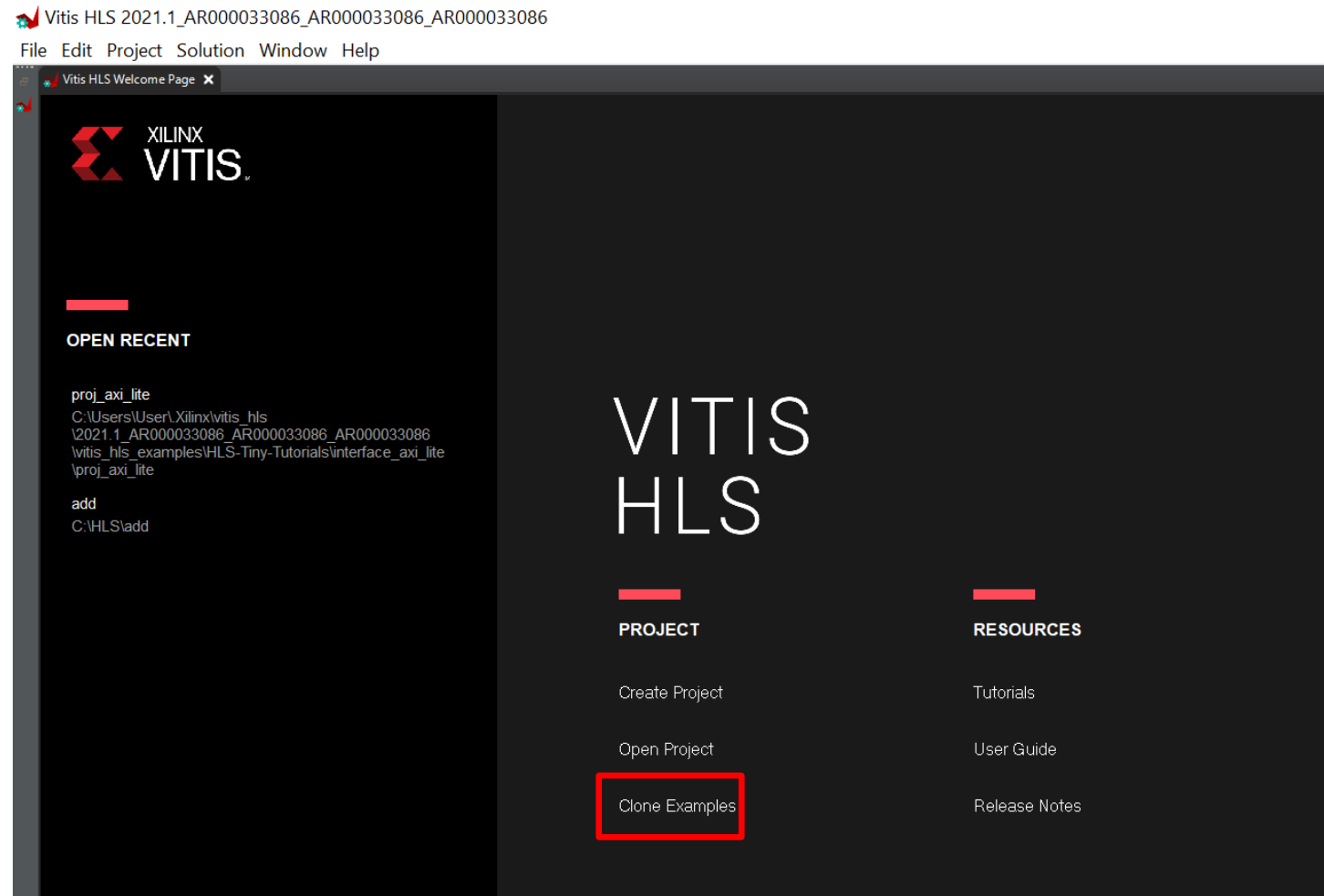
## 1. Create a custom project



# Vitis HLS Design Flow

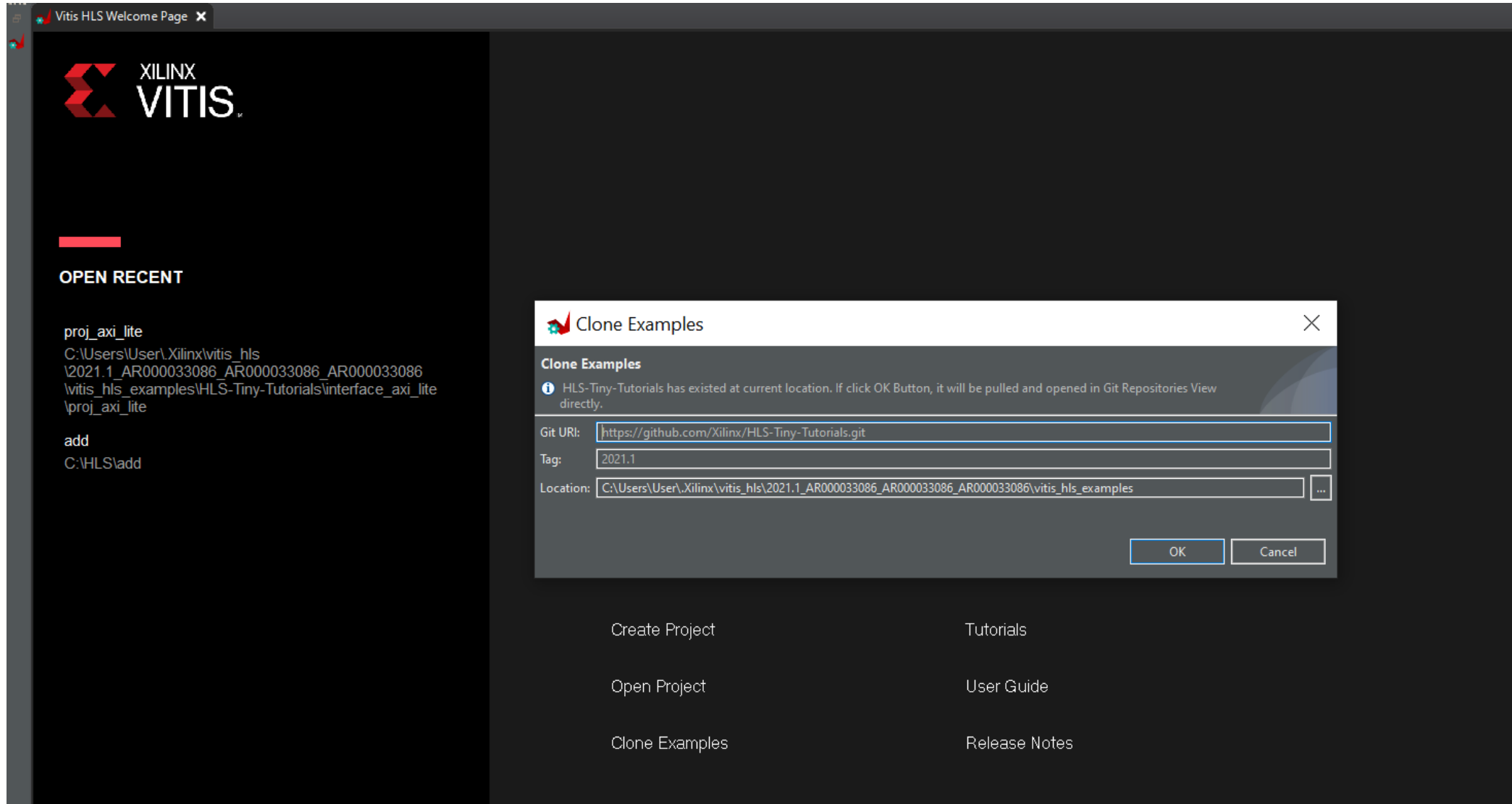
We can create a custom project or download example code from Xilinx Vitis HLS github

## 2. Download from Xilinx Vitis HLS github



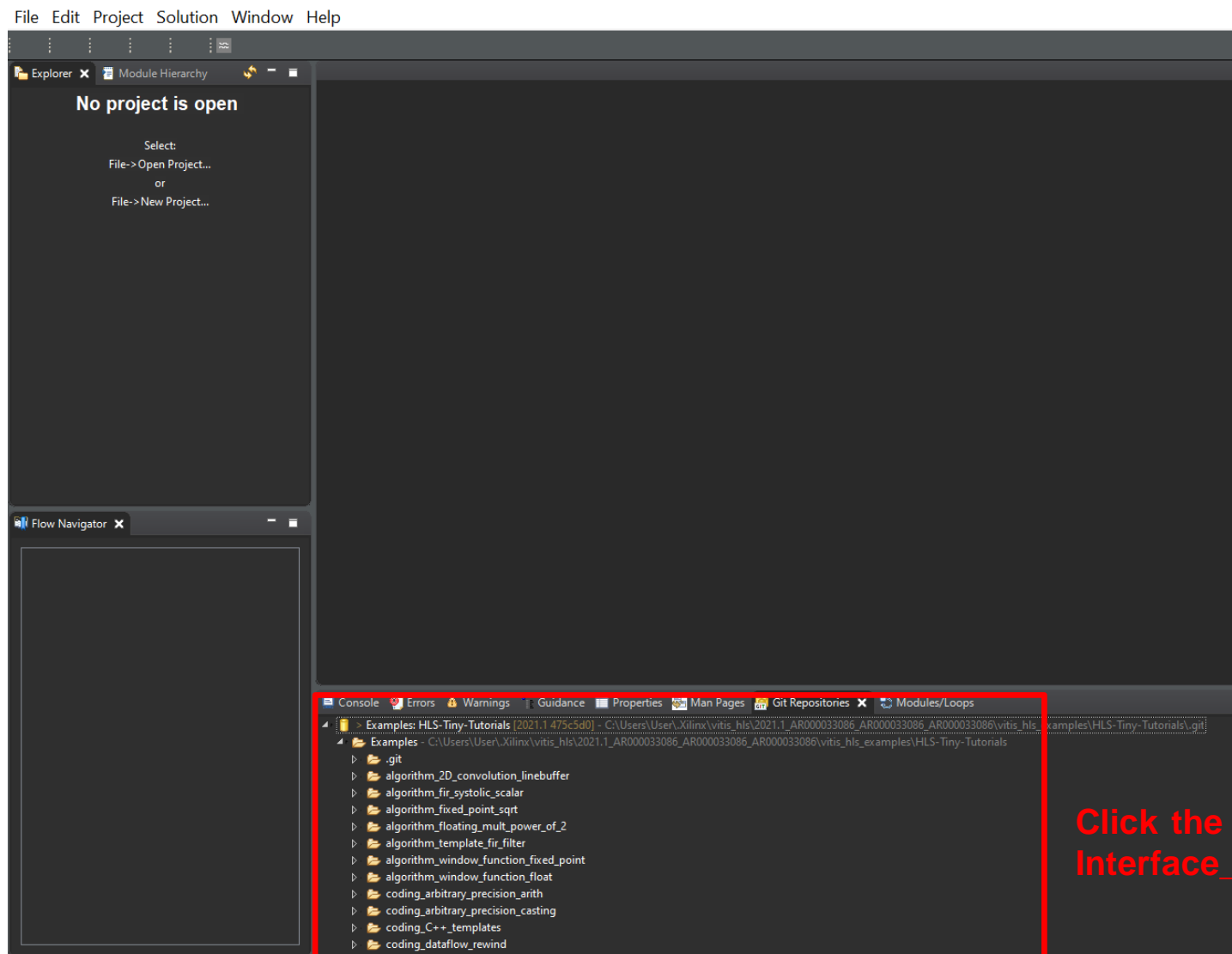
# Vitis HLS Design Flow

We will start from example code from Xilinx Vitis HLS github first, keep option default.



# Vitis HLS Design Flow

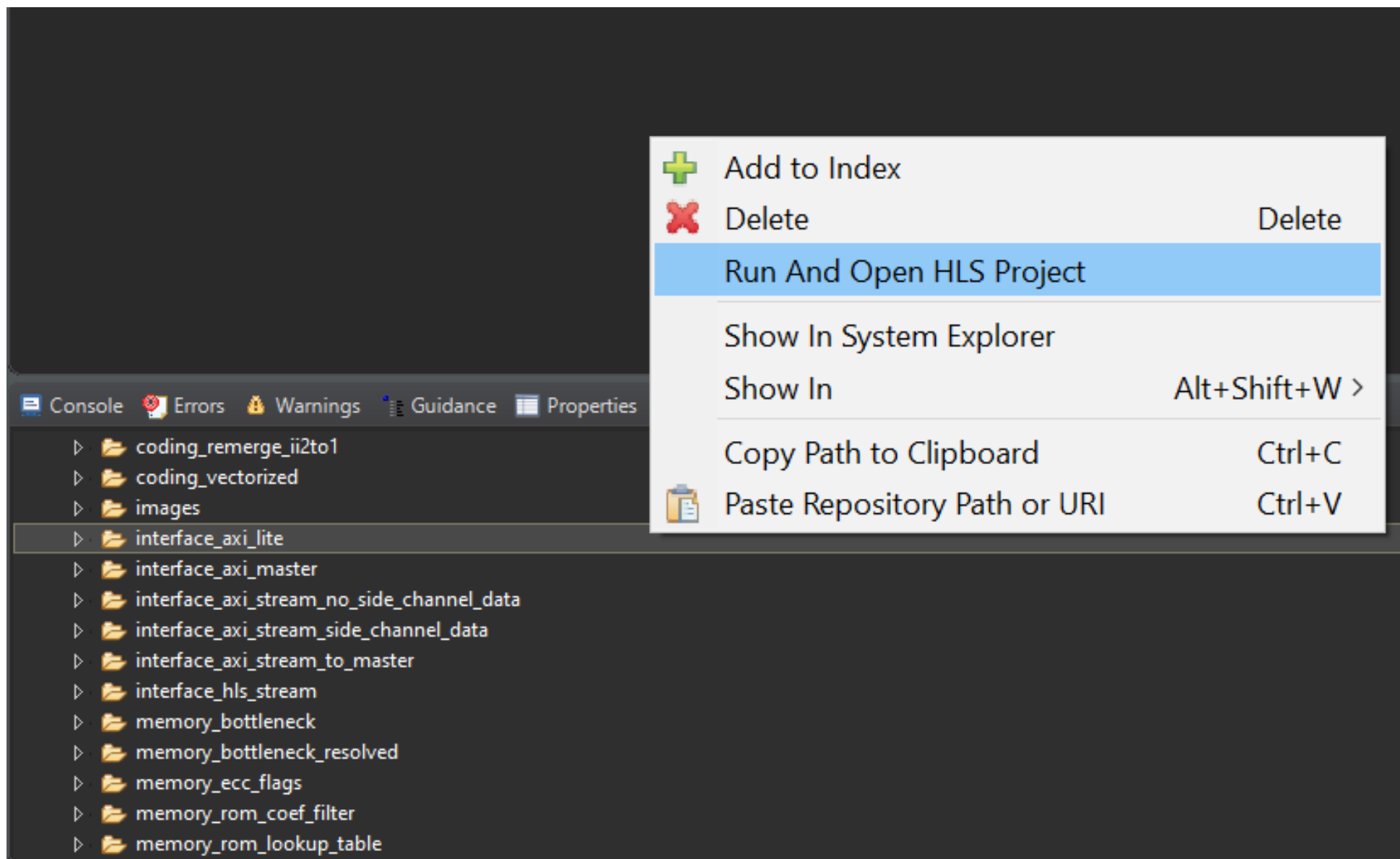
Click OK will download the whole example code directory automatically.



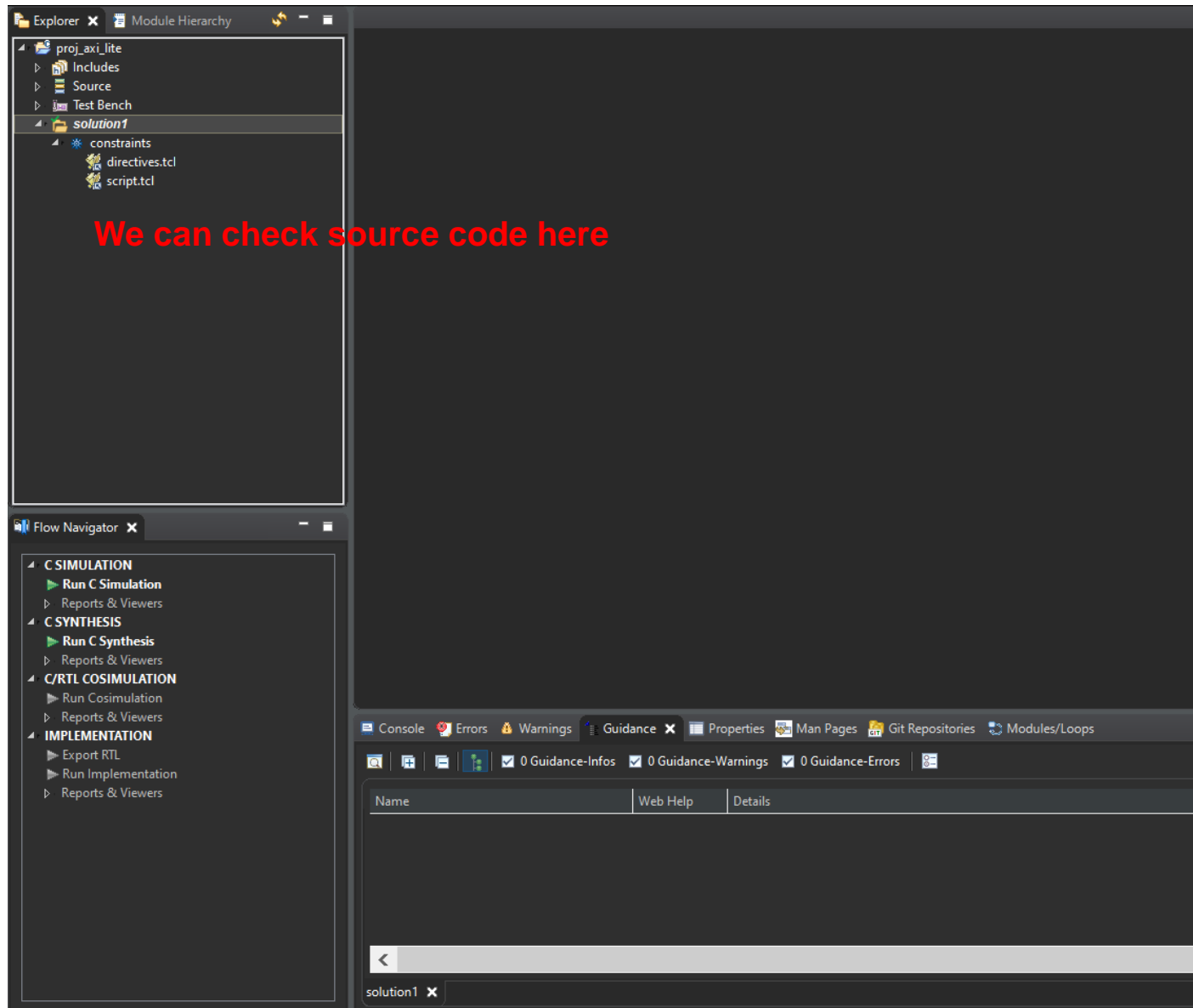
Click the directory you want, we choose `Interface_axi_lite` this time



# Vitis HLS Design Flow

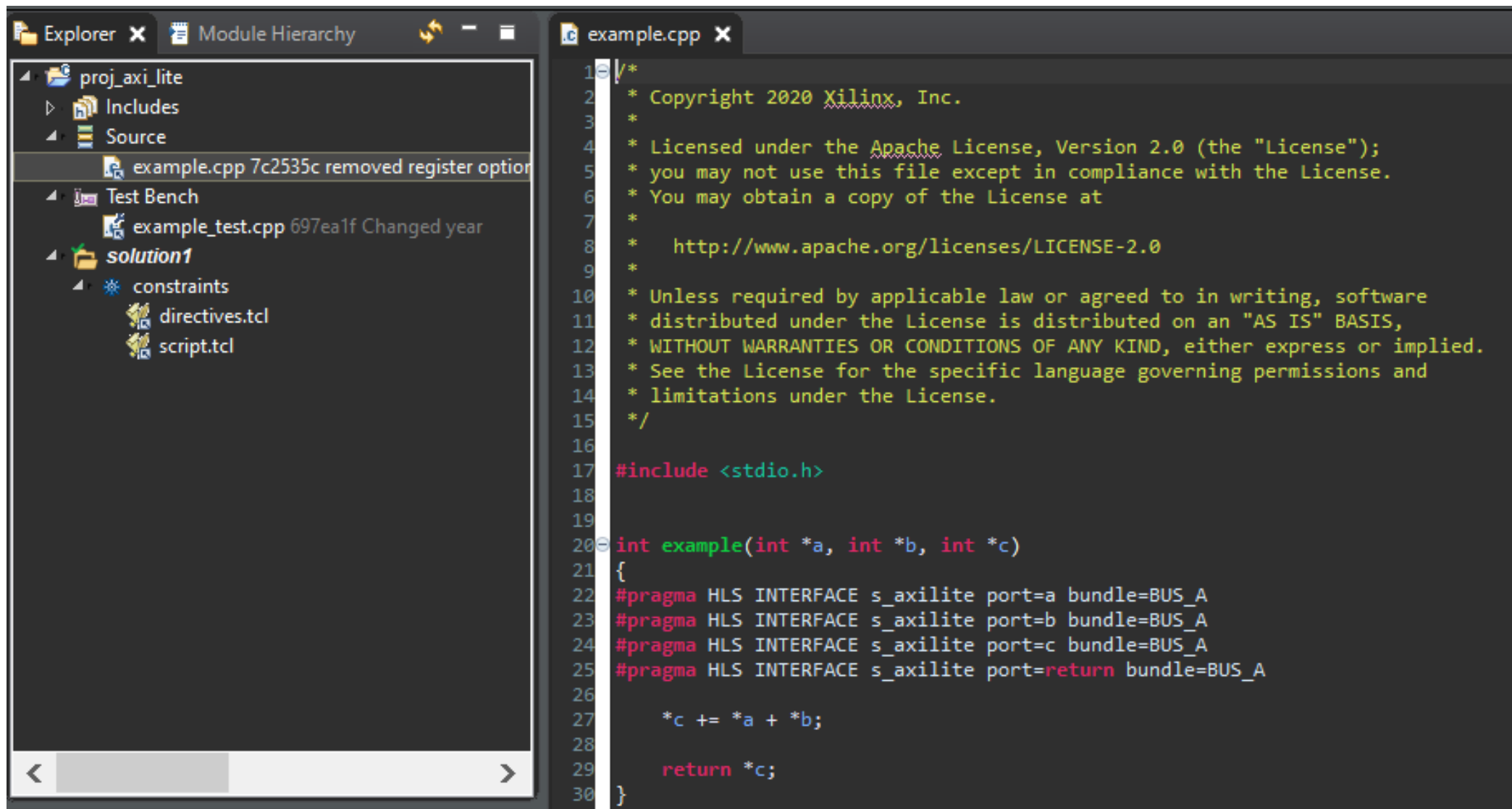


# Vitis HLS Design Flow



# Vitis HLS Design Flow

First we can check example.cpp



The screenshot displays the Vitis IDE interface. On the left, the 'Explorer' pane shows the project structure for 'proj\_axi\_lite'. It includes an 'Includes' directory and a 'Source' directory. Under 'Source', there is a file 'example.cpp' with a comment '7c2535c removed register option'. Below this, there is a 'Test Bench' section with 'example\_test.cpp' (commented '697ea1f Changed year') and a 'solution1' section containing 'constraints' (with sub-files 'directives.tcl' and 'script.tcl'). The main editor on the right shows the content of 'example.cpp'. The code starts with a multi-line comment block providing copyright information for Xilinx, Inc. (2020) and stating that the code is licensed under the Apache License, Version 2.0. It includes the URL 'http://www.apache.org/licenses/LICENSE-2.0'. The code then includes the standard C header '<stdio.h>'. The function 'example' is defined, taking three integer pointers as arguments. It uses three '#pragma HLS INTERFACE' directives to connect the pointers to the 's\_axilite' port of the 'BUS\_A' bundle. The function body increments the value pointed to by 'c' by the sum of the values pointed to by 'a' and 'b'. Finally, it returns the value of 'c'.

```
1  /*
2   * Copyright 2020 Xilinx, Inc.
3   *
4   * Licensed under the Apache License, Version 2.0 (the "License");
5   * you may not use this file except in compliance with the License.
6   * You may obtain a copy of the License at
7   *
8   *   http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in writing, software
11  * distributed under the License is distributed on an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13  * See the License for the specific language governing permissions and
14  * limitations under the License.
15  */
16
17  #include <stdio.h>
18
19
20  int example(int *a, int *b, int *c)
21  {
22    #pragma HLS INTERFACE s_axilite port=a bundle=BUS_A
23    #pragma HLS INTERFACE s_axilite port=b bundle=BUS_A
24    #pragma HLS INTERFACE s_axilite port=c bundle=BUS_A
25    #pragma HLS INTERFACE s_axilite port=return bundle=BUS_A
26
27    *c += *a + *b;
28
29    return *c;
30  }
```

# Vitis HLS Design Flow

```
#include <stdio.h>

int example(int *a, int *b, int *c)
{
    #pragma HLS INTERFACE s_axilite port=a bundle=BUS_A
    #pragma HLS INTERFACE s_axilite port=b bundle=BUS_A
    #pragma HLS INTERFACE s_axilite port=c bundle=BUS_A
    #pragma HLS INTERFACE s_axilite port=return bundle=BUS_A

    *c += *a + *b;

    return *c;
}
```

Define AXI-Lite Port

#pragma HLS interface <mode> port=<name> (register) bundle=<string>

# Vitis HLS Design Flow

```
#pragma HLS interface <mode> port=<name> (register) bundle=<string>
```

- *<mode>*:

Specifies the interface protocol mode for the function arguments. In this case we chose s\_axilite.

- *port=<name>*:

Specifies the name of the function argument which the INTERFACE pragma applies to.

- *(register)*:

Is an optional keyword to register (i.e. store) the signal and any associated protocol signals. It causes the signals to persist until at least the last cycle of the function execution. This option is not applicable to axilite.

# Vitis HLS Design Flow

```
#pragma HLS interface <mode> port=<name> (register) bundle=<string>
```

- bundle=<string>:

This keyword allows you to manually group the port signals into one data bus. If the function return is specified as an AXI4-Lite interface (i.e. line 25 in the example code), all of the data ports are automatically bundled into a single bus. Vitis HLS would use the default bundle name **control**, if we did not explicitly provide a bundle name here. This is a common practice when another device, such as a CPU, is used to configure and control when the block starts and stops operation.

- s\_axilite port=return:

Setting a function argument of type s\_axilite and port name 'return' will create an interrupt signal in the IP block. You can program the interrupt through the AXI4-Lite interface, and the C driver files.

# Vitis HLS Design Flow

```
#include <stdio.h>

int example(int *a, int *b, int *c)
{
    #pragma HLS INTERFACE s_axilite port=a bundle=BUS_A
    #pragma HLS INTERFACE s_axilite port=b bundle=BUS_A
    #pragma HLS INTERFACE s_axilite port=c bundle=BUS_A
    #pragma HLS INTERFACE s_axilite port=return bundle=BUS_A

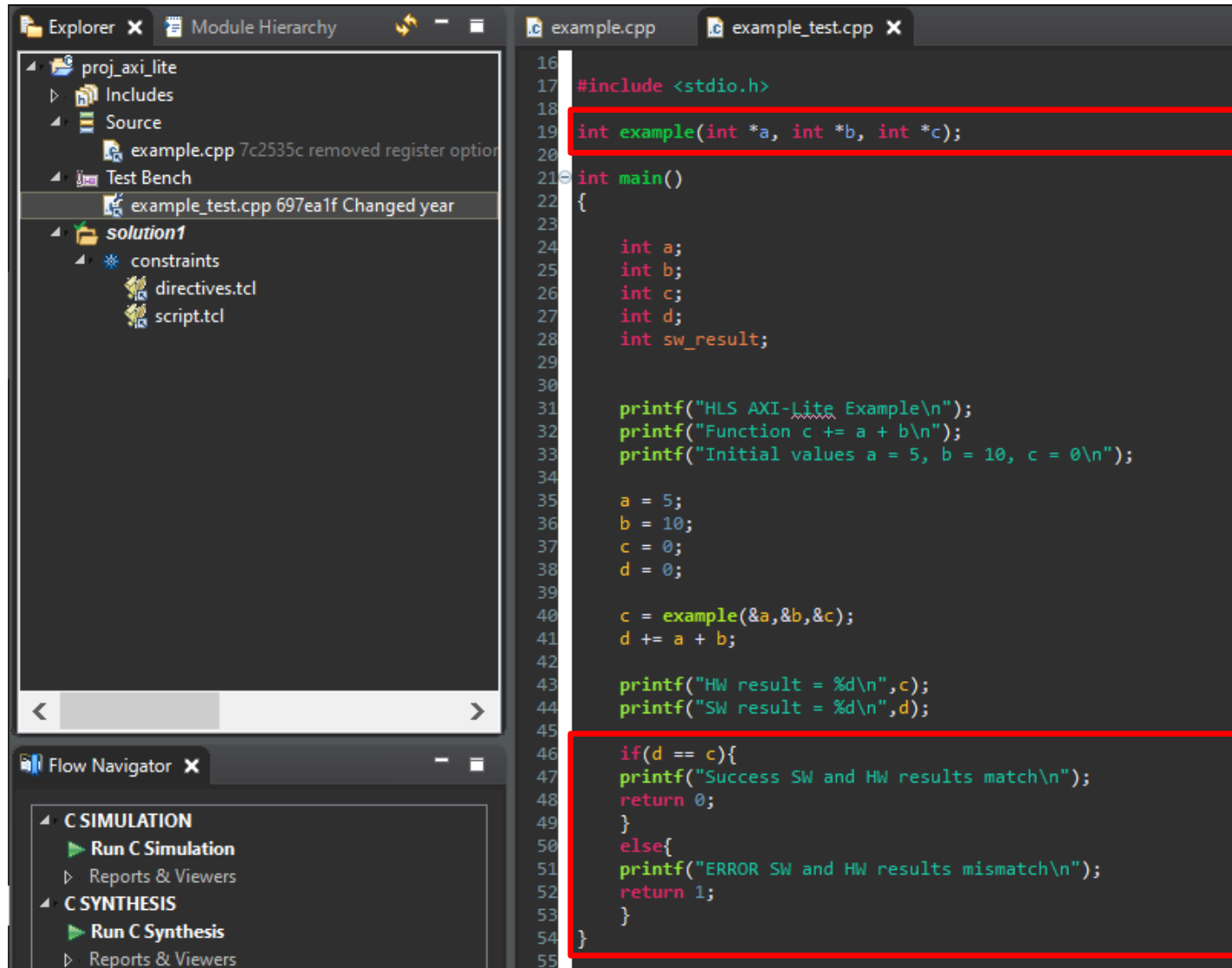
    *c += *a + *b;

    return *c;
}
```

Simple Function or Algorithm

# Vitis HLS Design Flow

Second we can check `example_test.cpp`, is like RTL testbench



```
16
17 #include <stdio.h>
18
19 int example(int *a, int *b, int *c);
20
21 int main()
22 {
23
24     int a;
25     int b;
26     int c;
27     int d;
28     int sw_result;
29
30
31     printf("HLS AXI-Lite Example\n");
32     printf("Function c += a + b\n");
33     printf("Initial values a = 5, b = 10, c = 0\n");
34
35     a = 5;
36     b = 10;
37     c = 0;
38     d = 0;
39
40     c = example(&a,&b,&c);
41     d += a + b;
42
43     printf("HW result = %d\n",c);
44     printf("SW result = %d\n",d);
45
46     if(d == c){
47         printf("Success SW and HW results match\n");
48         return 0;
49     }
50     else{
51         printf("ERROR SW and HW results mismatch\n");
52         return 1;
53     }
54 }
55
```

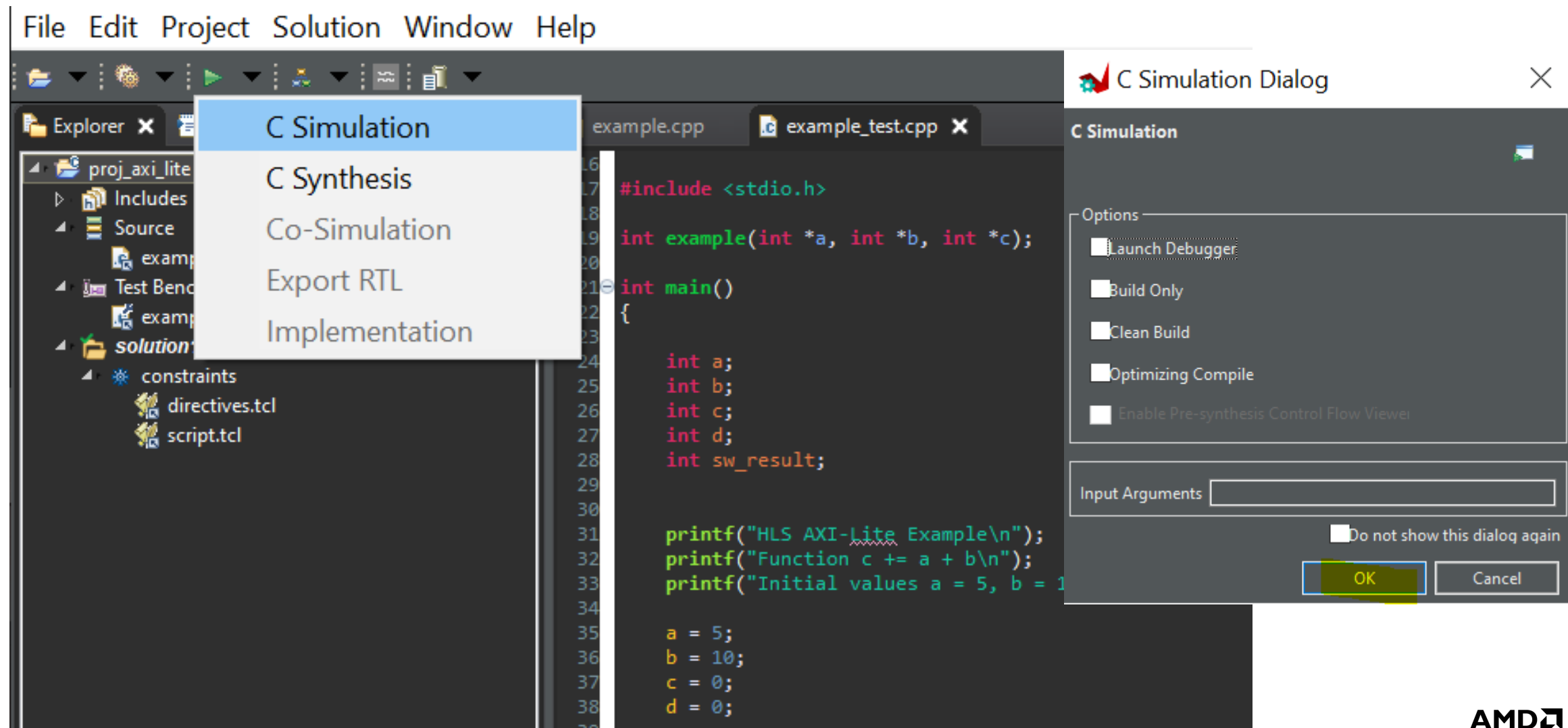
call function which will be tested

check function behavior



# Vitis HLS Design Flow

C simulation ---> run example.cpp and example\_test.cpp



# Vitis HLS Design Flow

C synthesis ---> Convert C code to RTL code

```
example.cpp  example_test.cpp  example_csim.log X
1 INFO: [SIM 2] ***** CSIM start *****
2 INFO: [SIM 4] CSIM will launch GCC as the compiler.
3   Compiling ../../../../example_test.cpp in debug mode
4   Compiling ../../../../example.cpp in debug mode
5   Generating csim.exe
6 HLS AXI-Lite Example
7 Function c += a + b
8 Initial values a = 5, b = 10, c = 0
9 HW result = 15
10 SW result = 15
11 Success SW and HW results match
12 INFO: [SIM 1] CSim done with 0 errors.
13 INFO: [SIM 3] ***** CSIM finish *****
14
```

```
Console X Errors Warnings Guidance Properties Man Pages Git Repositories Modules/Loops
Vitis HLS Console
  Compiling ../../../../example_test.cpp in debug mode
  Compiling ../../../../example.cpp in debug mode
  Generating csim.exe
HLS AXI-Lite Example
Function c += a + b
Initial values a = 5, b = 10, c = 0
HW result = 15
SW result = 15
Success SW and HW results match
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
INFO: [HLS 200-111] Finished Command csim_design CPU user time: 0 seconds. CPU system time: 1 seconds. Elapsed time: 2.286 seconds; current allocated memory: 288.811 MB.
Finished C simulation.
```

# Vitis HLS Design Flow

C synthesis ---> Convert C code to RTL code

The screenshot displays the Vitis IDE interface during the C synthesis process. The 'Tools' menu is open, showing options like 'C Simulation', 'C Synthesis', 'Co-Simulation', 'Export RTL', and 'Implementation'. The console window shows the following log:

```
example.cpp example_test.cpp example_csim.log
1 INFO: [SIM 2] ***** CSIM start *****
2 INFO: [SIM 4] CSIM will launch GCC as the compiler.
3   Compiling ../../../../example_test.cpp in debug mode
4   Compiling ../../../../example.cpp in debug mode
5   Generating csim.exe
6 HLS AXI-Lite Example
7 Function c += a + b
8 Initial values a = 5, b = 10, c = 0
9 HW result = 15
10 SW result = 15
11 Success SW and HW results match
12 INFO: [SIM 1] CSim done with
13 INFO: [SIM 3] *****
14
```

The 'C Synthesis - Active Solution' dialog is open, showing the 'Part Selection' section. The 'Part' is set to 'ZYNQ-7 ZC702 Evaluation Board (xc7z020clg484-1)'. A red box highlights the '...' button next to the part name, with a yellow callout 'choose parts/boards' pointing to it. The 'Flow Target' is set to 'Vivado IP Flow Target'. The 'Clock' section shows 'Period: 200MHz' and 'Uncertainty:'. The 'Do not show this dialog again' checkbox is checked. The 'OK' and 'Cancel' buttons are at the bottom.

# Vitis HLS Design Flow

C synthesis --> the result

example.cppexample\_test.cppexample\_csim.logSynthesis Summary(solution1) x

Synthesis Summary Report of 'example'

General Information

Date:Wed Nov 8 17:10:34 2023

Version:2021.1\_AR000033086\_AR000033086 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)

Project:proj\_axi\_lite

Solution:solution1 (Vivado IP Flow Target)

Product family:zynq

Target device:xc7z020-clg484-1

Timing Estimate

Target	Estimated	Uncertainty
5.00 ns	4.371 ns	1.35 ns

Performance & Resource Estimates

Modules && Loops

Issue Type

Violation Type

Distance

Slack

Latency(cycles)

Latency(ns)

Iteration Latency

Interval

Trip Count

Pipelined

BRAM

DSP

FF

LUT

URAM

example

Timing Violation

-0.72

2

10.000

-

3

-

no

0

0

351

444

0

HW Interfaces

S\_AXILITE

Interface	Data Width	Address Width	Offset	Register
s_axi_BUS_A	32	6	24	0

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst_n	reset	ap_rst_n
interrupt	interrupt	interrupt
ap_ctrl	ap_ctrl_hs	

SW I/O Information

Top Function Arguments

Argument	Direction	Datatype
----------	-----------	----------

20

AMD

together we advance\_

# Vitis HLS Design Flow

## C synthesis --> the result

Synthesis Summary Report of 'example'

General Information

Date: Wed Nov 8 17:10:34 2023

Version: 2021.1\_AR000033086\_AR000033086 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)

Project: proj\_axi\_lite

Solution: solution1 (Vivado IP Flow Target)

Product family: zynq

Target device: xc7z020-clg484-1

Timing Estimate

Target	Estimated	Uncertainty
5.00 ns	4.371 ns	1.35 ns

Performance & Resource Estimates

Modules && Loops

Issue Type

Violation Type

Distance

Slack

Latency(cycles)

Latency(ns)

Iteration Latency

Interval

Trip Count

Pipelined

BRAM

DSP

FF

LUT

URAM

example

Timing Violation

-0.72

2

10.000

3

no

0

0

351

444

0

HW Interfaces

S\_AXILITE

Interface	Data Width	Address Width	Offset	Register
s_axi_BUS_A	32	6	24	0

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst_n	reset	ap_rst_n
interrupt	interrupt	interrupt
ap_ctrl	ap_ctrl_hs	

SW I/O Information

Top Function Arguments

Argument	Direction	Datatype
----------	-----------	----------

Console

Errors

Warnings

Guidance

Properties

Man Pages

Git Repository

Vitis HLS Console

INFO: [HLS 200-106] Finished creating RTL model for 'example'.

INFO: [HLS 200-111] Finished Creating RTL model: CPU user time: 0 seconds. CPU sy

INFO: [HLS 200-111] Finished Generating all RTL models: CPU user time: 0 seconds.

INFO: [HLS 200-111] Finished Updating report files: CPU user time: 1 seconds. CPU

INFO: [VHDL 208-304] Generating VHDL RTL for example.

INFO: [VLOG 209-307] Generating Verilog RTL for example.

INFO: [HLS 200-789] \*\*\*\* Estimated Fmax: 228.78 MHz

INFO: [HLS 200-111] Finished Command csynth\_design CPU user time: 2 seconds. CPU

INFO: [HLS 200-112] Total CPU user time: 6 seconds. Total CPU system time: 2 sec

Finished C synthesis.

21

together we advance\_

# Vitis HLS Design Flow

C Co-Simulation ----> Co-simulation C with RTL code to verify the same behavior they have

The screenshot displays the Vitis IDE interface. On the left, the 'Explorer' pane shows a project named 'proj\_axi\_lite' with folders for 'Includes', 'Source', 'Test Bench', and 'solution'. The 'solution' folder is expanded, showing 'constraints', 'directives.tcl', 'script.tcl', 'csim', 'impl', and 'syn'. A context menu is open over the 'solution' folder, listing options: 'C Simulation', 'C Synthesis', 'Co-Simulation' (highlighted), 'Export RTL', and 'Implementation'. The main window shows the 'example.cpp' file and the 'example\_test' file. The 'Synthesis Summary Report' is visible, with sections for 'General Information' and 'Timing Estimate'. The 'Timing Estimate' section contains a table with the following data:

Target	Estimated	Uncertainty
5.00 ns	4.371 ns	1.35 ns

On the right, the 'Co-simulation Dialog' is open. It has a title bar 'Co-simulation Dialog' and a close button. The dialog is titled 'C/RTL Co-simulation' and has a checked checkbox. Under 'RTL Simulator Settings', 'Vivado XSIM' is selected in a dropdown, and 'Verilog' and 'VHDL' radio buttons are present. There are checkboxes for 'Setup Only', 'Optimizing Compile', and 'Random Stall'. The 'Input Arguments' field is empty. The 'Dump Trace' dropdown is set to 'none'. The 'Compiled Library Location' field is empty with a 'Browse...' button. Under 'Extra Options for DATAFLOW', there are checkboxes for 'Wave Debug (Vivado XSIM only and "Dump Trace" != none)', 'Disable Deadlock Detection', 'Channel (PIPO/FIFO) Profiling', and 'Dynamic Deadlock Prevention'. At the bottom, there is a checkbox for 'Do not show this dialog again' and 'OK' and 'Cancel' buttons.

# Vitis HLS Design Flow

**C Co-Simulation ---> Co-simulation C with RTL code to verify the same behavior they have**

测试激励文件会为综合验证顶层函数输出，如果输出正确，则会向测试激励文件的main() 函数返回 0 值。Vitis HLS 针对 C 语言仿真和 C/RTL 协同仿真使用相同的返回值，以判定结果是否正确。如果C 语言测试激励文件返回非 0 值，Vitis HLS 就会报告仿真失败

# Vitis HLS Design Flow

## C Co-Simulation ----> the result

The screenshot shows the Vitis Co-simulation Report for a project named 'example'. The report is displayed in a window with tabs for 'example.cpp', 'example\_test.cpp', 'example\_csim.log', 'Synthesis Summary(solution1)', and 'Co-simulation Report(solution1)'. The report is organized into three main sections: General Information, Cosim Options, and Performance Estimates.

**General Information**

Date:	Wed Nov 8 17:20:33 TST 2023	Solution:	solution1 (Vivado IP Flow Target)
Version:	2021.1_AR000033086_AR000033086_AR000033086 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)	Product family:	zynq
Project:	proj_axi_lite	Target device:	xc7z020-clg484-1
Status:	Pass		

**Cosim Options**

Tool: Vivado XSIM  
RTL: Verilog

**Performance Estimates**

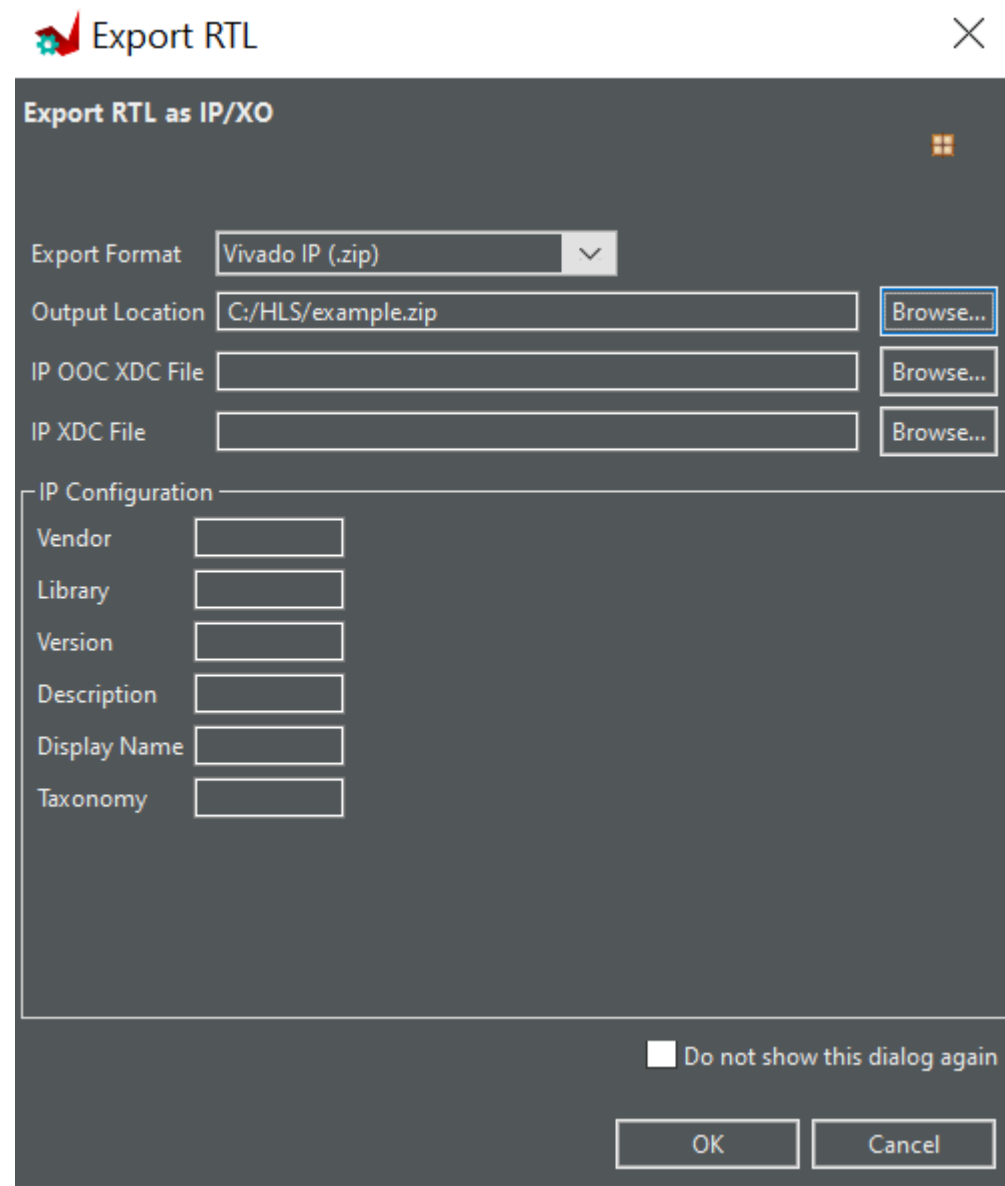
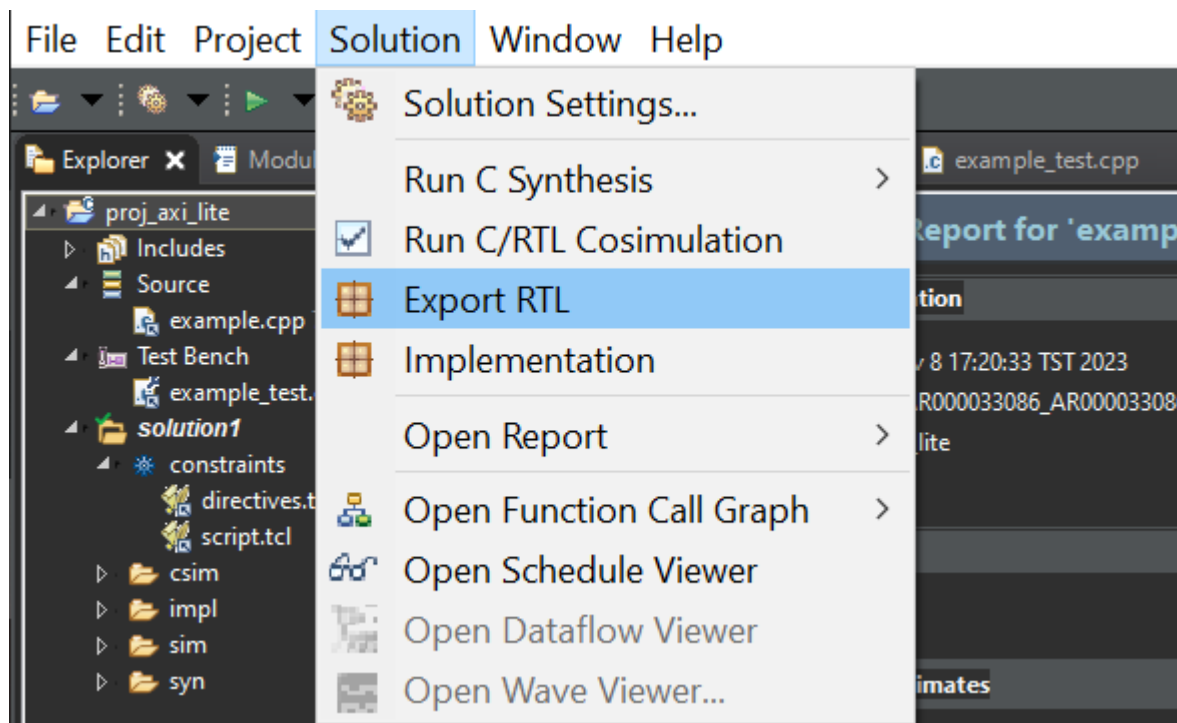
Icons: [Tree], [Add], [File], [Run]

Modules	Avg II	Max II	Min II	Avg Latency	Max Latency	Min Latency
example				2	2	2



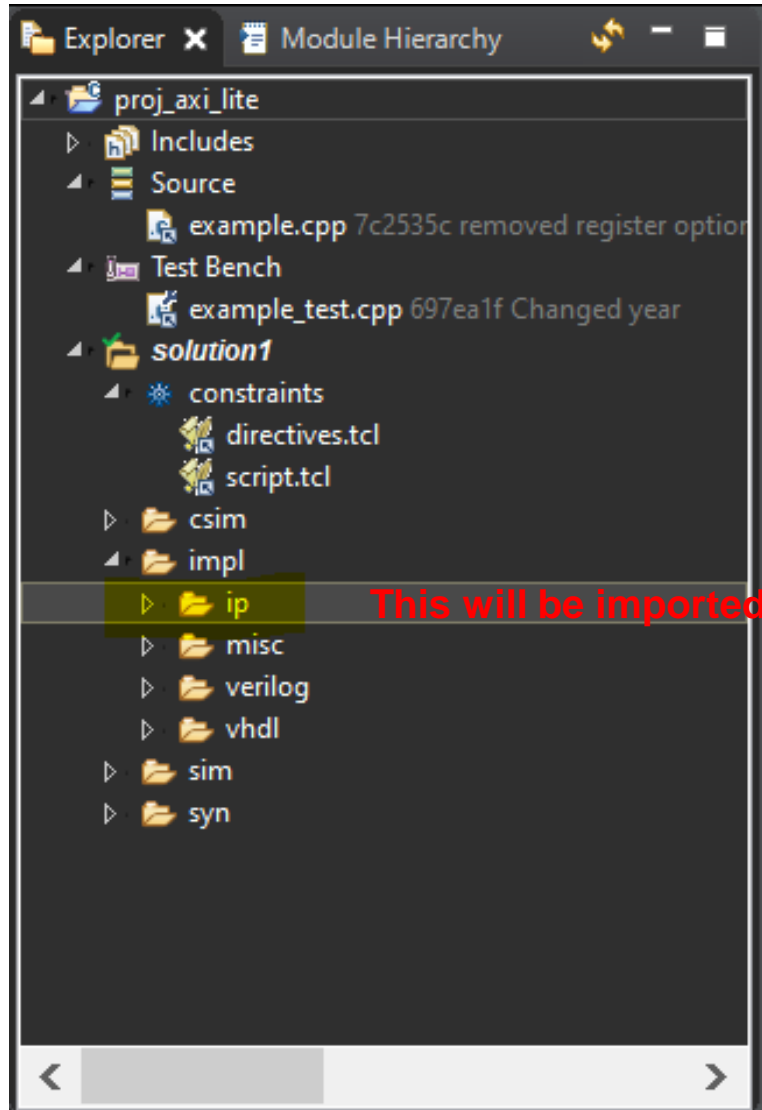
# Vitis HLS Design Flow

## Export HLS IP



# Vitis HLS Design Flow

## Export HLS IP

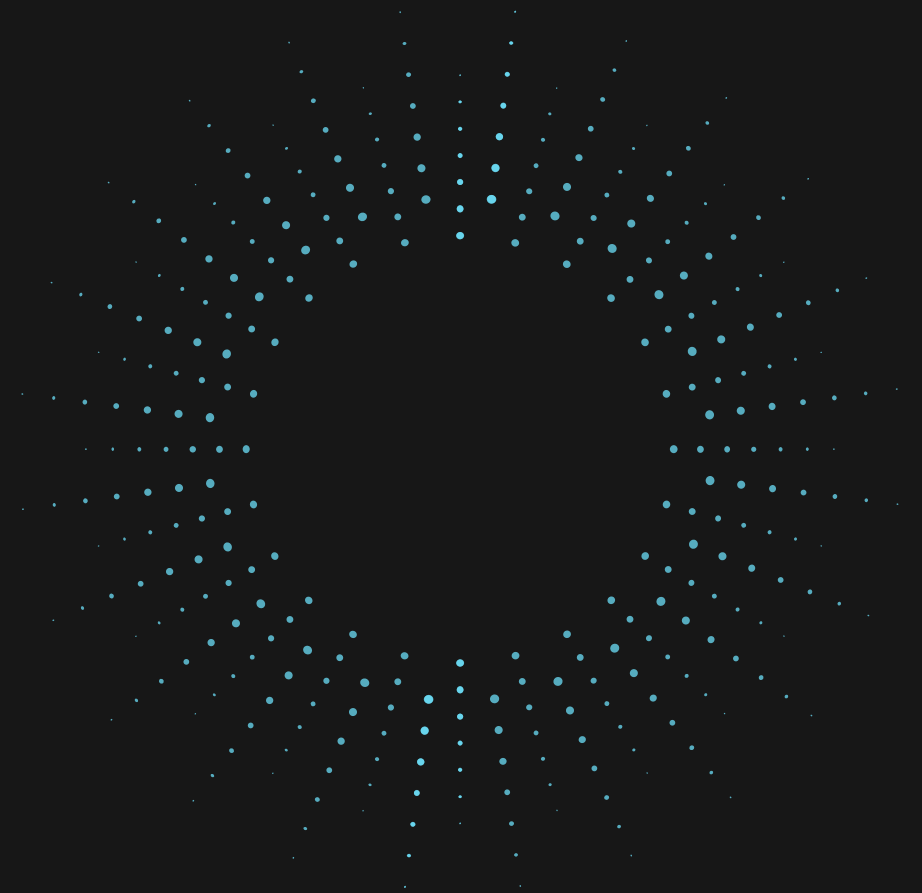


This will be imported to Vivado IP catalog

---

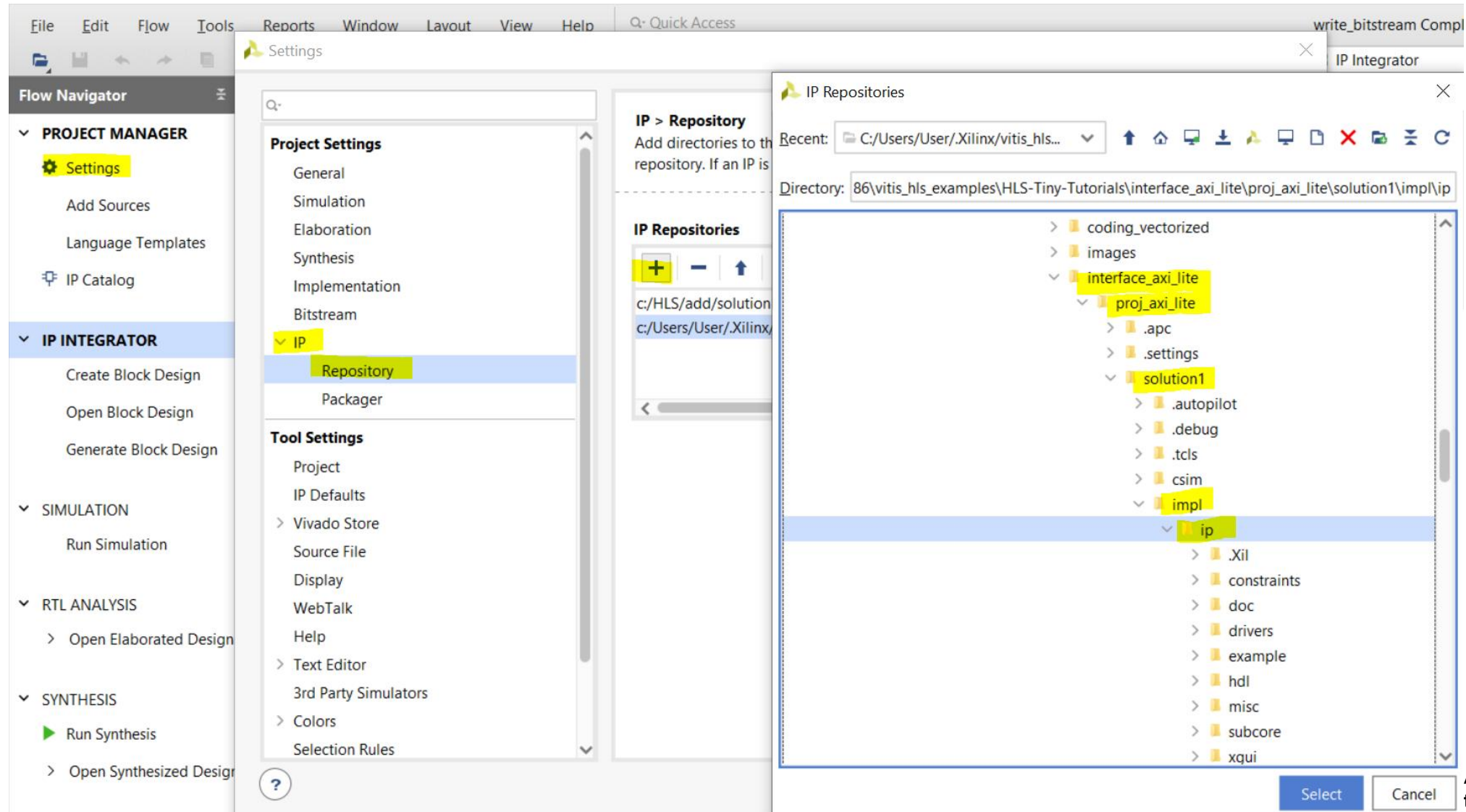
# Vivado 2021.1 Part

---



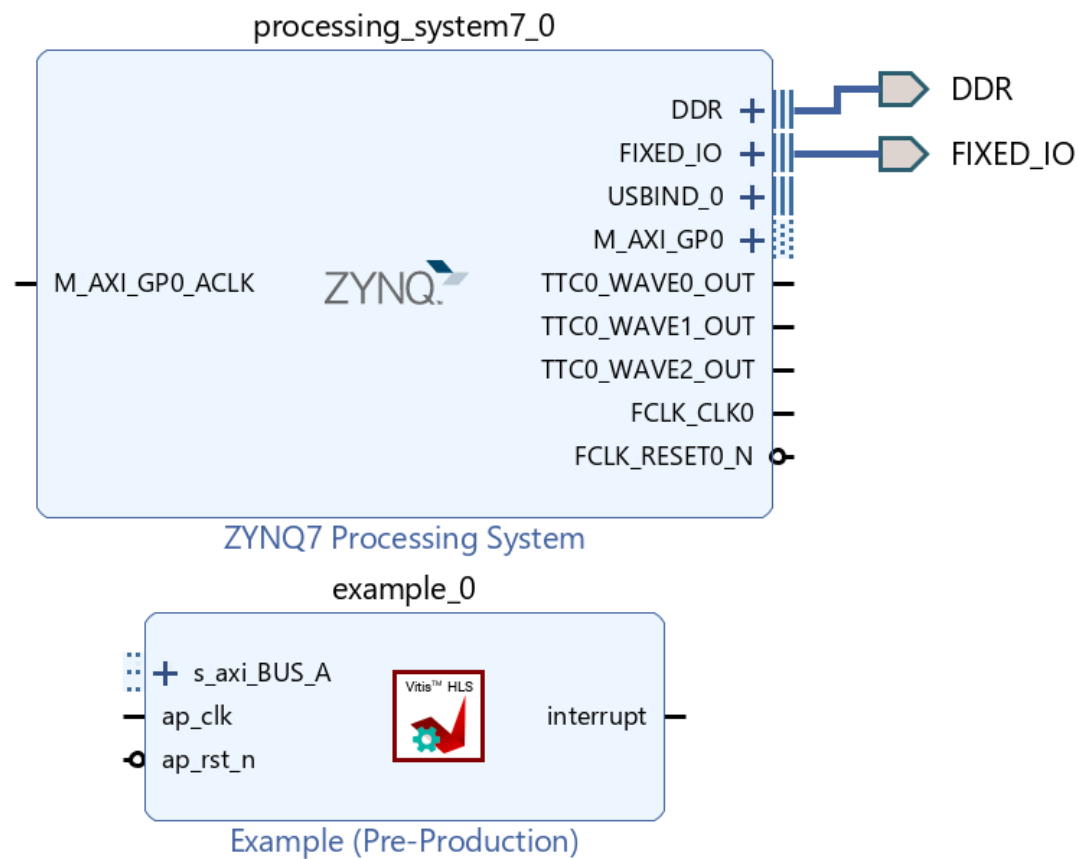
# Vitis HLS Design Flow

Create project with ZC702 and add IP from Vitis HLS directory



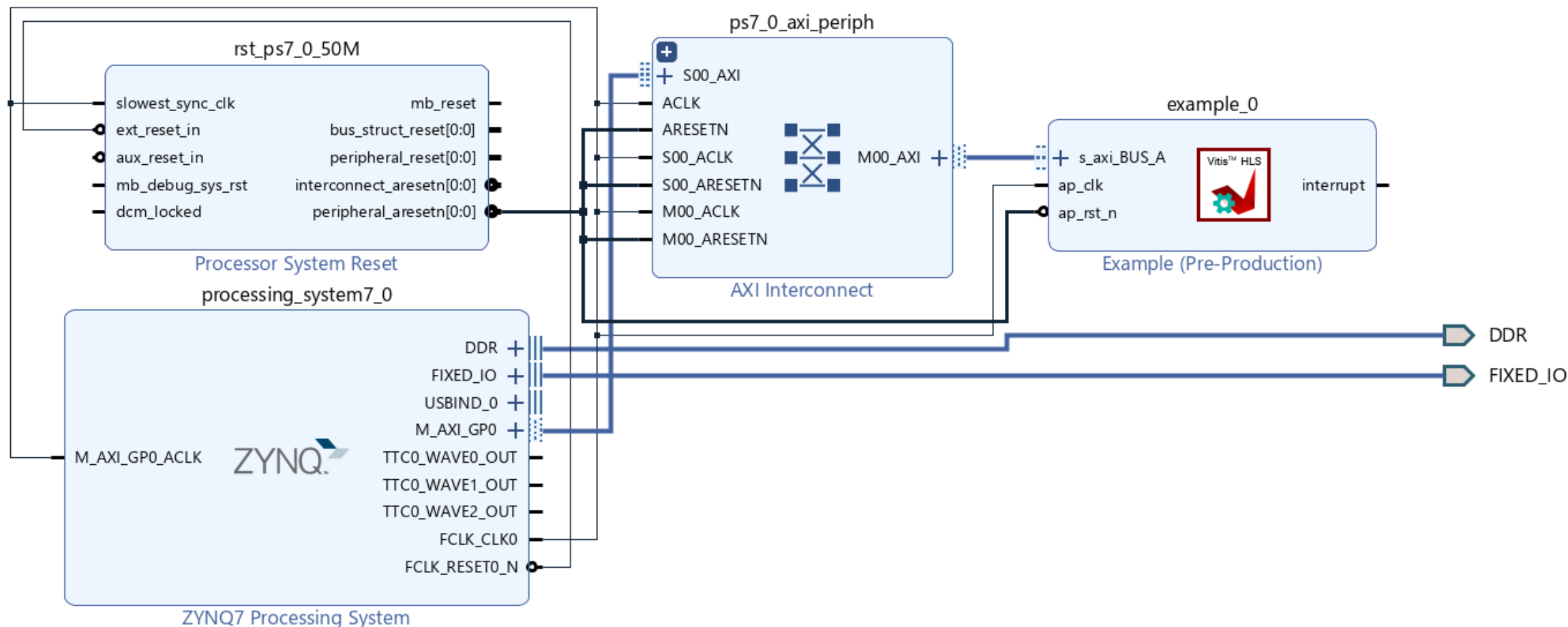
# Vitis HLS Design Flow

Create Block Design and put two IP including HLS on it



# Vitis HLS Design Flow

## Run Connection Automation



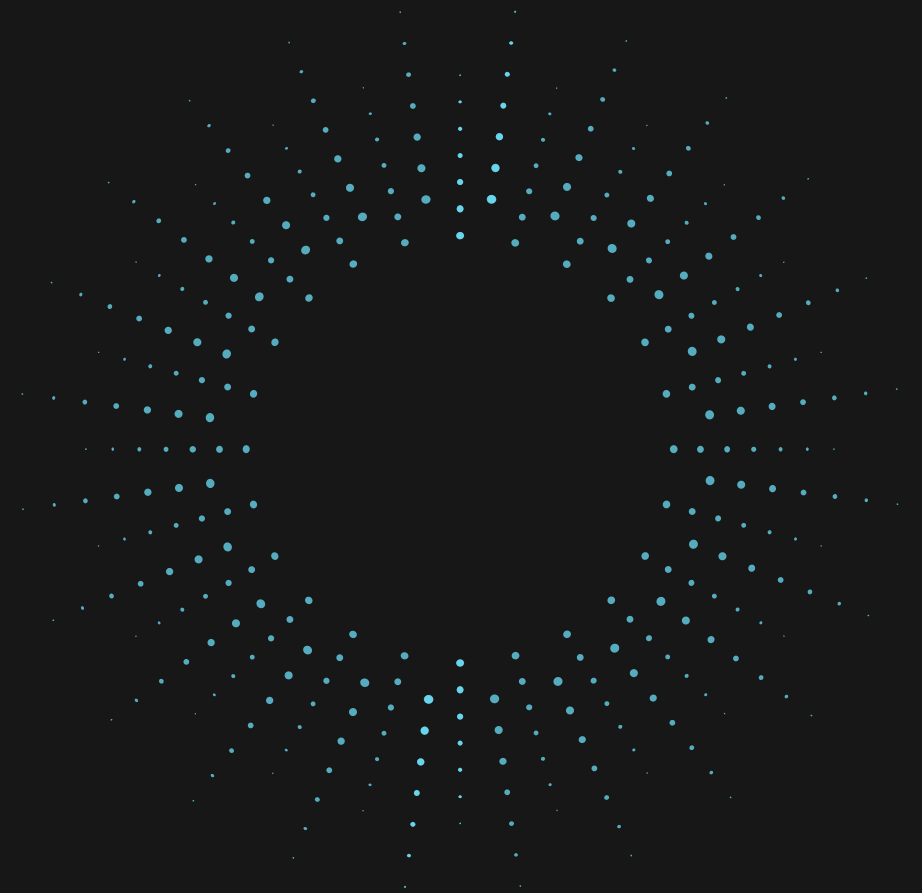
# Vitis HLS Design Flow

Generate HDL wrapper and bitstream  
Generate .xsa finally

---

# Vitis 2021.1 Part

---





# Vitis HLS Design Flow

Create Platform from .xsa

Create HelloWorld.c application and modify code to the following

## Part 1

```
#include "platform.h"
#include "xbasic_types.h"
#include "xparameters.h" // Contains definitions for all peripherals
#include "xexample.h" // Contains hls example (axilite) IP macros and functions

// Define global values for HLS example IP
XExample do_hls_example;
XExample_Config *do_hls_example_cfg;
```

# Vitis HLS Design Flow

Create Platform from .xsa

Create HelloWorld.c application and modify code to the following

## Part 2

```
// Initialize the HLS example IP
void init_HLS_example(){

    int status;
    // Create HLS example IP pointer
    do_hls_example_cfg = XExample_LookupConfig(
        XPAR_XEXAMPLE_0_DEVICE_ID);

    if (!do_hls_example_cfg) {
        xil_printf(
            "Error loading configuration for do_hls_example_cfg \n\r");
    }

    status = XExample_CfgInitialize(&do_hls_example,
        do_hls_example_cfg);
    if (status != XST_SUCCESS) {
        xil_printf("Error initializing for do_hls_example \n\r");
    }

    XExample_Initialize(&do_hls_example,
        XPAR_XEXAMPLE_0_DEVICE_ID);
}
```

# Vitis HLS Design Flow

Create Platform from .xsa

Create HelloWorld.c application and modify code to the following

## Part 3

```
// Function that adds using HLS example IP
// The functions used here are defined in xexample.h
void example_hls(int a, int b) {
    unsigned int c;
    c = 0; // result output from HLS IP

    // Write inputs
    XExample_Set_a(&do_hls_example, a);
    XExample_Set_b(&do_hls_example, b);
    xil_printf("Write a: %d \n\r", a);
    xil_printf("Write b: %d \n\r", b);

    // Start HLS IP
    XExample_Start(&do_hls_example);
    xil_printf("Started HLS Example IP \n\r");

    // Wait until it is finished
    while (!XExample_IsDone(&do_hls_example))
        ;

    // Get hls_multiplier returned value
    c = XExample_Get_return(&do_hls_example);

    xil_printf("HLS IP Return Value: %d\n\r", c);
    xil_printf("End of test\n\n\r");
}
```

# Vitis HLS Design Flow

Create Platform from .xsa

Create HelloWorld.c application and modify code to the following

## Part 4

```
int main() {
    // setup
    init_platform();
    init_HLS_example();
    int a = 0;
    int b = 0;

    while (1) {
        // Enter the HLS IP inputs - a and b (Defined in Vitis HLS)
        xil_printf("Enter value for A: ");
        scanf("%d", &a);
        xil_printf("%d\n\r", a);
        xil_printf("Enter value for B: ");
        scanf("%d", &b);
        xil_printf("%d\n\r", b);

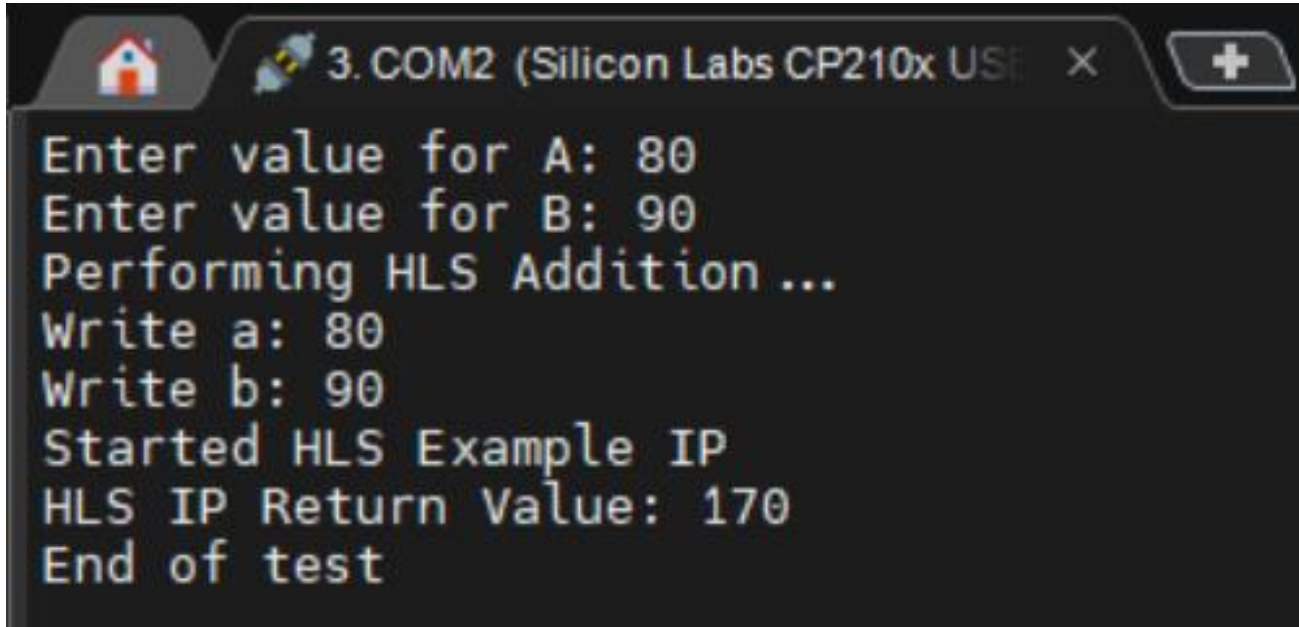
        xil_printf("Performing HLS Addition... \n\r");

        // perform addition in HLS IP
        example_hls(a, b);
    }

    cleanup_platform();
    return 0;
}
```

# Vitis HLS Design Flow

## Build and Run --- Result



```
3. COM2 (Silicon Labs CP210x USB) X  
Enter value for A: 80  
Enter value for B: 90  
Performing HLS Addition...  
Write a: 80  
Write b: 90  
Started HLS Example IP  
HLS IP Return Value: 170  
End of test
```

# Reference

## English:

1. [AXI Basics 6 - Introduction to AXI4-Lite in Vitis HLS \(xilinx.com\)](#)
2. [AXI Basics 7 - Connecting to the PS using AXI4-Lite and Vitis HLS \(xilinx.com\)](#)

## Chinese:

1. [AXI 基础第 6 讲 - Vitis HLS 中的 AXI4-Lite 简介 \(第 1 部分\) \(xilinx.com\)](#)
2. [AXI 基础第 7 讲 - 使用 AXI4-Lite 将 Vitis HLS 创建的 IP 连接到 PS \(xilinx.com\)](#)
3. [Vitis HLS 加法器\(整数\)设计\\_hls print-CSDN博客](#)

