

Courses @ NECST

Xilinx Vivado HLS
18/01/2018

Lorenzo Di Tucci <lorenzo.ditucci@polimi.it>
Emanuele Del Sozzo <emanuele.delsozzo@polimi.it>
Marco D. Santambrogio <marco.santambrogio@polimi.it>

Agenda

- Introduction to the Hardware Design Flow
- Vivado HLS:
 - Design Flow
 - Kernel Creation
 - Communication Infrastructure
 - Kernel Optimizations
- Hands-On Example

Installation Party

Use this Google Doc to provide your data

<https://goo.gl/FRCG6y>

First, install the VPN we have provided you.
(Mac: Tunnelblick - Windows/Linux: OpenVPN)

To **SSH** to the machine:

*ssh <name>.<surname>@nags31.local.necst.it
password: user*

Installation Party

You can change your password here:

<http://changepassword.local.necst.it/>

You can also **RDP** to the instance using

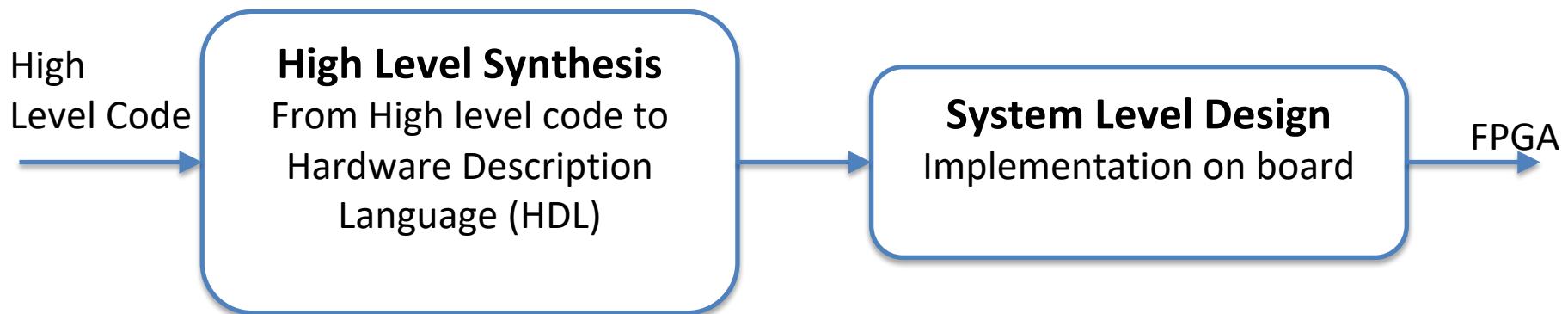
- Microsoft Remote Desktop (Microsoft/Mac OS)
- Remmina (Linux)

To connect to the machine, or change your password you **must** have started the VPN.

Hardware Design Flow for HPC

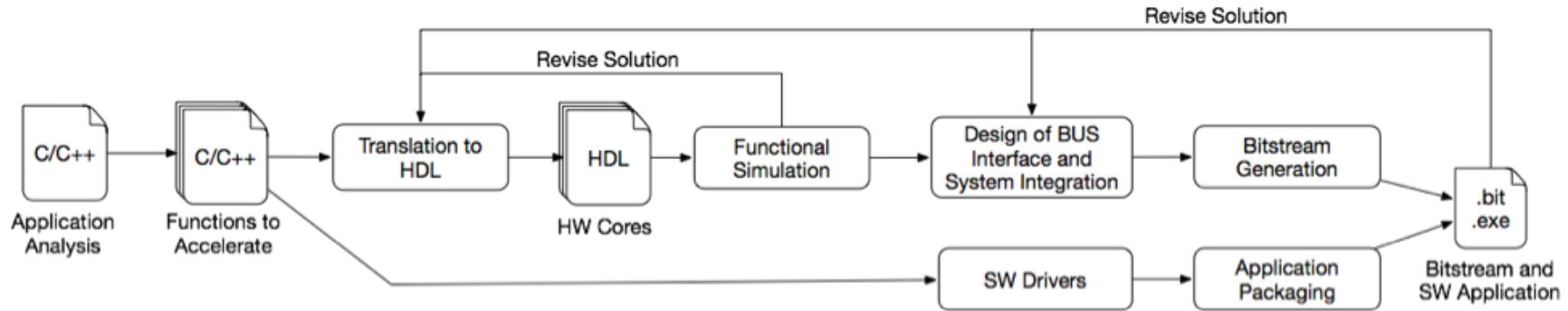
5

- *Hardware Design Flow (HDF)*: process to realize a hardware module
- HDF for FPGAs can be seen as a 2 step process



The Hardware Design Flow

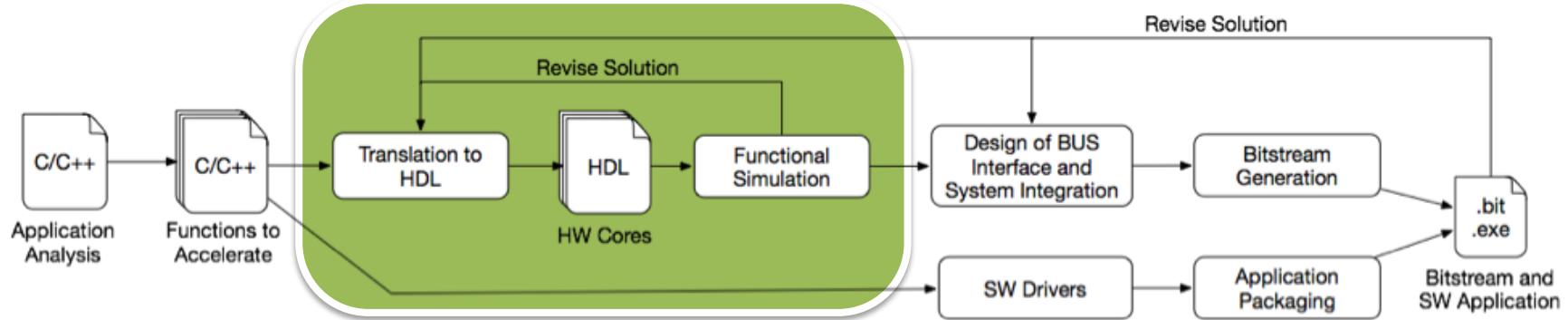
6



The Hardware Design Flow

7

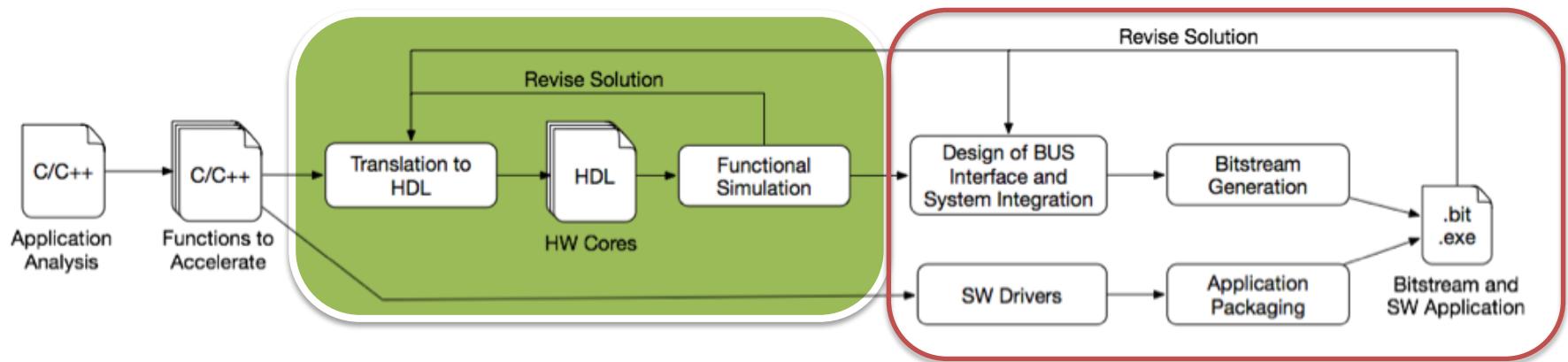
- CAD tools aims at alleviating the design on FPGA



The Hardware Design Flow

8

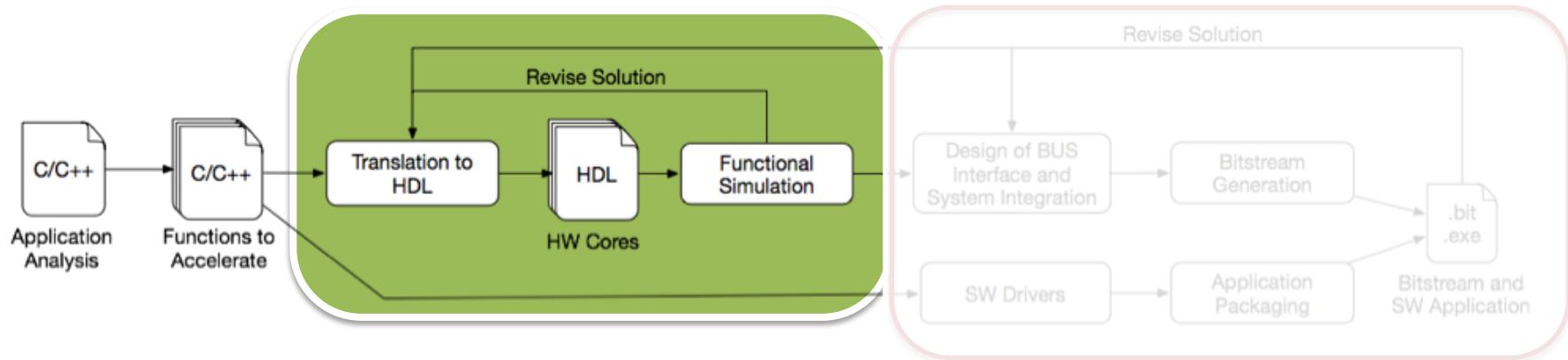
- CAD tools aims at alleviating the design on FPGA



- Designer still need to manually perform system integration, driver generation and runtime management

The Hardware Design Flow

- CAD tools aims at alleviating the design on FPGA

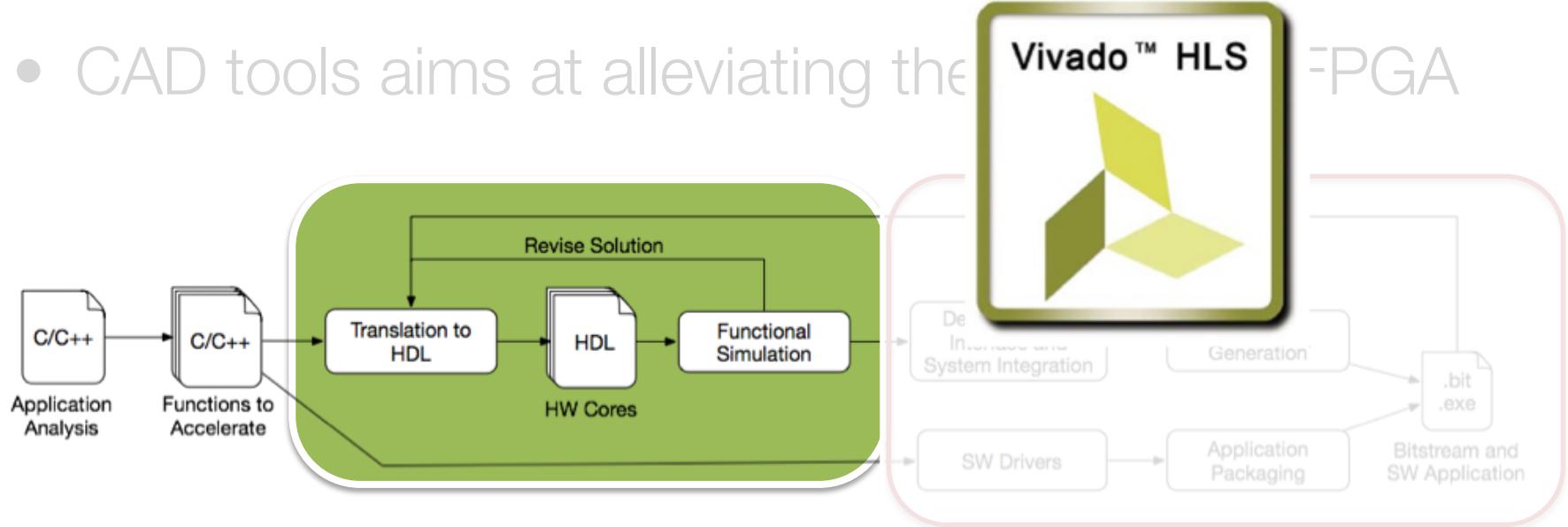


- Designer still need to manually perform system integration, driver generation and runtime management

The Hardware Design Flow

10

- CAD tools aims at alleviating the



- Designer still need to manually perform system integration, driver generation and runtime management

- High Level Synthesis tools
- provided a high level description of a kernel, allows to generate a HDL of the application
- increase the level of **abstraction** to ease the designing process
- provides multiple **hardware optimized** libraries and APIs
- **directives driven architecture-aware** synthesis



Vivado HLS Flow

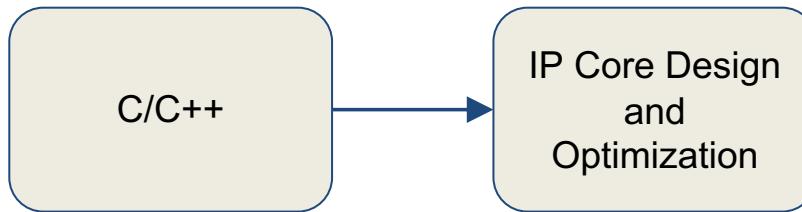
C/C++

Possibility to specify the **kernel** using High Level Code

- code must be refactored as a **subset of C/C++** is supported
- **no dynamic memory allocation**
- Multiple **API** provided (arbitrary precision, mathematical functions, video library, etc..)
- Possibility to specify a constraint file (clock uncertainty, clock period, target board)



Vivado HLS Flow

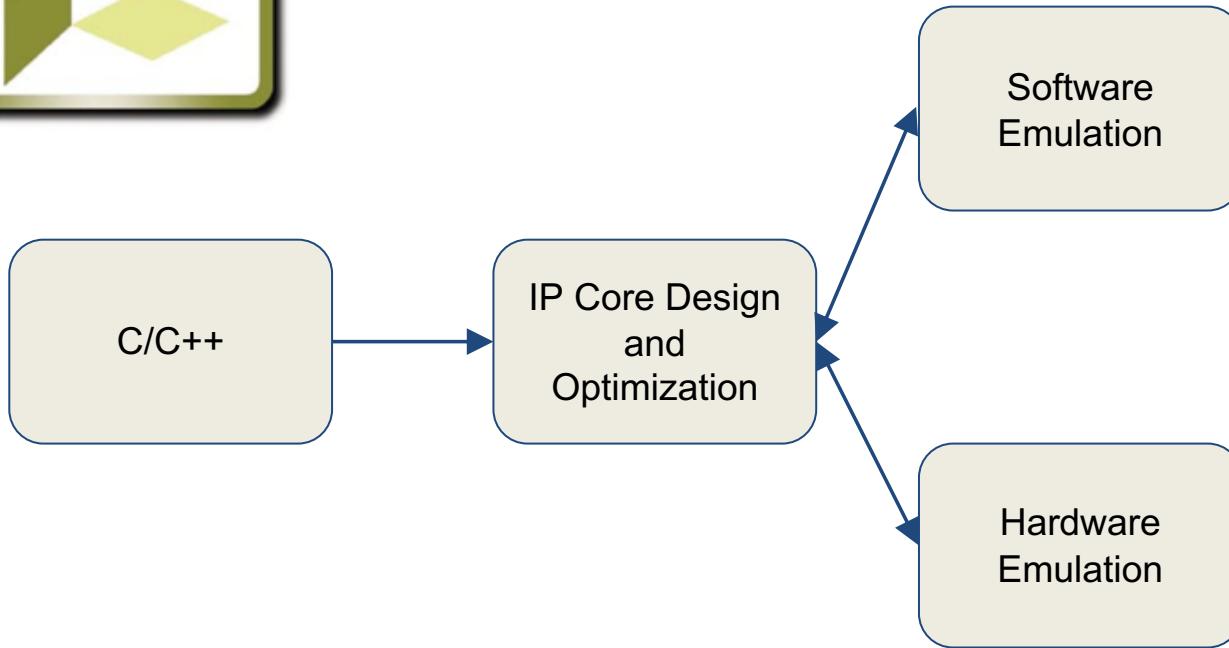


Directives driven design and optimization

- necessary to specify interconnection with memory (AXI4, AXI4Lite, AXI4-Stream)
- The designer still needs to have clear in mind what architecture to implement



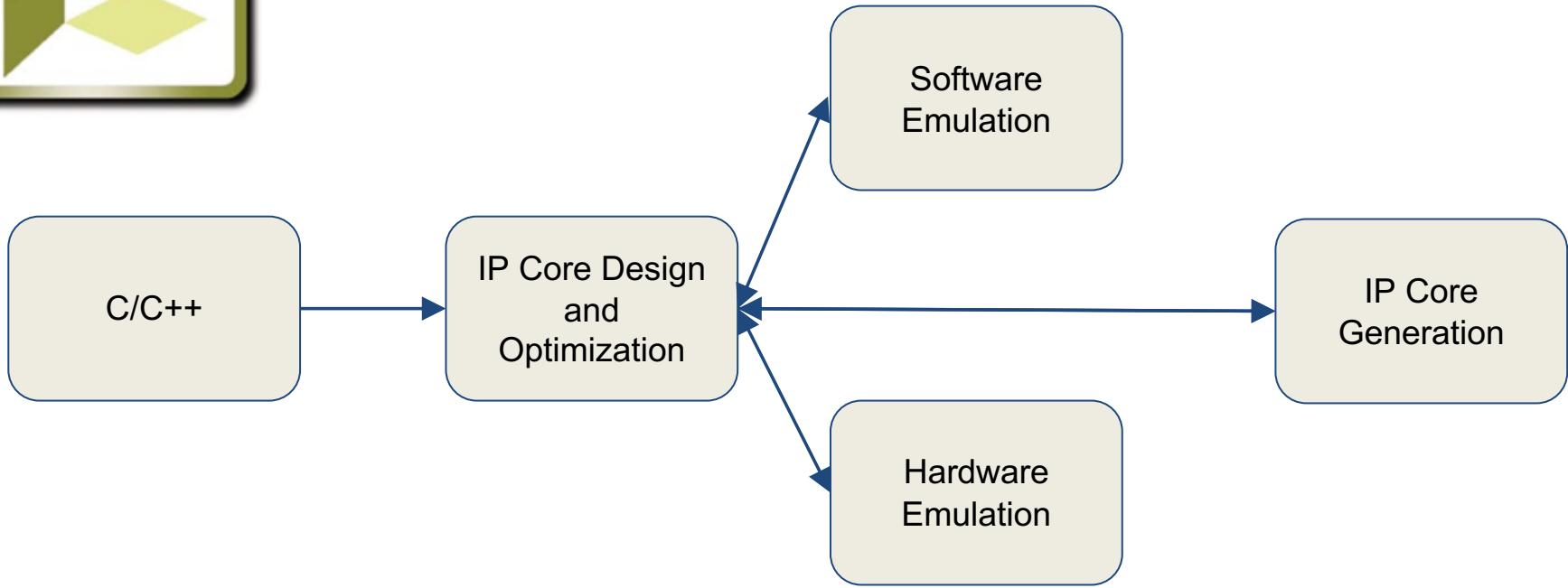
Vivado HLS Flow



2 levels of emulation

- **Software Emulation** used to check *functional correctness* of the application - does not guarantee correctness on the FPGA
- **Hardware Emulation** check the correctness of the logic generated by the synthesizer

Vivado HLS Flow



- Eventually, IP Core can be generated and exported so that it can be used for the **System Level Design Step**.
- At each step of the pipeline, it is possible to come back to the optimization phase

Kernel Creation

- Not all the parts of an algorithm are suitable for hardware acceleration
- First, it is necessary to **profile** the application to identify the bottlenecks of the application, i.e. the most compute intensive parts/kernels
- Then, it is possible to start adapting and optimizing the kernels for the High Level Synthesis process
 - define the communication infrastructure
 - build the kernel architecture

Communication Infrastructure

17

- Mainly, 3 types of communication may be implemented using AXI4 protocol:
 - AXI-Lite (for control signals)
 - Axi-Stream (to stream data)
 - Axi-Master (direct connection with the memory)
- Vivado HLS provides directives to specify which type of communication to implement
- However, the user has to develop the data transfer in an **efficient** way

Kernel Optimizations

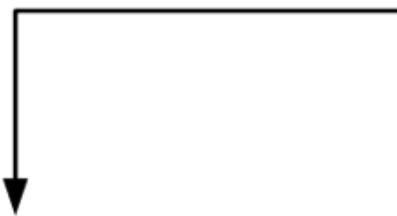
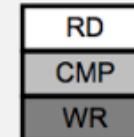
18

- Vivado HLS provides directives to optimize the kernel
- The directives may refer to:
 - how to carry out the computation
 - loop pipelining/unrolling
 - function inlining
 - dataflow
 - resources to use for a certain computation
 - how to store the data on FPGA memory
 - array partitioning (block, cyclic, complete)

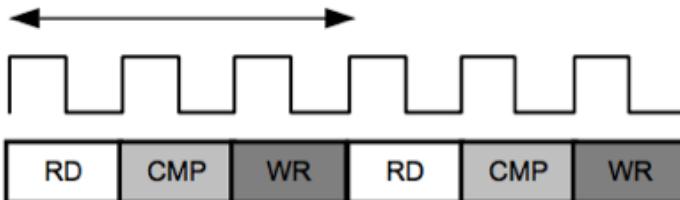
Loop Pipelining

Without Pipelining

```
Loop:for(i=1;i<3;i++) {
    op_Read;
    op_Compute;
    op_Write;
}
```



Initiation Interval = 3 cycles

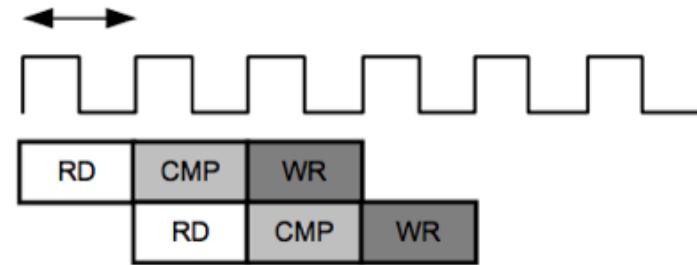


Latency = 3 cycles

Loop Latency = 6 cycles

With Pipelining

Initiation Interval = 1 cycle

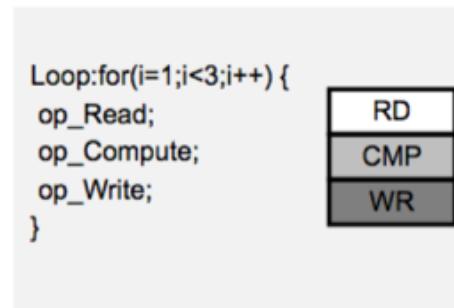


Latency = 3 cycles

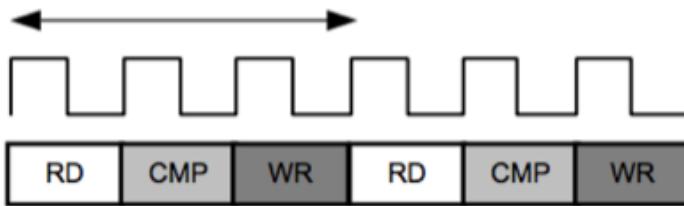
Loop Latency = 4 cycles

Loop Unrolling

Without Unrolling



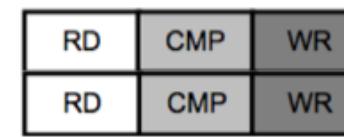
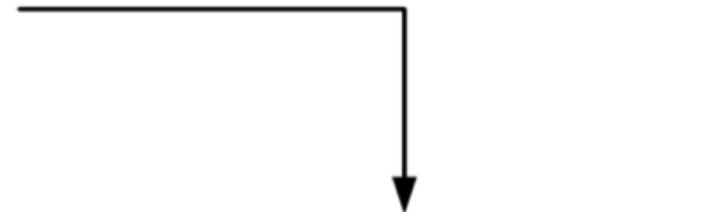
Initiation Interval = 3 cycles



Latency = 3 cycles

Loop Latency = 6 cycles

With Unrolling



Latency = 3 cycles

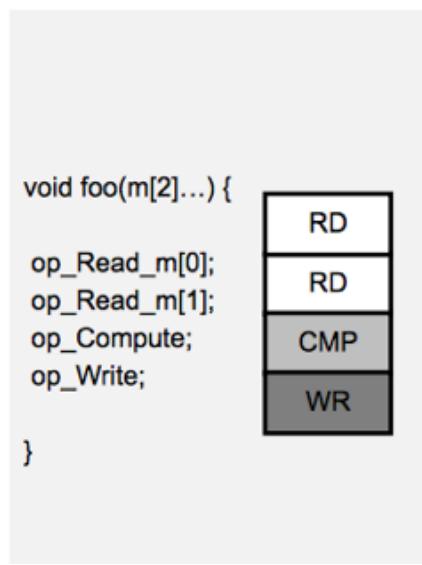
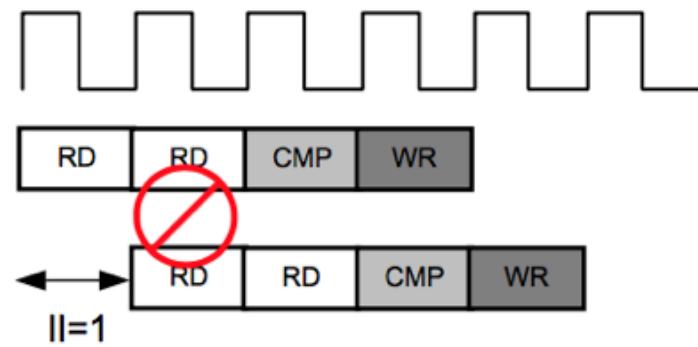
Loop Latency = 3 cycles



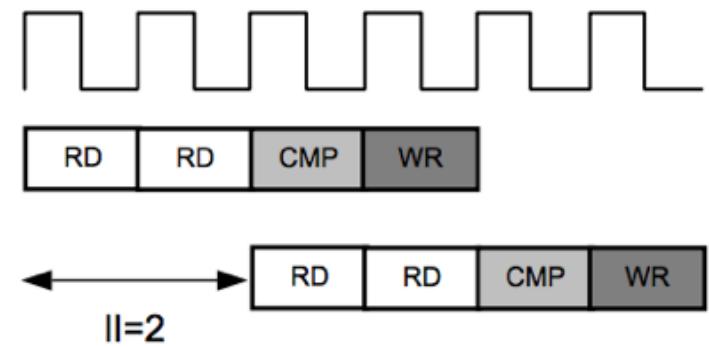
Resource Contention

21

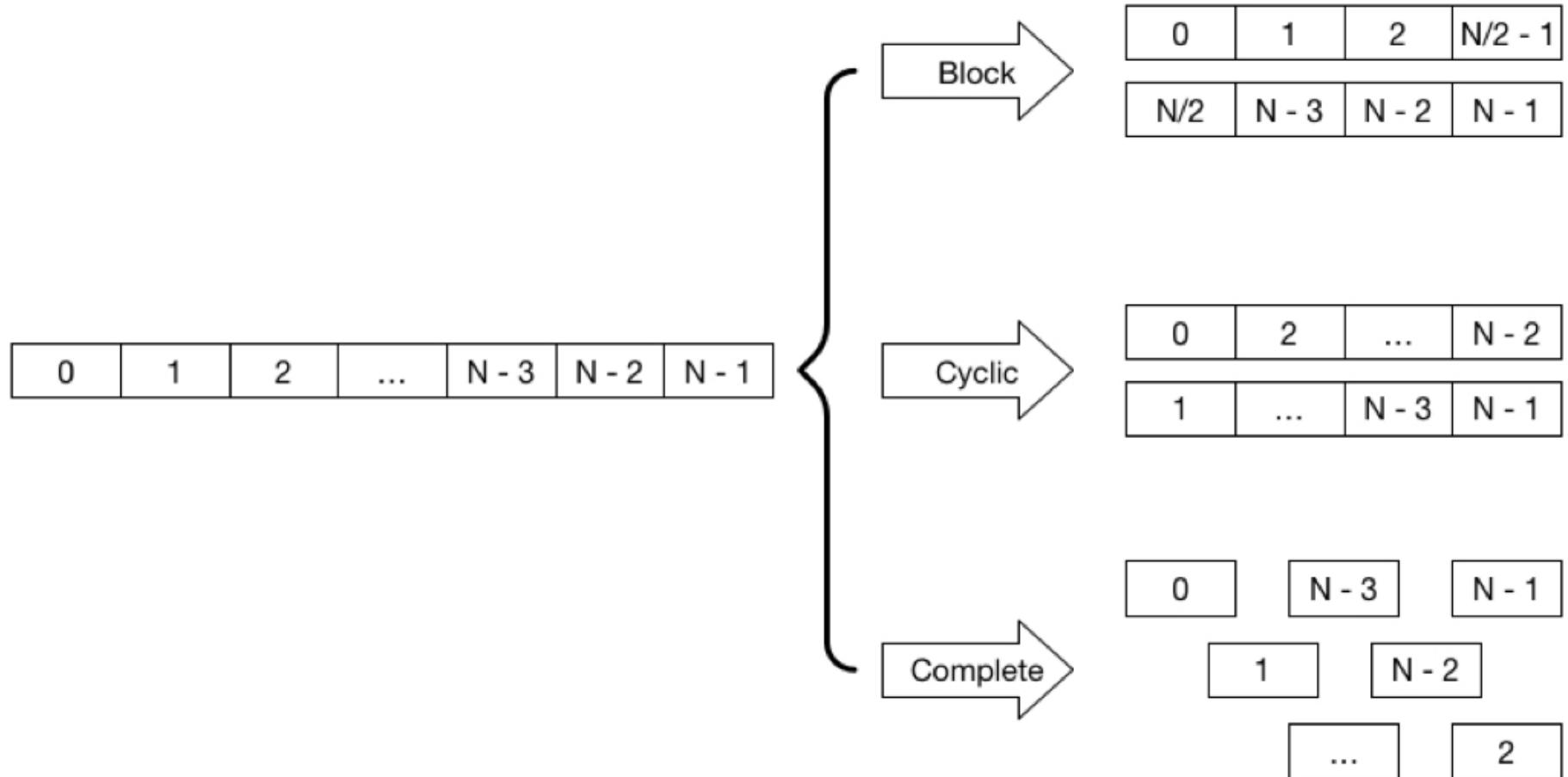
(A) Pipeline with II=1



(B) Pipeline with II=2



Array Partitioning



Example

- Now we are going to see how to implement a vector addition of this form:

$$\mathbf{a} = \mathbf{b} + \mathbf{c}^*d$$

where **a**, **b**, **c** are vectors, and d is a constant

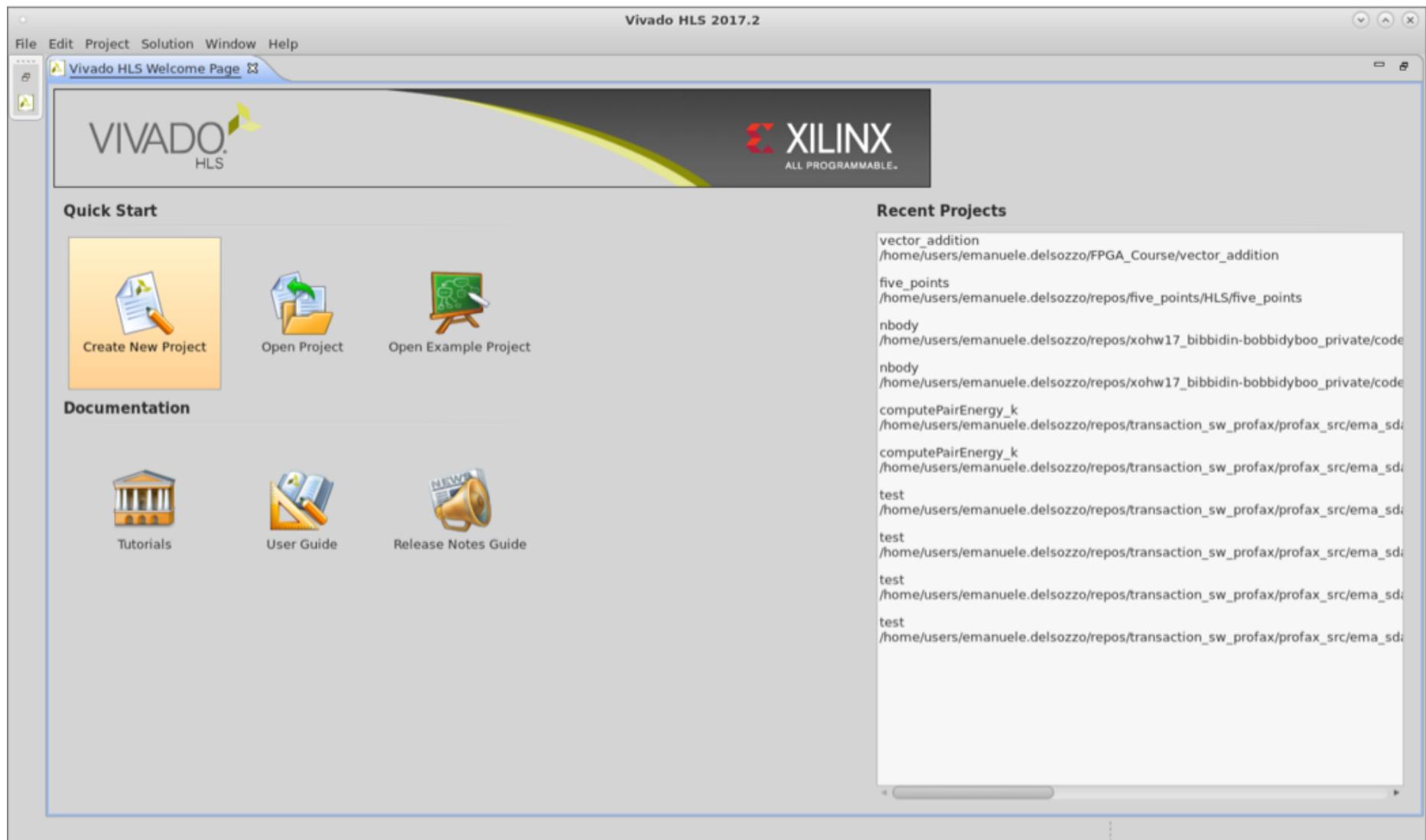
- We will go through all optimization steps with Vivado HLS, from the project creation to the IP core generation

Launch Vivado HLS

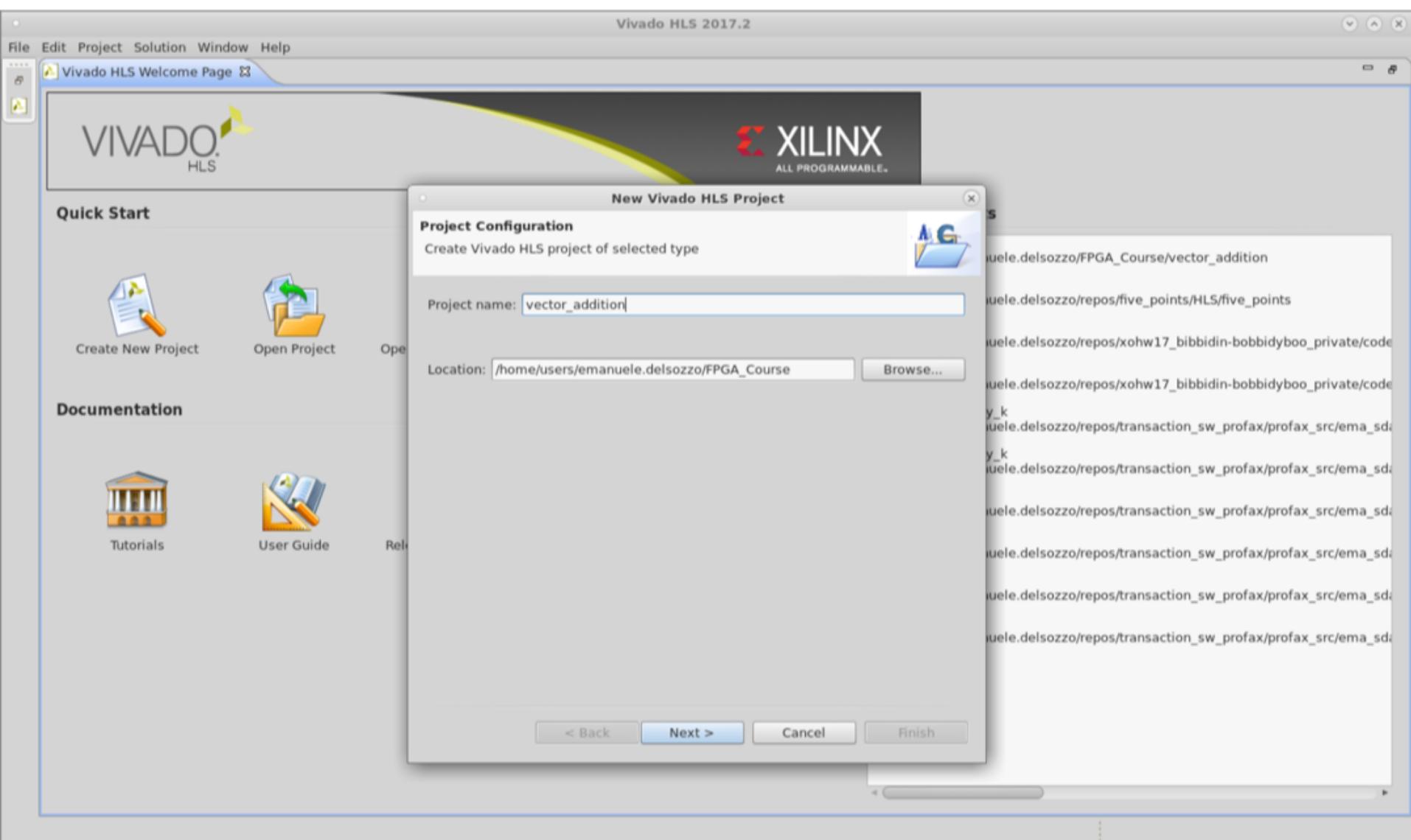
Source *settings64.sh* in Vivado folder, then run Vivado HLS

```
[emanuele.delsozzo@NAGS30 FPGA_Course]$ source /xilinx/software/Vivado/2017.2/settings64.sh
[emanuele.delsozzo@NAGS30 FPGA_Course]$ vivado_hls
=====
Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC
Version 2017.2
Build 1909853 on Thu Jun 15 18:55:24 MDT 2017
Copyright (c) 1986-2017 Xilinx, Inc. All Rights Reserved.
=====
INFO: [HLS 200-10] Running '/xilinx/software/Vivado_HLS/2017.2/bin/unwrapped/lnx64.o/vivado_hls'
INFO: [HLS 200-10] For user 'emanuele.delsozzo' on host 'NAGS30' (Linux_x86_64 version 3.10.0-514.
INFO: [HLS 200-10] On os "CentOS Linux release 7.3.1611 (Core) "
INFO: [HLS 200-10] In directory '/home/users/emanuele.delsozzo/FPGA_Course'
INFO: [HLS 200-10] Bringing up Vivado HLS GUI ...
```

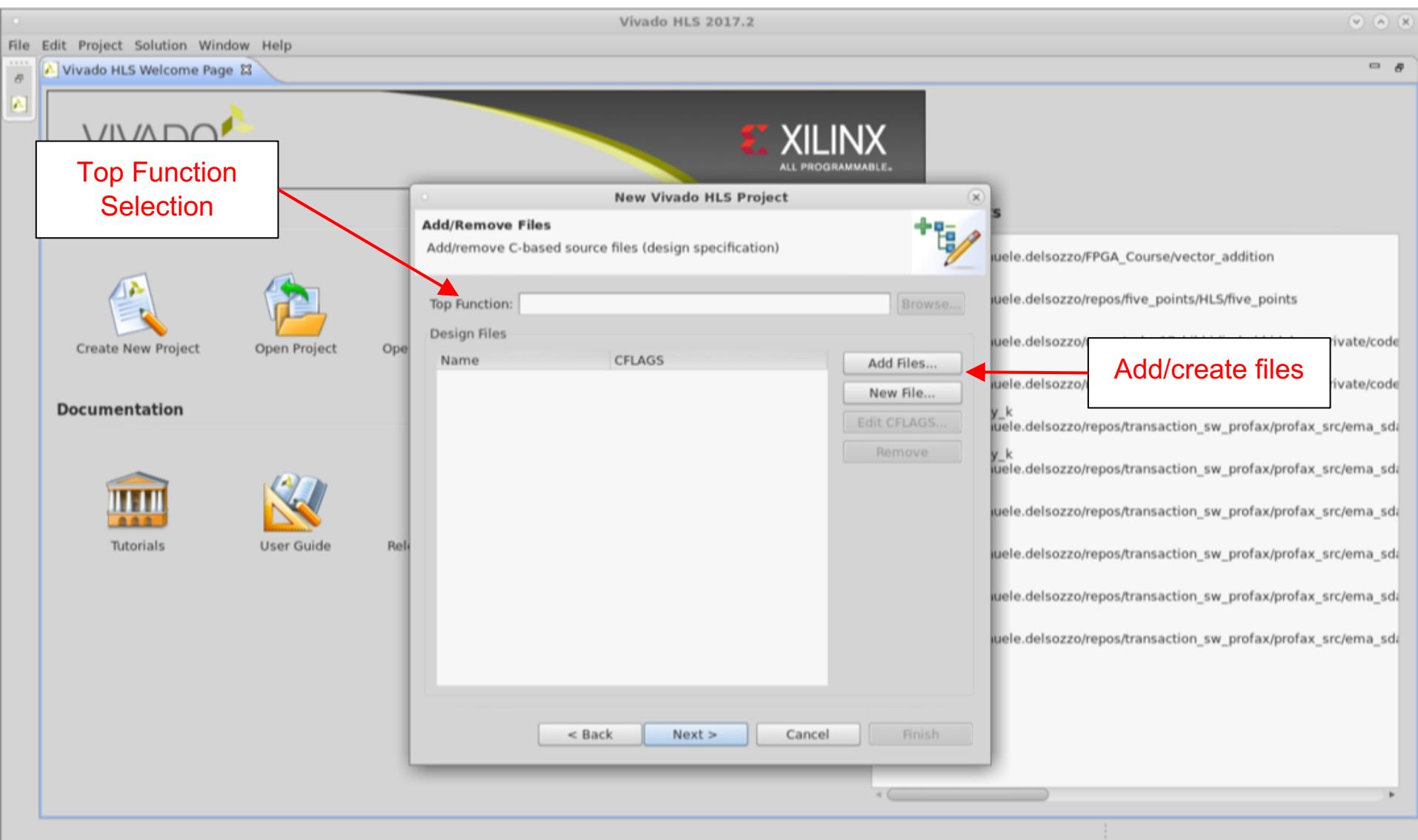
Vivado HLS GUI



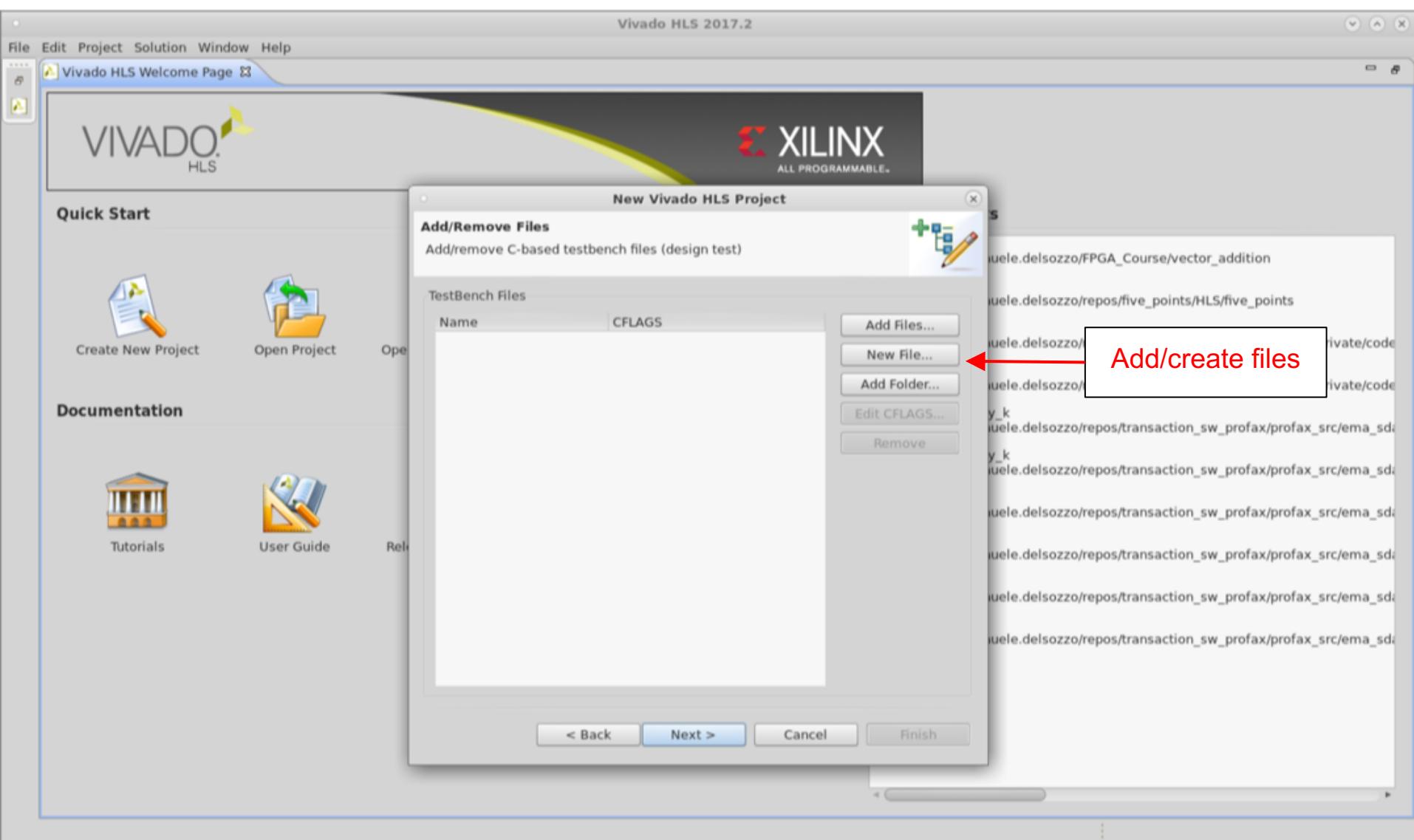
Project Configuration



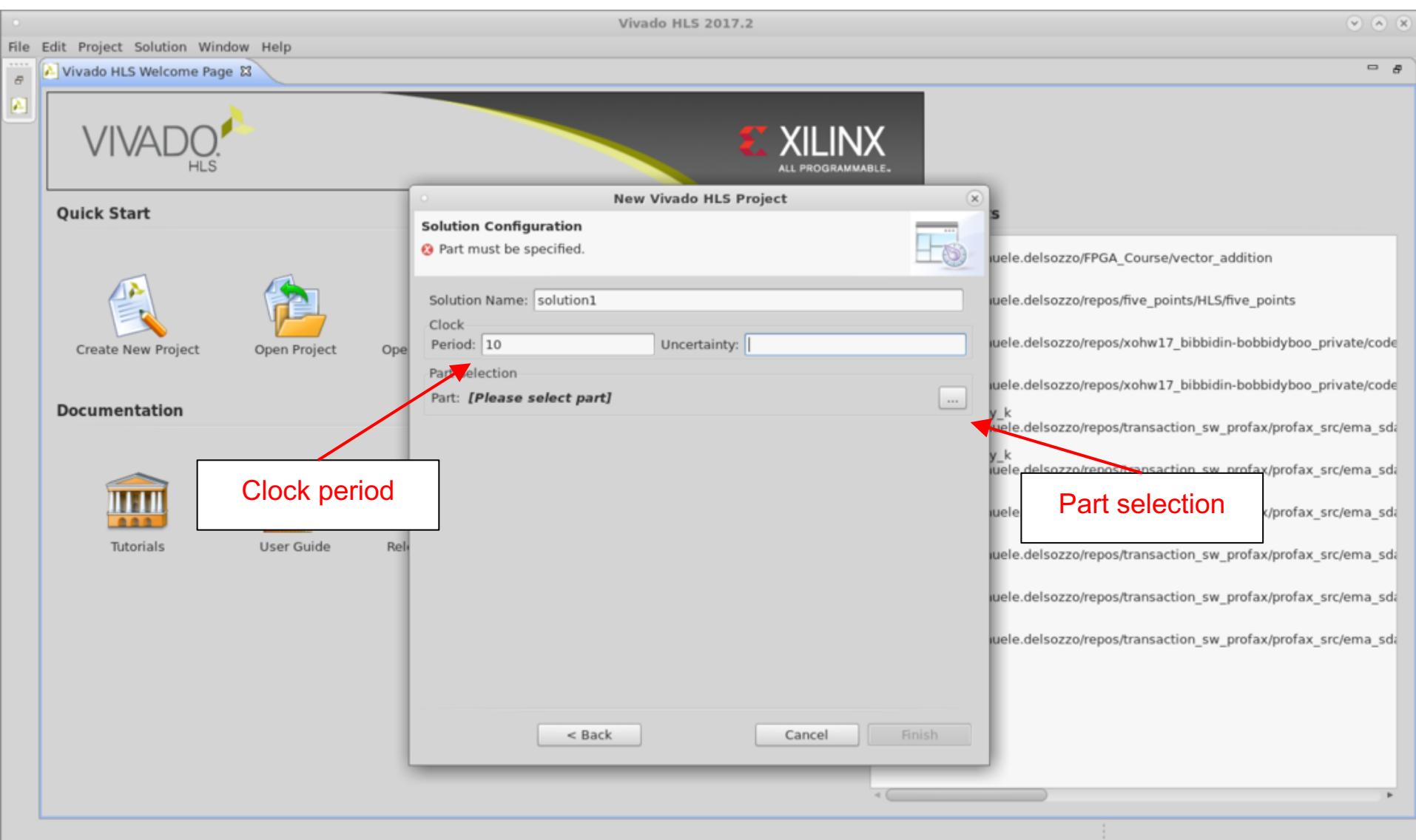
Kernel files



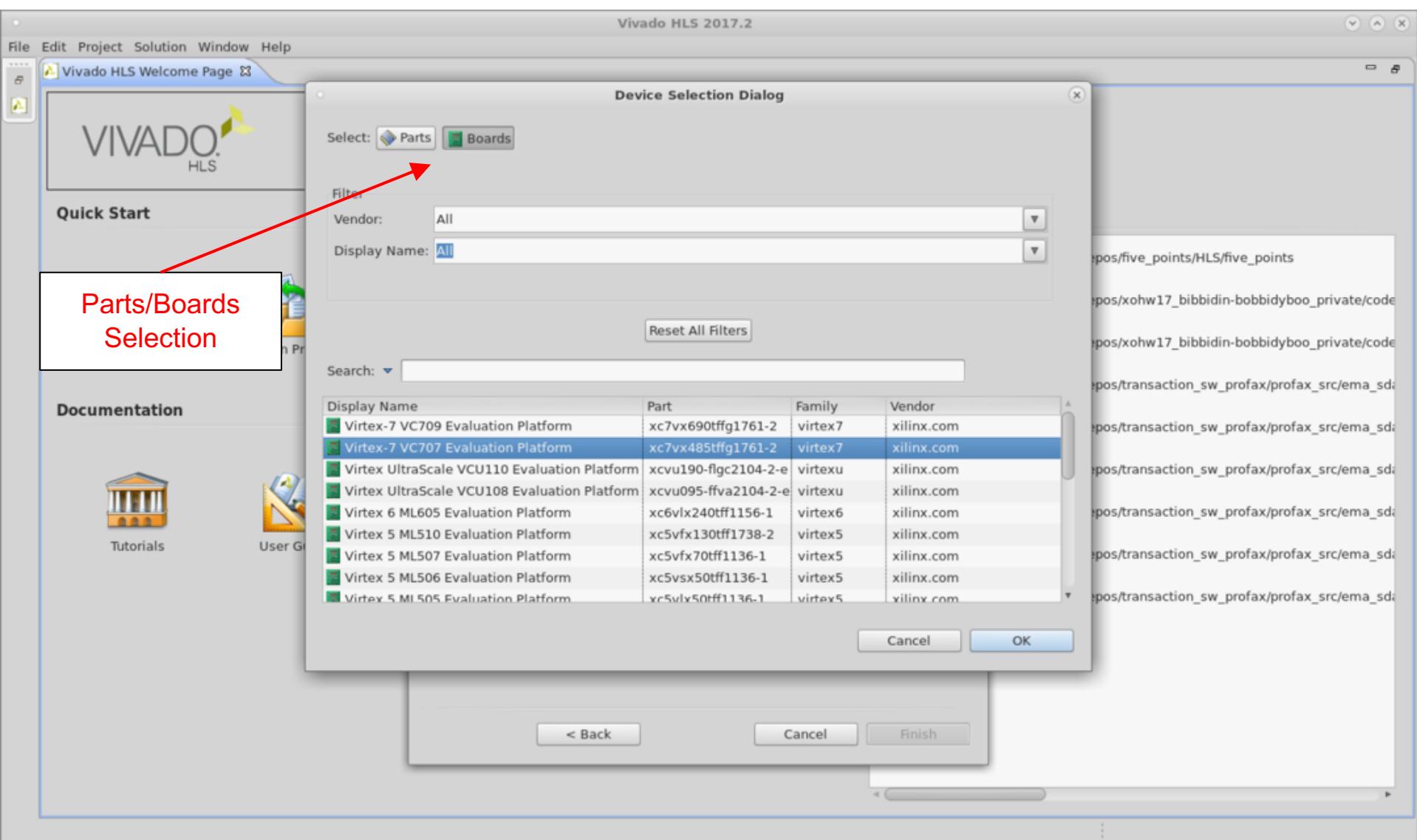
Testbench files



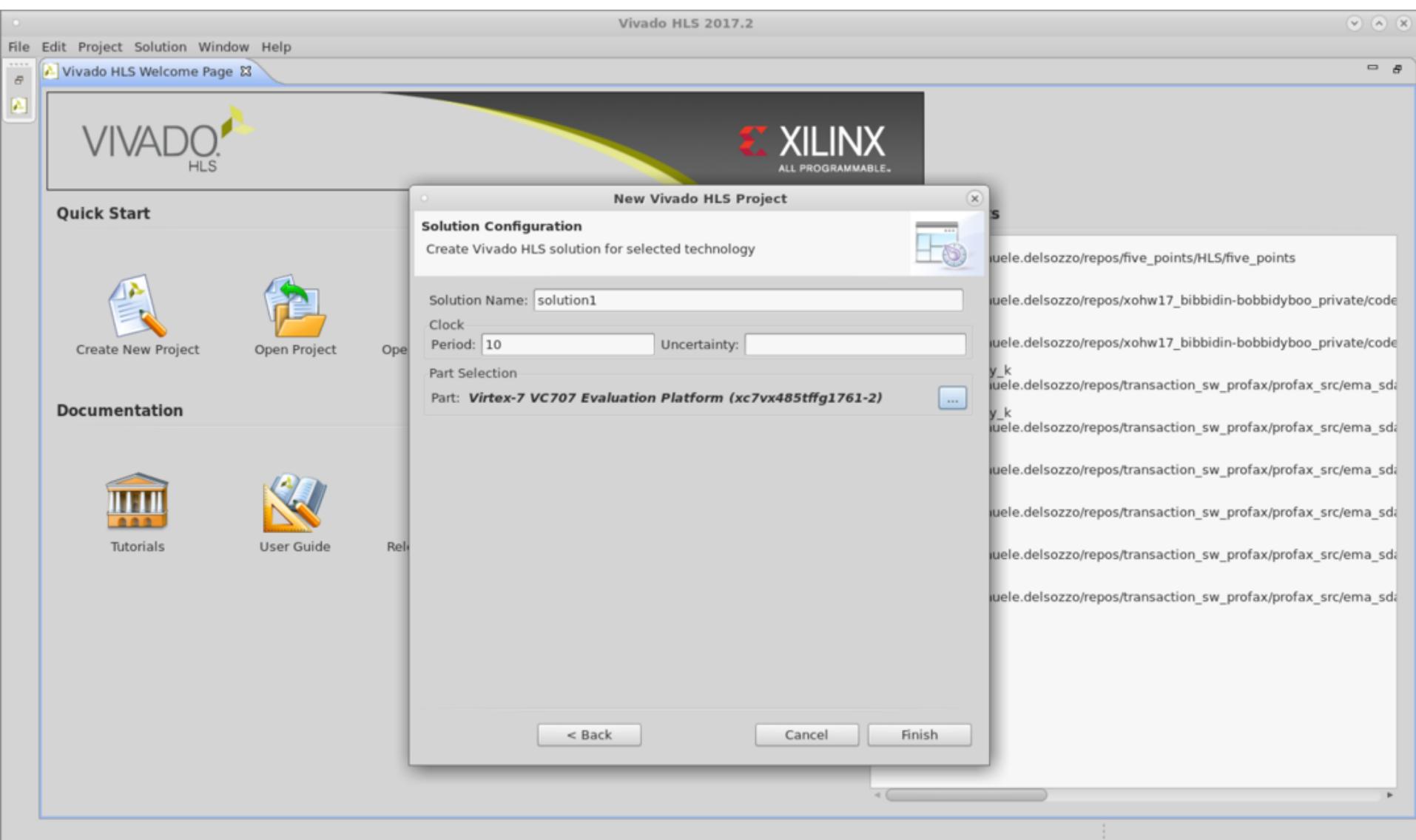
Solution Configuration



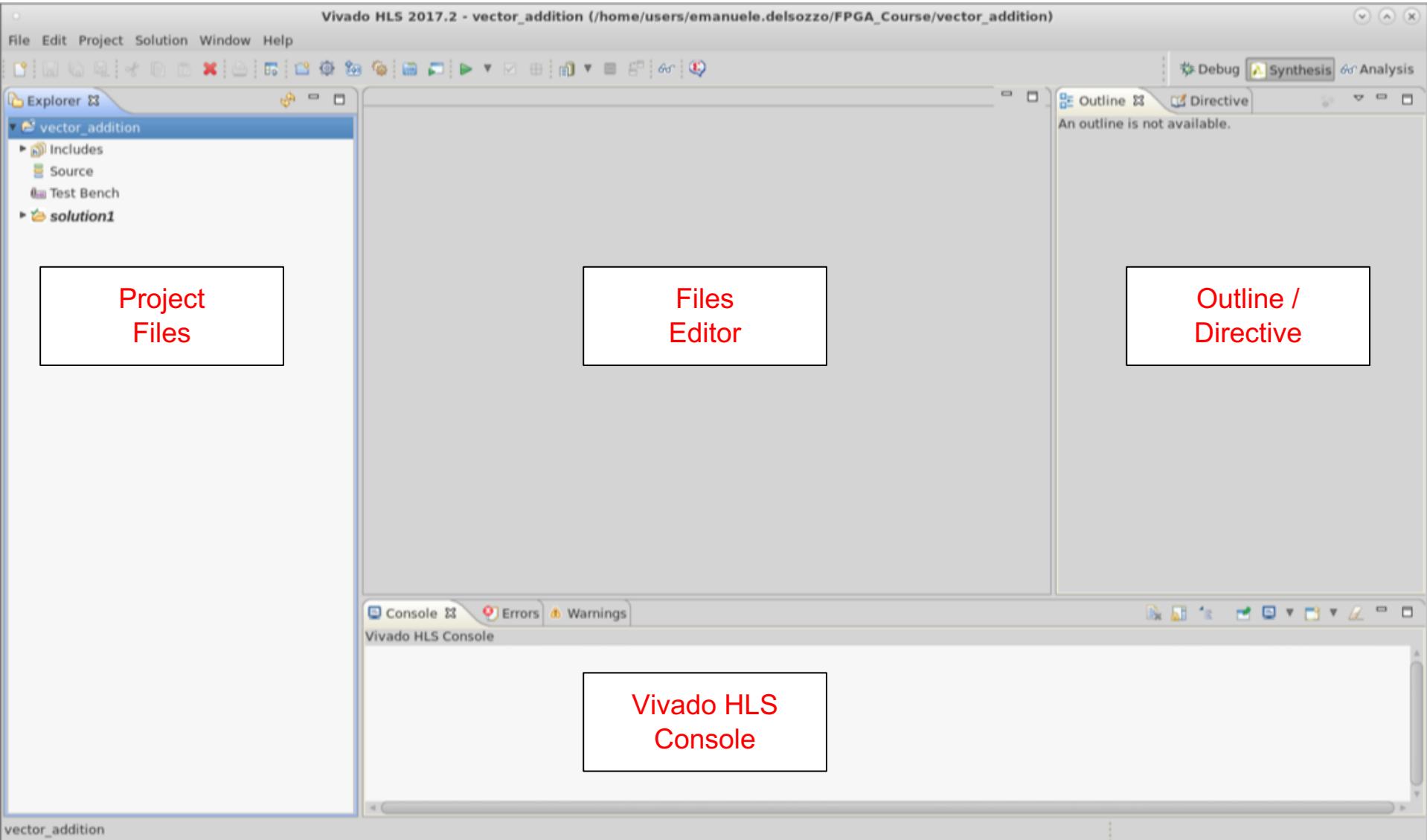
Device Selection



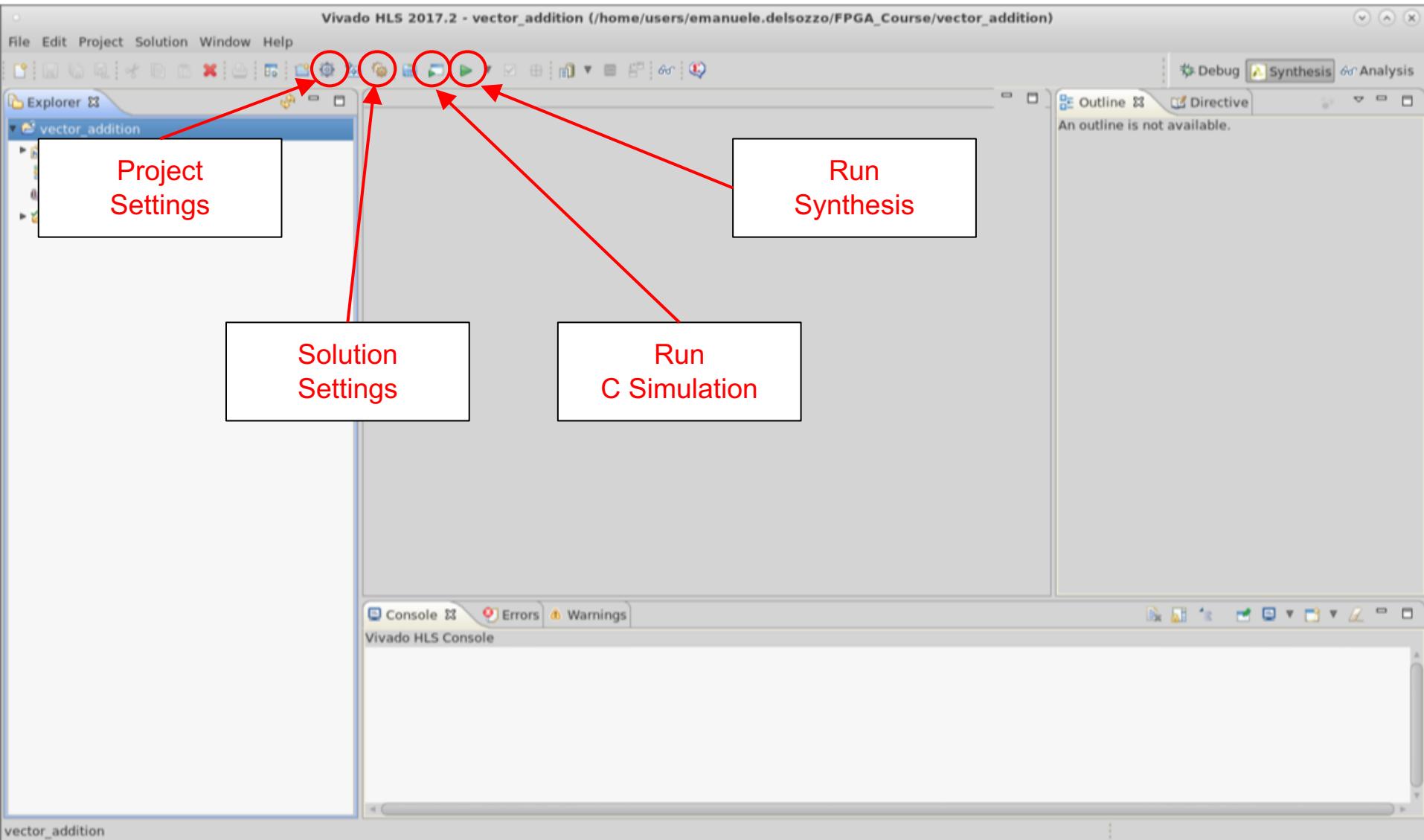
Solution Configuration



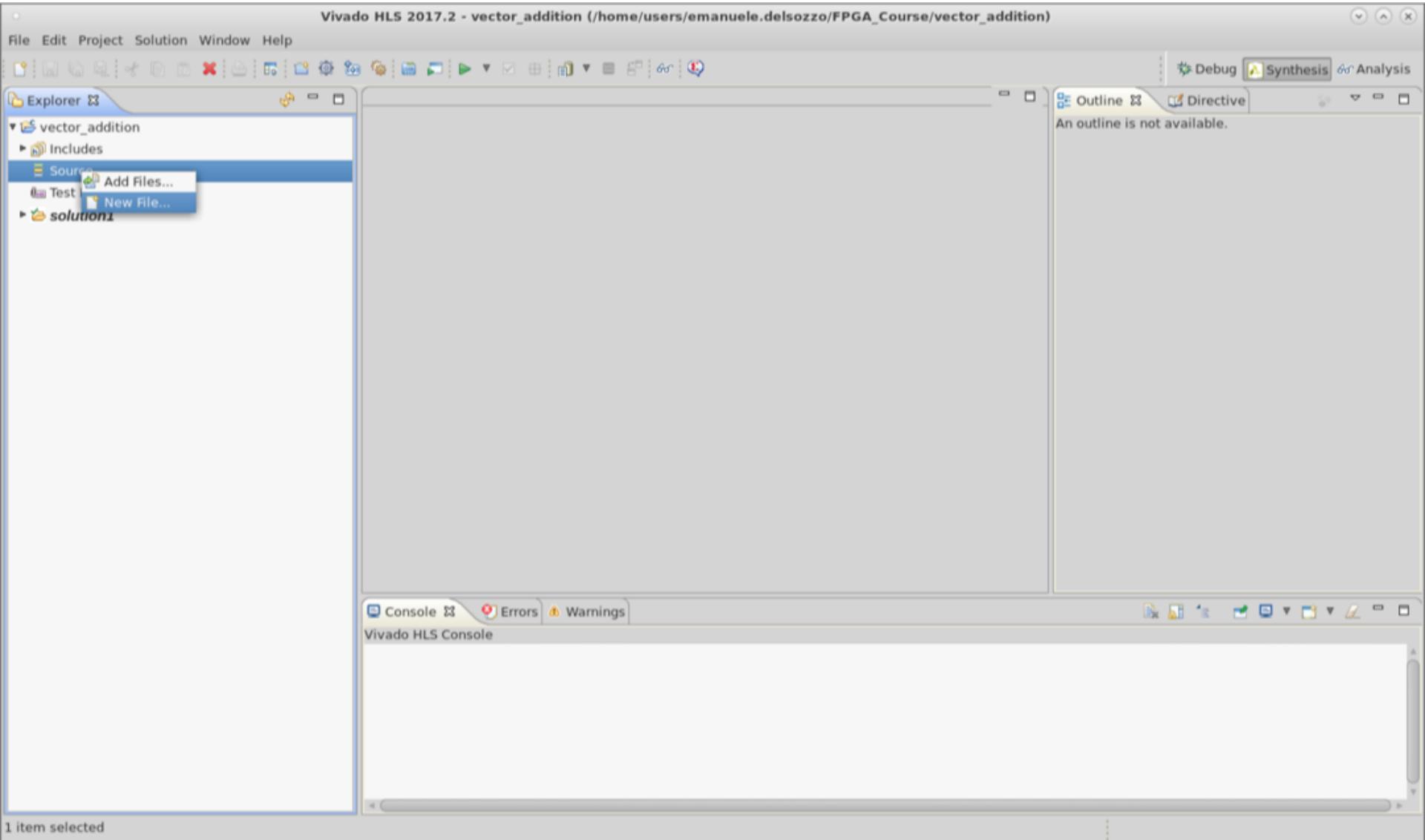
Editor Interface



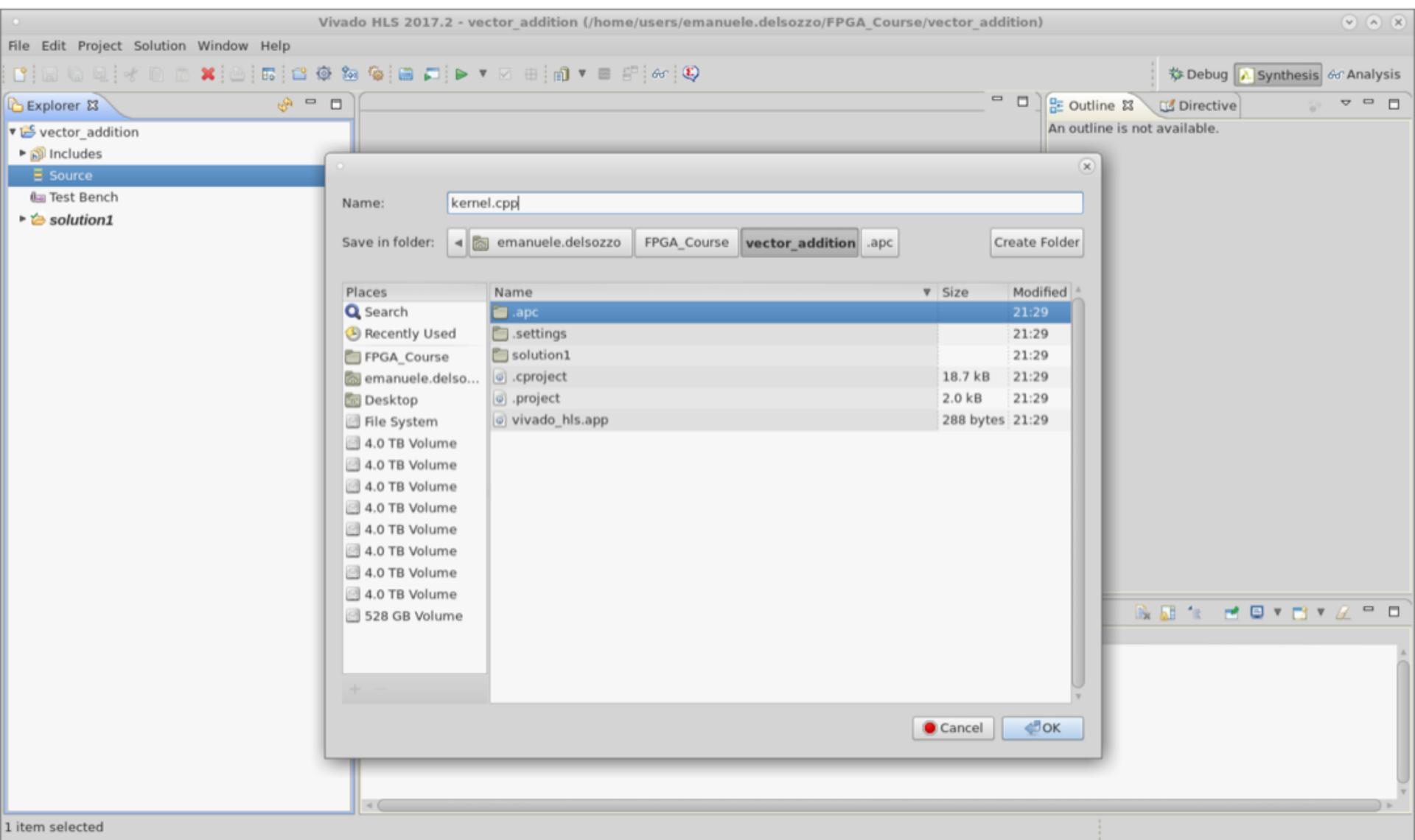
Editor Interface



Kernel Creation



Kernel Creation



Kernel function

The screenshot shows the Vivado HLS 2017.2 software interface. The title bar reads "Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)". The menu bar includes File, Edit, Project, Solution, Window, Help. The toolbar has various icons for file operations. The left sidebar is the "Explorer" view, showing a project named "vector_addition" with subfolders "Includes", "Source", "Test Bench", and a solution named "solution1". The main workspace contains a code editor window titled "kernel.cpp" with the following content:

```
1
2
3 void kernel(float *a, float *b, float *c, float d){
4
5
6
7 }
8
```

Below the code editor is a "Console" window titled "Vivado HLS Console" with tabs for "Console", "Errors", and "Warnings". The status bar at the bottom shows "Writable", "Smart Insert", and "5 : 1". The top right of the interface has tabs for "Debug", "Synthesis", and "Analysis".

Kernel function

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer □

- vector_addition
- Includes
- Source
- Test Bench
- solution1

kernel.cpp □

```
1
2
3 void kernel(float *a, float *b, float *c, float d){
4
5
6
7 }
8
```

Outline □ Directive □

- kernel(float*, float*, float*, float) : void

Let's add the directives for the communication interfaces

Console □ Errors □ Warnings □

Vivado HLS Console

Writable Smart Insert 5 : 1

Directives

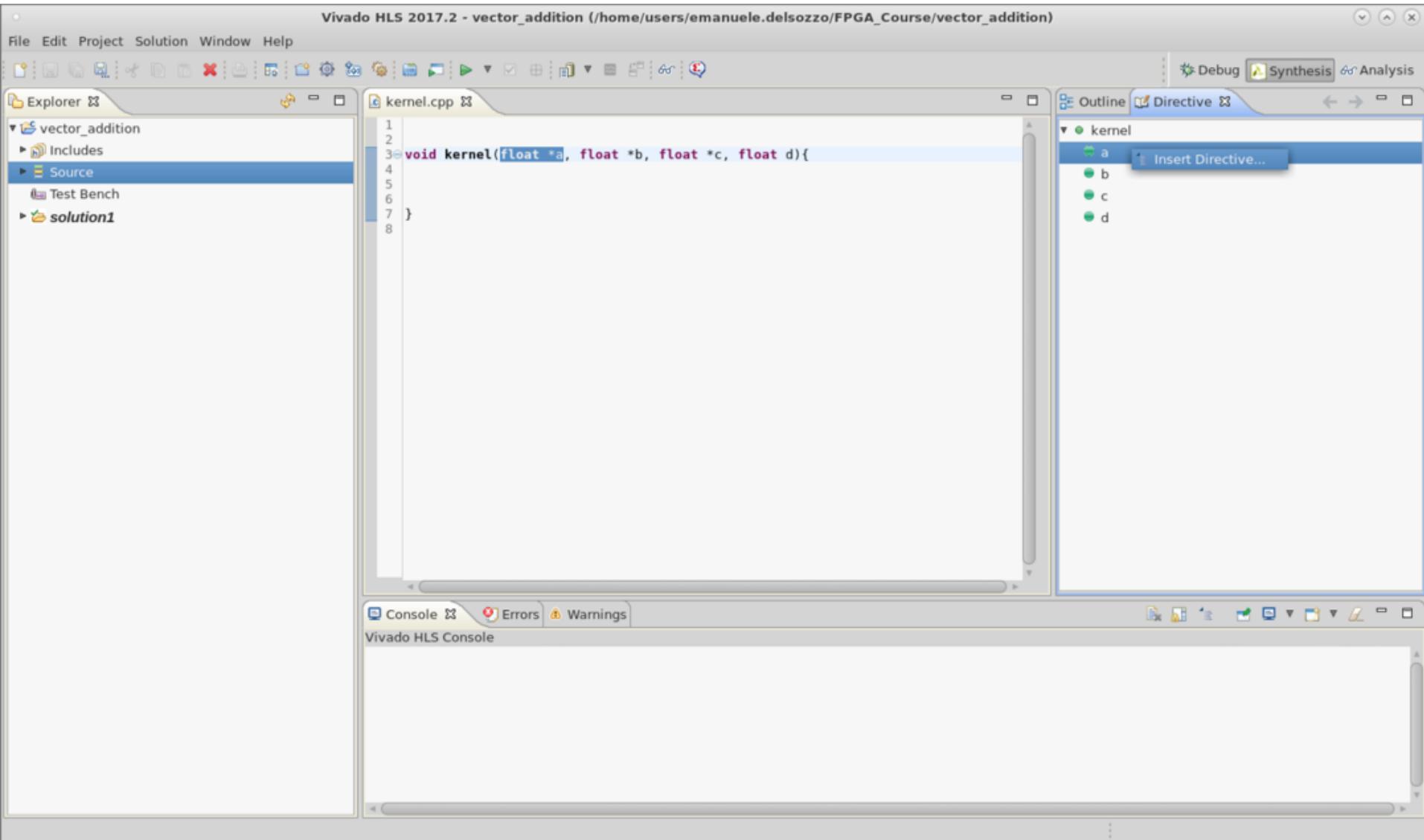
The screenshot shows the Vivado HLS 2017.2 interface with the following components:

- File Edit Project Solution Window Help**: Standard application menu.
- Explorer**: Shows the project structure with **vector_addition**, **Includes**, **Source** (selected), **Test Bench**, and **solution1**.
- kernel.cpp**: Source code editor showing the following C++ code:

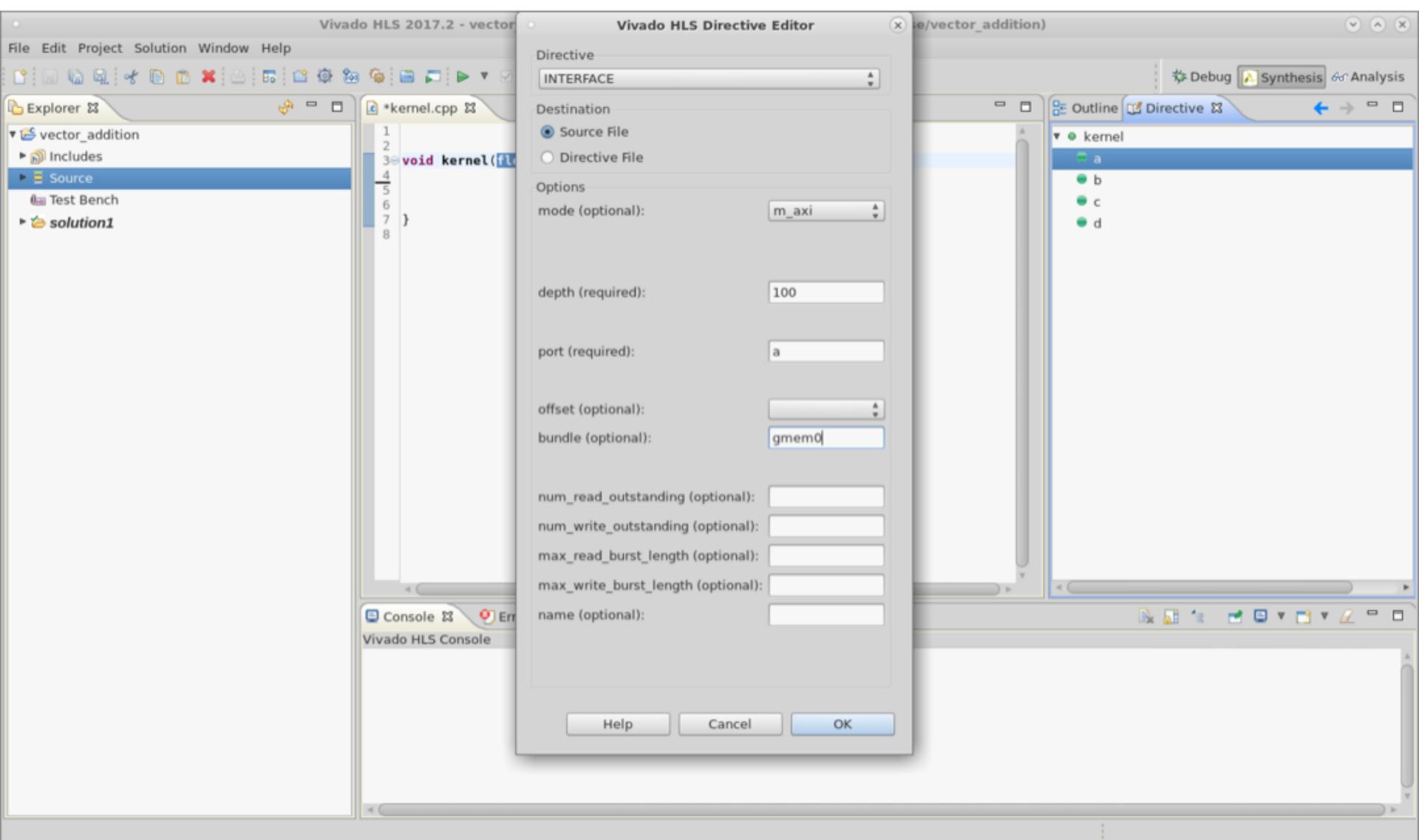
```
1
2
3 void kernel(float *a, float *b, float *c, float d){
4
5
6
7 }
```
- Outline**: Shows the hierarchical structure of the code, including **kernel**, **a**, **b**, **c**, and **d**.
- Directive**: A red box highlights this tab, indicating it is the current focus.
- Console**: Vivado HLS Console tab with **Errors** and **Warnings** sections.

Directives tab

Insert Directive for port a



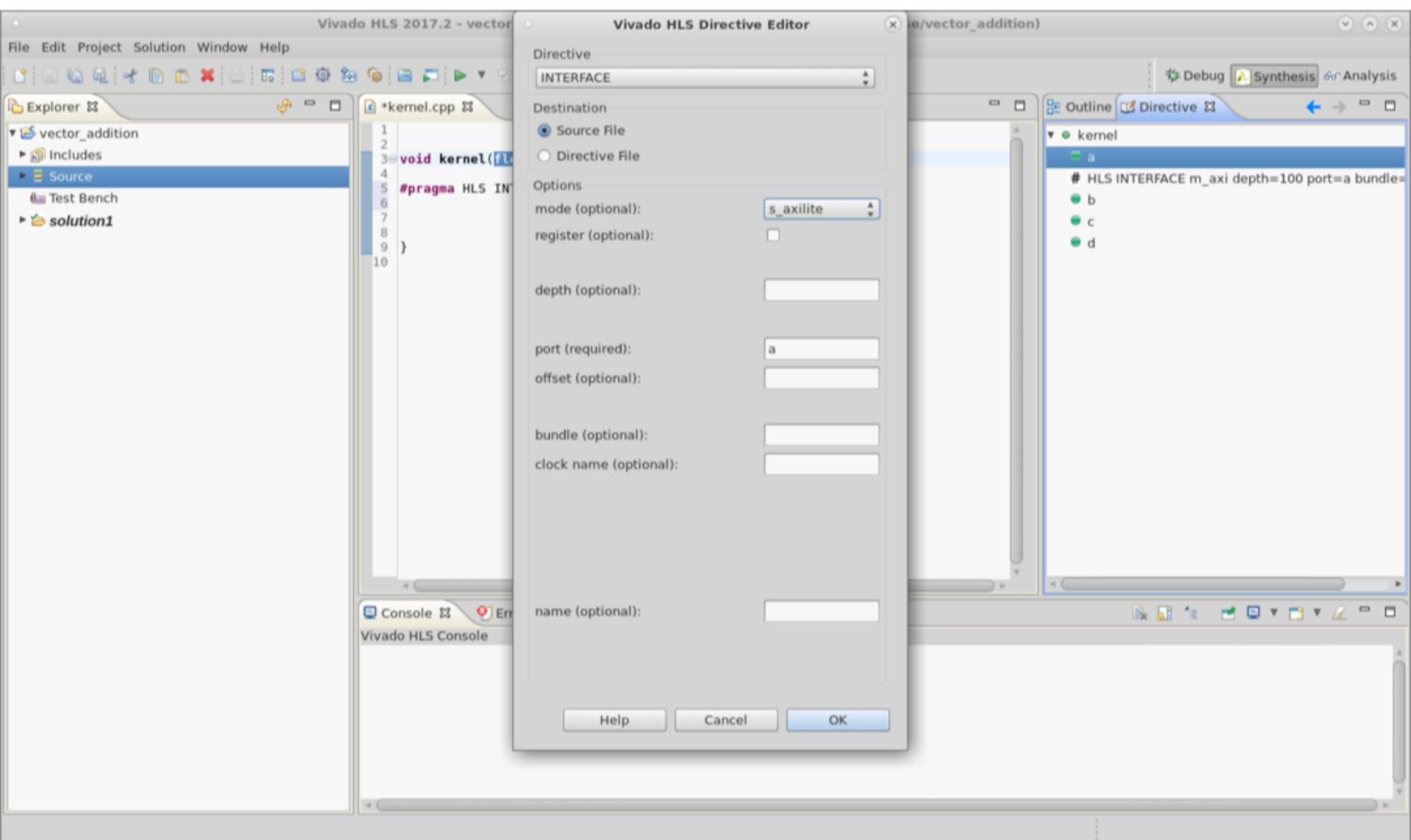
Master Axi Directive



Insert Directive for port a

The screenshot shows the Vivado HLS 2017.2 interface with the project "vector_addition" open. The "Source" tab in the Explorer panel is selected, displaying the file "kernel.cpp". The code contains a function definition:void kernel(float *a, float *b, float *c, float d){
#pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
}In the "Directive" tab of the Outline panel, there is a context menu for the variable "a" with the option "Insert Directive...". A tooltip for this option shows the directive "# HLS INTERFACE m_axi depth=100 port=a bundle=gmem0". The "Console" panel at the bottom is empty.

Axilite Directive



Port **a** directives

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer □

- vector_addition
- Includes
- Source
- Test Bench
- solution1

*kernel.cpp □

```
1
2
3 void kernel(float *a, float *b, float *c, float d){
4
5 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
6
7 #pragma HLS INTERFACE s_axilite port=a
8
9
10}
11
```

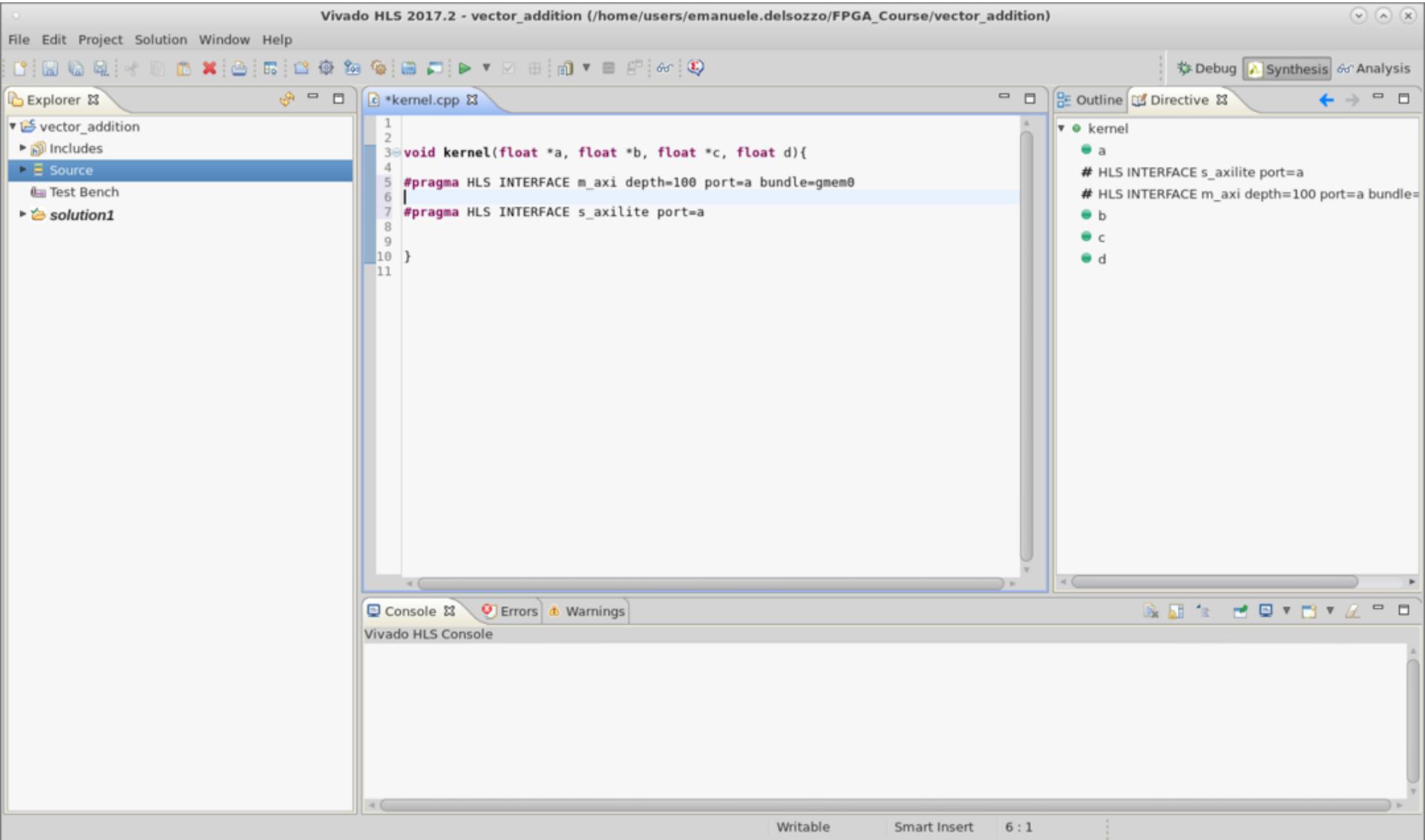
Console □ Errors Warnings

Vivado HLS Console

Writable Smart Insert 6 : 1

Outline Directive □

- kernel
 - a
 - # HLS INTERFACE s_axilite port=a
 - # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
 - b
 - c
 - d



Ports directives

The screenshot shows the Vivado HLS 2017.2 interface with the project "vector_addition". The "kernel.cpp" file is open in the center editor, displaying the following code:

```
1
2
3 void kernel(float *a, float *b, float *c, float d){
4
5 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
6 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
7 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
8
9 #pragma HLS INTERFACE s_axilite port=d
10 #pragma HLS INTERFACE s_axilite port=a
11 #pragma HLS INTERFACE s_axilite port=b
12 #pragma HLS INTERFACE s_axilite port=c
13
14
15 }
16
17 }
```

A red arrow points from a callout box to the line "#pragma HLS INTERFACE s_axilite port=d". The callout box contains the text:

Only axilite for port d
We pass it as a control signal

The right panel shows the "Directive" tab with a list of ports and their configurations:

- a: # HLS INTERFACE s_axilite port=a
- b: # HLS INTERFACE s_axilite port=a
- c: # HLS INTERFACE s_axilite port=a
- d: # HLS INTERFACE s_axilite port=a
- a: # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
- b: # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
- c: # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
- d: # HLS INTERFACE m_axi depth=100 port=d bundle=gmem3

Insert Directive for kernel

The screenshot shows the Vivado HLS 2017.2 interface with the project 'vector_addition' open. The 'Source' folder in the Explorer view is selected. In the center, the code editor displays 'kernel.cpp' with the following content:

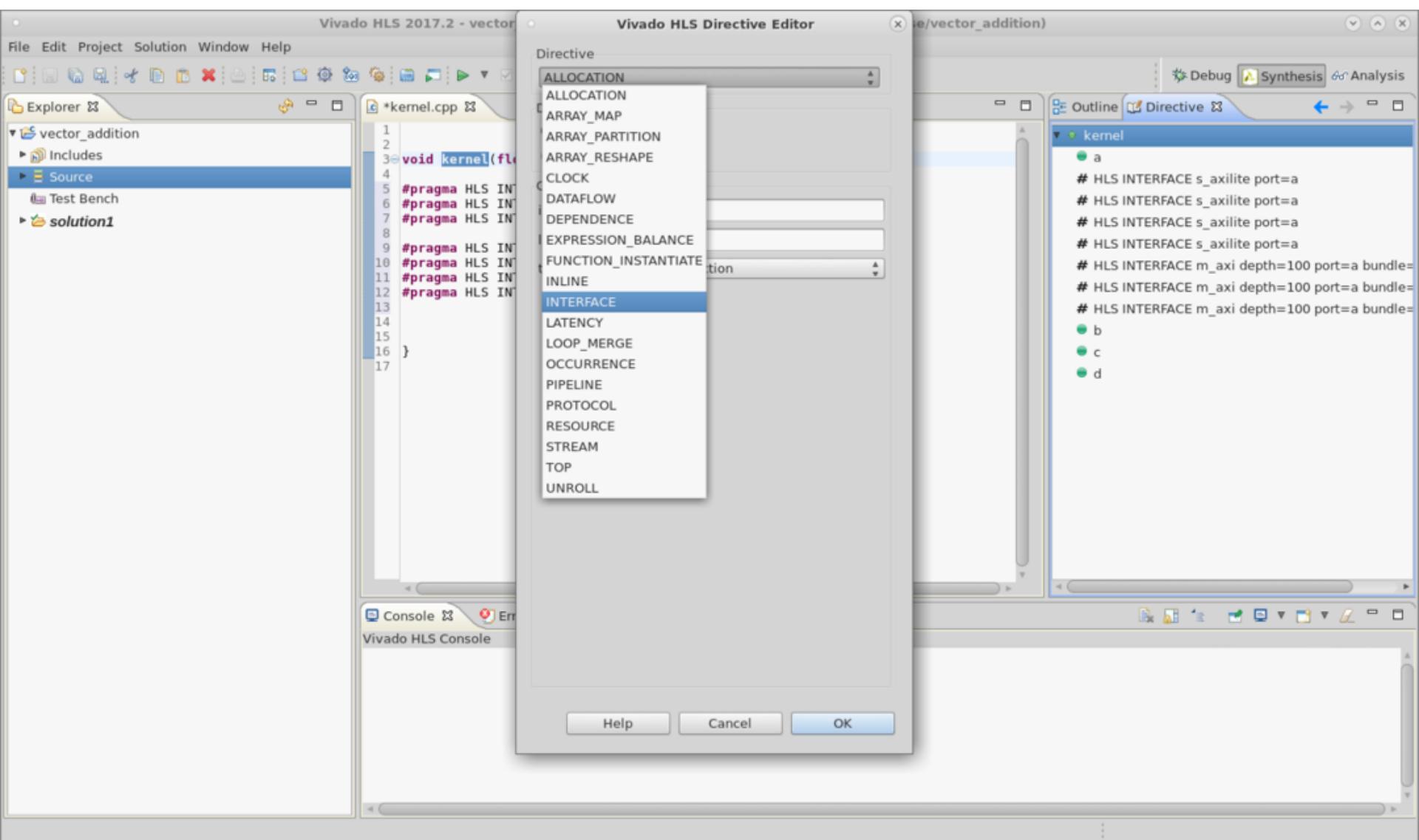
```
1
2
3 void kernel(float *a, float *b, float *c, float d){
4
5 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
6 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
7 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
8
9 #pragma HLS INTERFACE s_axilite port=a
10 #pragma HLS INTERFACE s_axilite port=b
11 #pragma HLS INTERFACE s_axilite port=c
12 #pragma HLS INTERFACE s_axilite port=d
13
14
15
16 }
```

A context menu is open over the line containing the first pragma directive (#pragma HLS INTERFACE m_axi depth=100 port=a). The menu is titled 'Insert Directive...' and contains the following options:

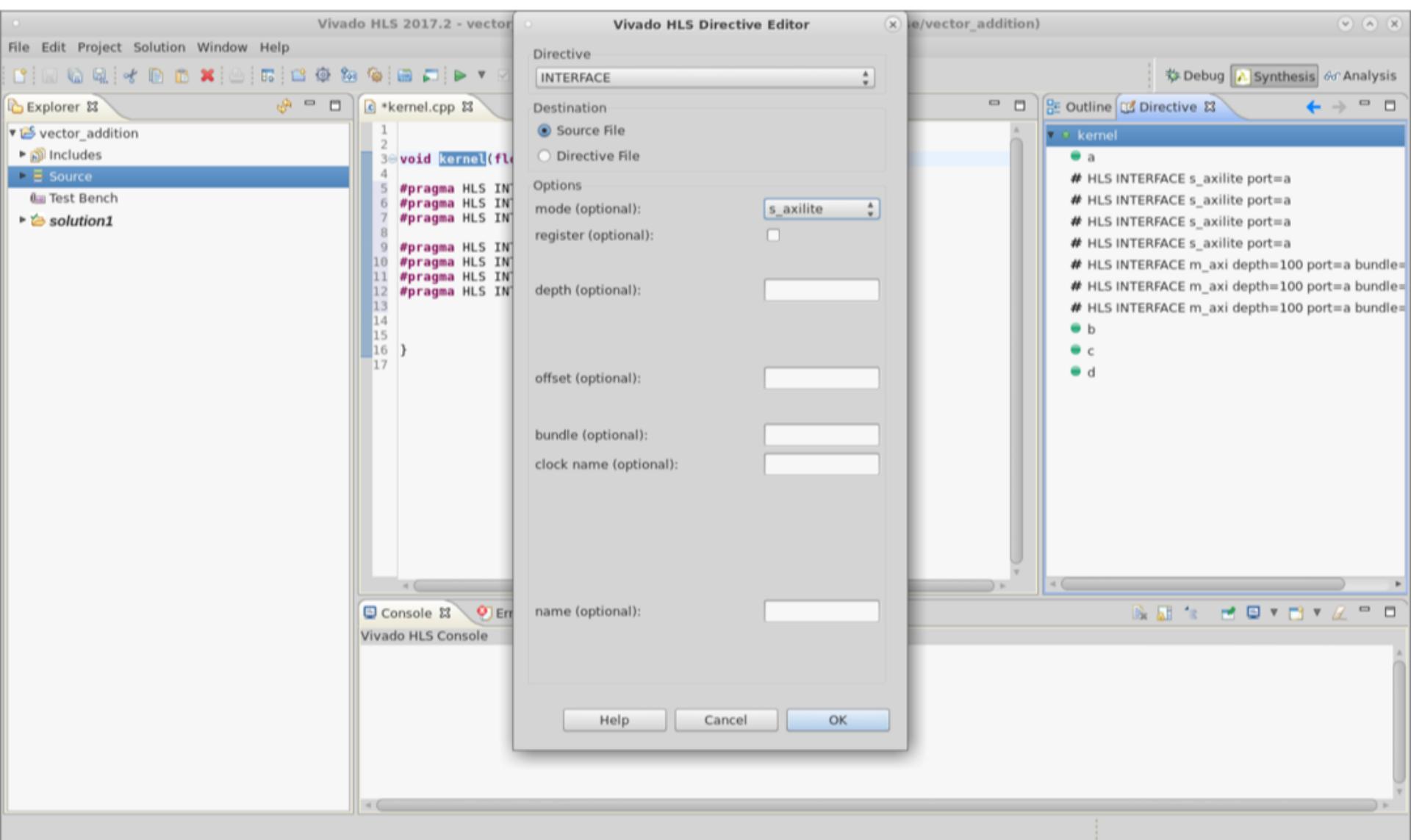
- # HLS INTERFACE s_axilite port=a
- # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
- # HLS INTERFACE m_axi depth=100 port=a bundle=gmem1
- # HLS INTERFACE m_axi depth=100 port=a bundle=gmem2

The 'Console' tab at the bottom shows the 'Vivado HLS Console'.

Interface Directive



Axilite Directive



Kernel Directive

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

The screenshot shows the Vivado HLS 2017.2 interface. The top menu bar includes File, Edit, Project, Solution, Window, Help, Debug, Synthesis, and Analysis. The left sidebar has sections for Explorer, vector_addition, Includes, Source (which is selected), Test Bench, and solution1. The main workspace contains a code editor for 'kernel.cpp' and a 'Directive' panel. The code editor shows the following C++ code:

```
1
2
3 void kernel(float *a, float *b, float *c, float d){
4
5 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
6 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
7 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
8
9 #pragma HLS INTERFACE s_axilite port=a
10 #pragma HLS INTERFACE s_axilite port=b
11 #pragma HLS INTERFACE s_axilite port=c
12 #pragma HLS INTERFACE s_axilite port=d
13 #pragma HLS INTERFACE s_axilite port=return
14
15
16
17
18 }
```

The 'Directive' panel on the right lists the generated directives:

- # HLS INTERFACE s_axilite port=return
- a
- # HLS INTERFACE s_axilite port=a
- # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
- b
- # HLS INTERFACE s_axilite port=b
- # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
- c
- # HLS INTERFACE s_axilite port=c
- # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
- d
- # HLS INTERFACE s_axilite port=d

The bottom of the interface shows tabs for Console, Errors, and Warnings, and the Vivado HLS Console area.

Naive Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer □

- vector_addition
- Includes
- Source
- Test Bench
- solution1

kernel.cpp □

```
1 #define N 100
2
3 void kernel(float *a, float *b, float *c, float d){
4
5     #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
6     #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
7     #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
8
9     #pragma HLS INTERFACE s_axilite port=a
10    #pragma HLS INTERFACE s_axilite port=b
11    #pragma HLS INTERFACE s_axilite port=c
12    #pragma HLS INTERFACE s_axilite port=d
13    #pragma HLS INTERFACE s_axilite port=return
14
15    loop:for(int i = 0; i < N; i++){
16        a[i] = b[i] + c[i]*d;
17    }
18
19 }
20
21
```

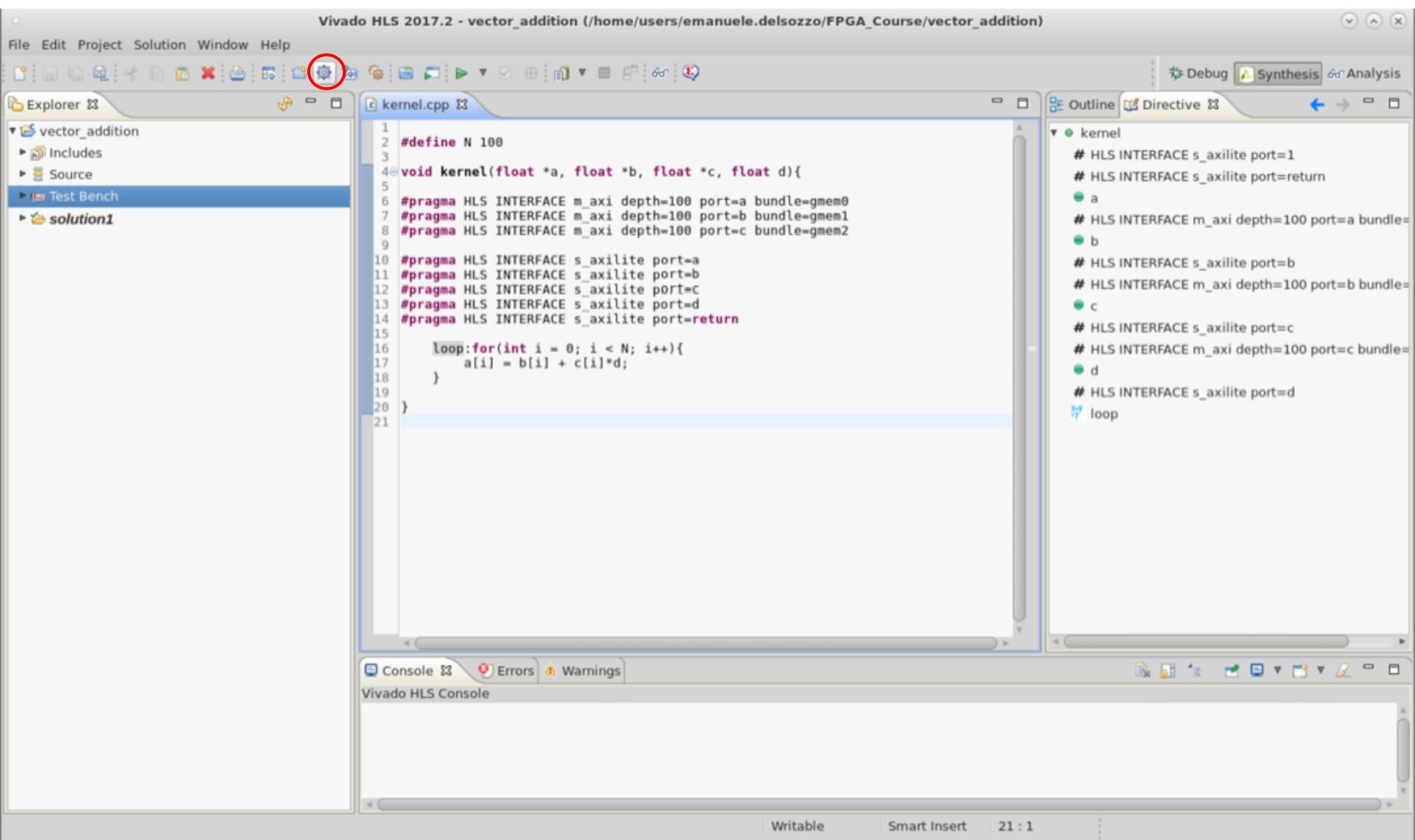
Outline □ Directive □

- kernel
 - # HLS INTERFACE s_axilite port=1
 - # HLS INTERFACE s_axilite port=return
 - a
 - b
 - c
 - d
 - loop

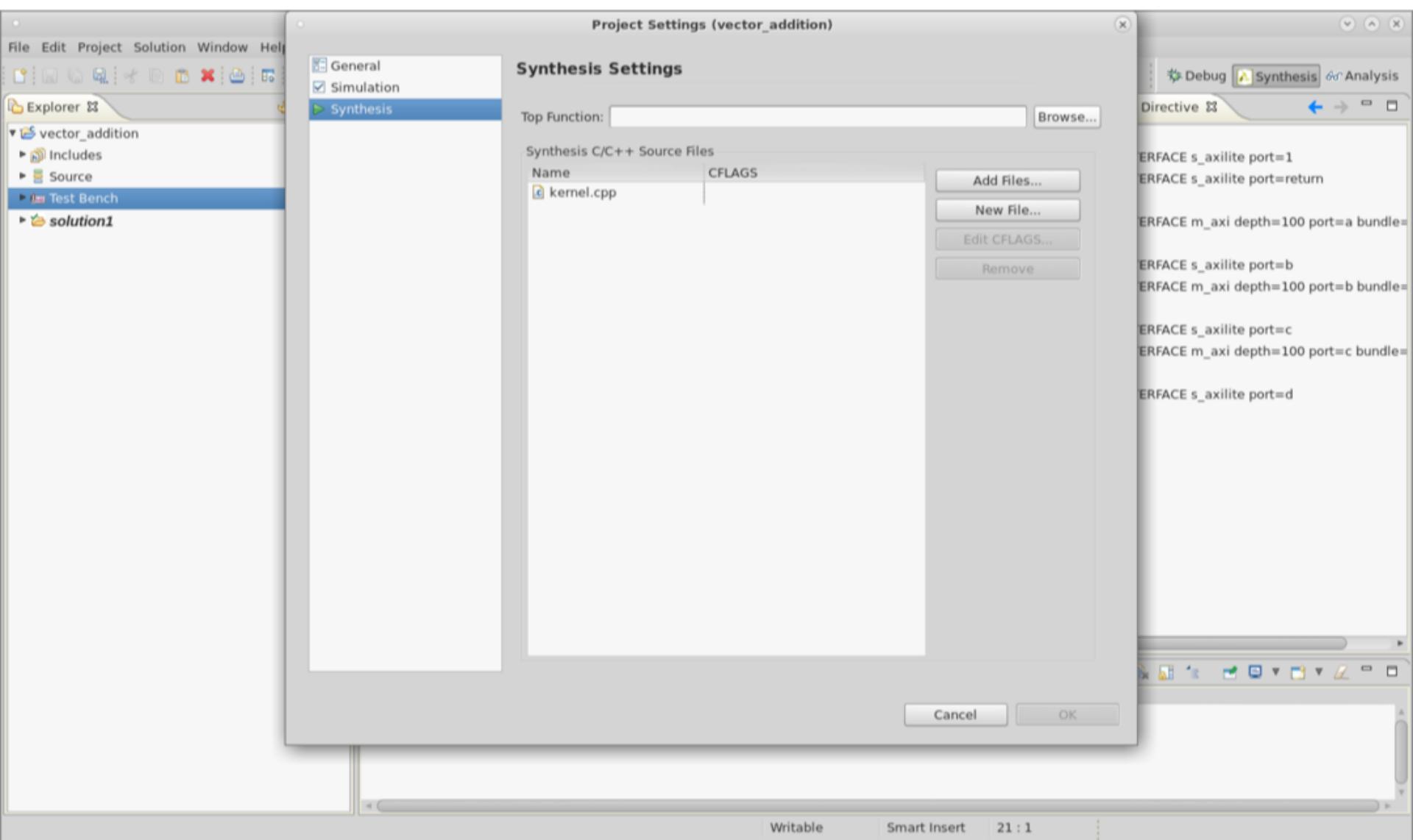
Console □ Errors □ Warnings □

Vivado HLS Console

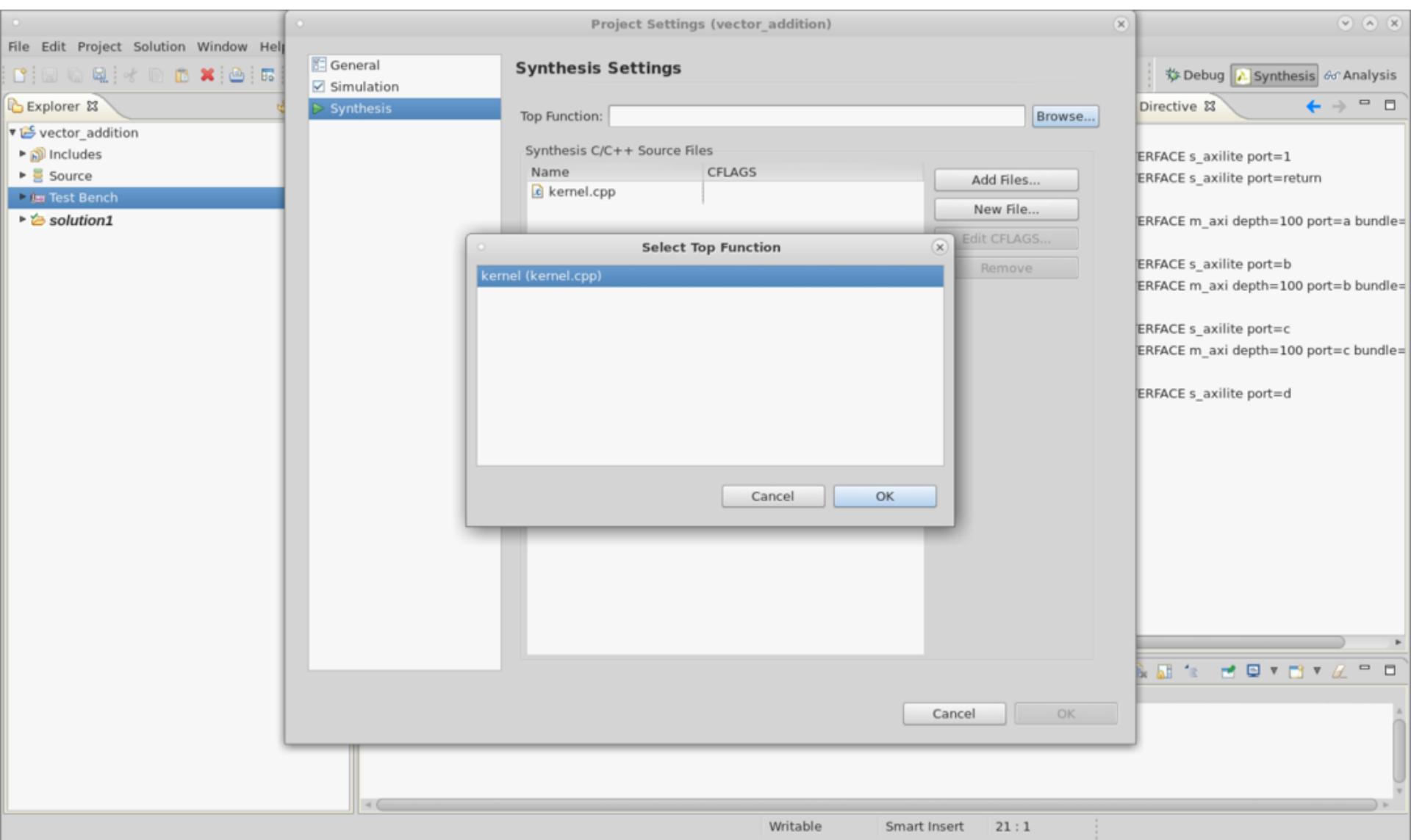
Project Settings



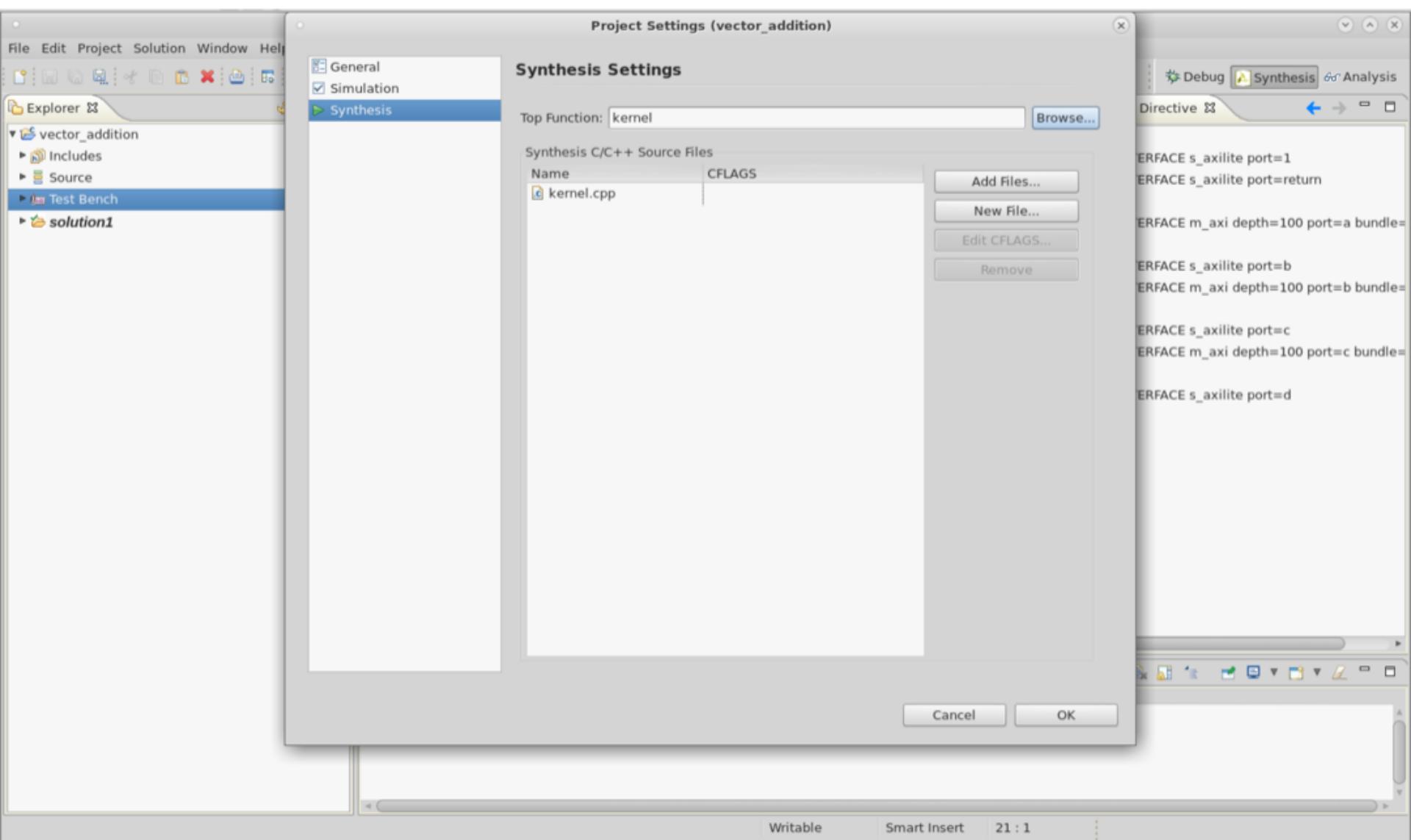
Top Function Selection



Top Function Selection



Top Function Selection



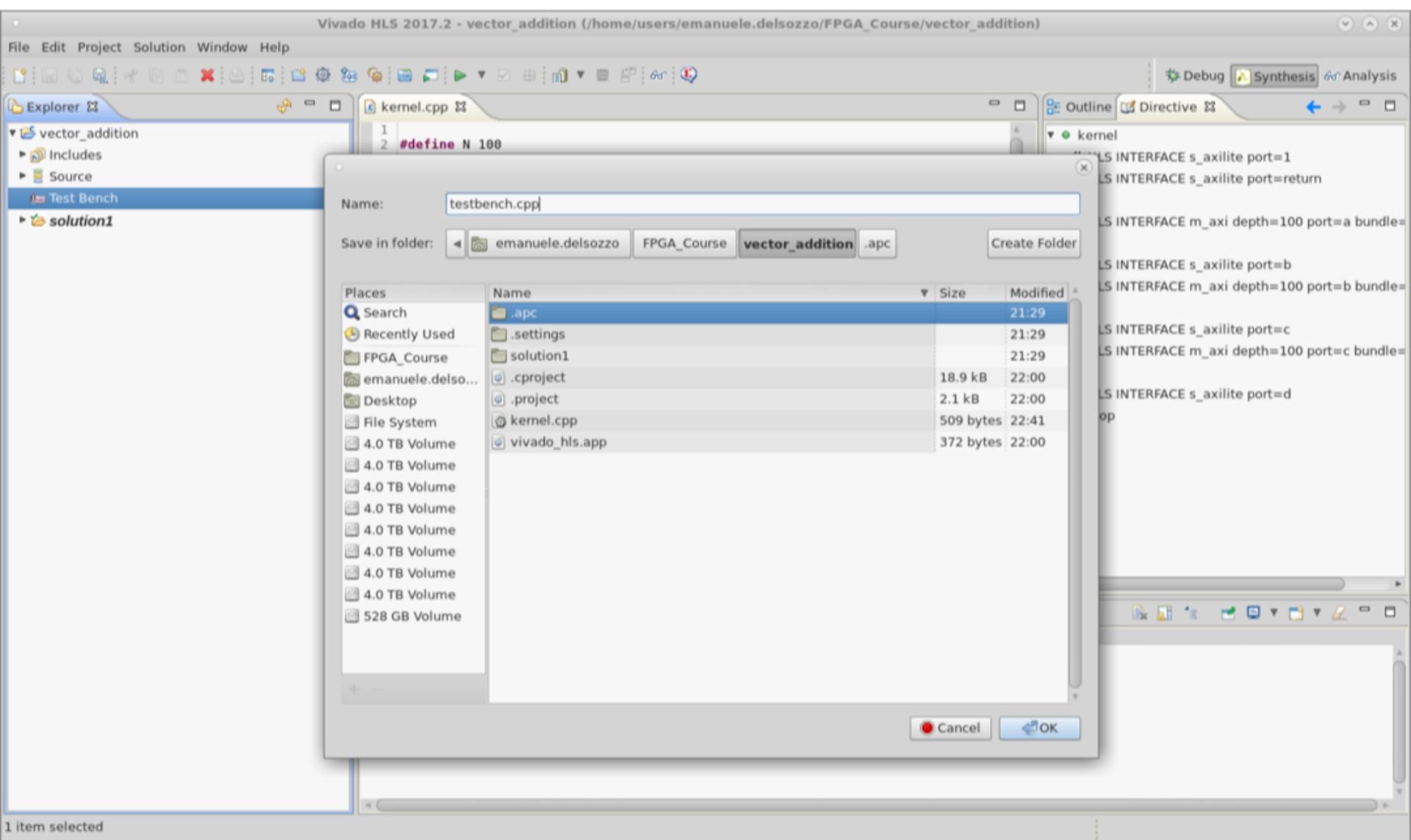
Testbench Creation

The screenshot shows the Vivado HLS 2017.2 interface with the following components:

- File Edit Project Solution Window Help**: Standard application menu.
- Explorer**: Shows the project structure under "vector_addition". A context menu is open over "Test Bench" with options: "Add File...", "New File...", and "Add Folder...".
- kernel.cpp**: The code editor window containing the following C++ code:

```
1 #define N 100
2
3 void kernel(float *a, float *b, float *c, float d){
4
5     #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
6     #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
7     #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
8
9     #pragma HLS INTERFACE s_axilite port=a
10    #pragma HLS INTERFACE s_axilite port=b
11    #pragma HLS INTERFACE s_axilite port=c
12    #pragma HLS INTERFACE s_axilite port=d
13    #pragma HLS INTERFACE s_axilite port=return
14
15    loop:for(int i = 0; i < N; i++){
16        a[i] = b[i] + c[i]*d;
17    }
18
19 }
20
21 }
```
- Outline**: Shows the hierarchical structure of the kernel code, including HLS INTERFACE declarations for ports a, b, c, d, and return, along with a loop block.
- Console**: The Vivado HLS Console window, which is currently empty.
- Errors** and **Warnings**: Status indicators for the build process.

Testbench Creation



Testbench

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer

- vector_addition
 - Includes
 - Source
 - kernel.cpp
 - Test Bench
 - testbench.cpp
- solution1

kernel.cpp testbench.cpp kernel_csim.log

```
1 #define N 100
2 #define X 20
3
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 void kernel(float *a, float *b, float *c, float d);
8
9 int main(){
10     float aCPU[N], aFPGA[N], b[N], c[N], d;
11
12     srand(0);
13     d = (float) rand() / ((float) (RAND_MAX/X));
14     for(int i = 0; i < N; i++){
15         b[i] = (float) rand() / ((float) (RAND_MAX/X));
16         c[i] = (float) rand() / ((float) (RAND_MAX/X));
17     }
18
19     for(int i = 0; i < N; i++){
20         aCPU[i] = b[i] + c[i]*d;
21     }
22
23     kernel(aFPGA, b, c, d);
24
25     for(int i = 0; i < N; i++){
26         if(aCPU[i] != aFPGA[i]){
27             printf("Error at index %d: %f != %f\n", i, aCPU[i], aFPGA[i]);
28             return 1;
29         }
30     }
31
32     return 0;
33 }
34 }
```

Outline Directive

- kernel
- main
 - aCPU
 - aFPGA
 - b
 - c
 - for Statement
 - for Statement
 - for Statement

Console Errors Warnings

Vivado HLS Console

The screenshot shows the Vivado HLS 2017.2 IDE interface. The main window displays the 'testbench.cpp' file with C++ code for a vector addition testbench. The code includes defines for N=100 and X=20, includes stdio.h and stdlib.h, and defines a kernel function. The main function generates random data for arrays a, b, and c, then calls the kernel with array d. It then compares the results of the CPU implementation (aCPU) with the FPGA implementation (aFPGA) and prints an error message if they differ. The 'Outline' tab on the right shows the hierarchical structure of the code, including declarations for aCPU, aFPGA, b, c, and three for loops. The 'Directive' tab shows the generated hardware description. At the bottom, there are tabs for 'Console', 'Errors', and 'Warnings', and a 'Vivado HLS Console' window which is currently empty.

Run C Simulation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

kernel.cpp

```

1 #define N 100
2 #define X 20
3
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 void kernel(float *a, float *b, float *c, float d);
8
9 int main(){
10
11     float aCPU[N], aFPGA[N], b[N], c[N], d;
12
13     srand(0);
14     d = (float) rand() / ((float) (RAND_MAX/X));
15     for(int i = 0; i < N; i++){
16         b[i] = (float) rand() / ((float) (RAND_MAX/X));
17         c[i] = (float) rand() / ((float) (RAND_MAX/X));
18     }
19
20     for(int i = 0; i < N; i++){
21         aCPU[i] = b[i] + c[i]*d;
22     }
23
24     kernel(aFPGA, b, c, d);
25
26     for(int i = 0; i < N; i++){
27         if(aCPU[i] != aFPGA[i]){
28             printf("Error at index %d: %f != %f\n", i, aCPU[i], aFPGA[i]);
29             return 1;
30         }
31     }
32
33     return 0;
34 }
35

```

testbench.cpp

Outline

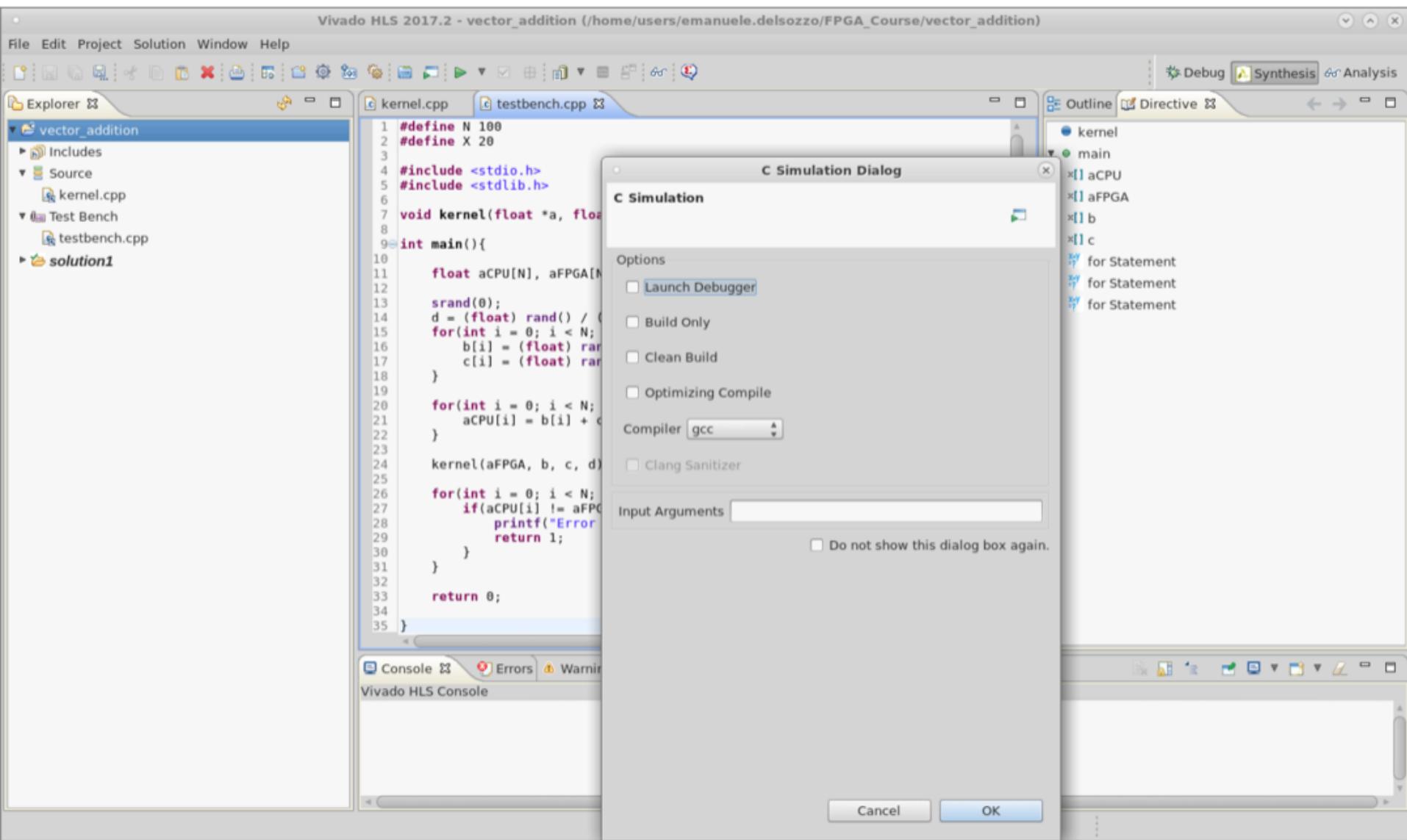
- kernel
- main
 - aCPU
 - aFPGA
 - b
 - c
 - for Statement
 - for Statement
 - for Statement

Console

Vivado HLS Console

Writable Smart Insert 35 : 2

C Simulation Settings



C Simulation Result

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer X kernel.cpp testbench.cpp kernel_csim.log X

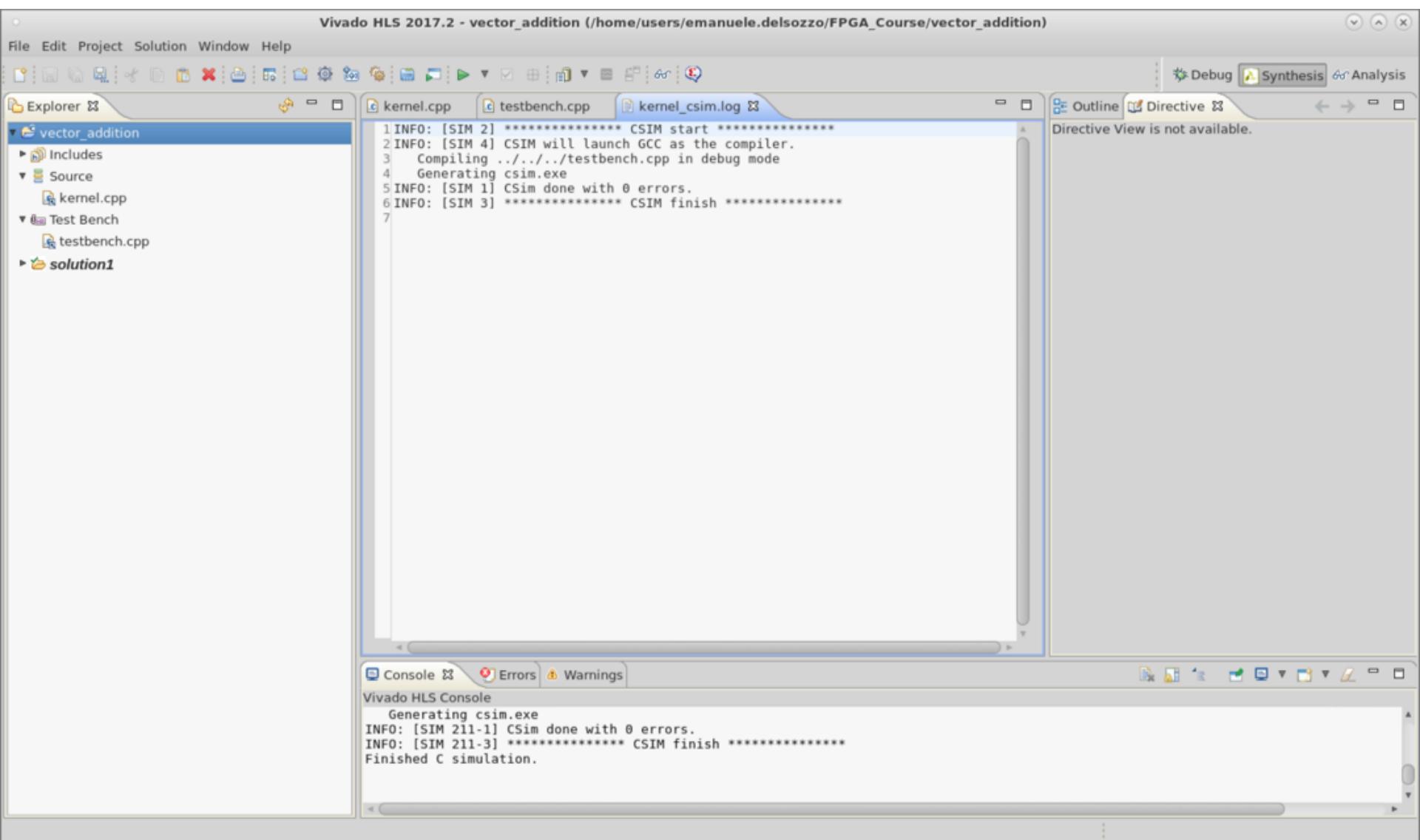
vector_addition Includes Source kernel.cpp Test Bench testbench.cpp solution1

1 INFO: [SIM 2] ***** CSIM start *****
2 INFO: [SIM 4] CSIM will launch GCC as the compiler.
3 Compiling ../../testbench.cpp in debug mode
4 Generating csim.exe
5 INFO: [SIM 1] CSim done with 0 errors.
6 INFO: [SIM 3] ***** CSIM finish *****
7

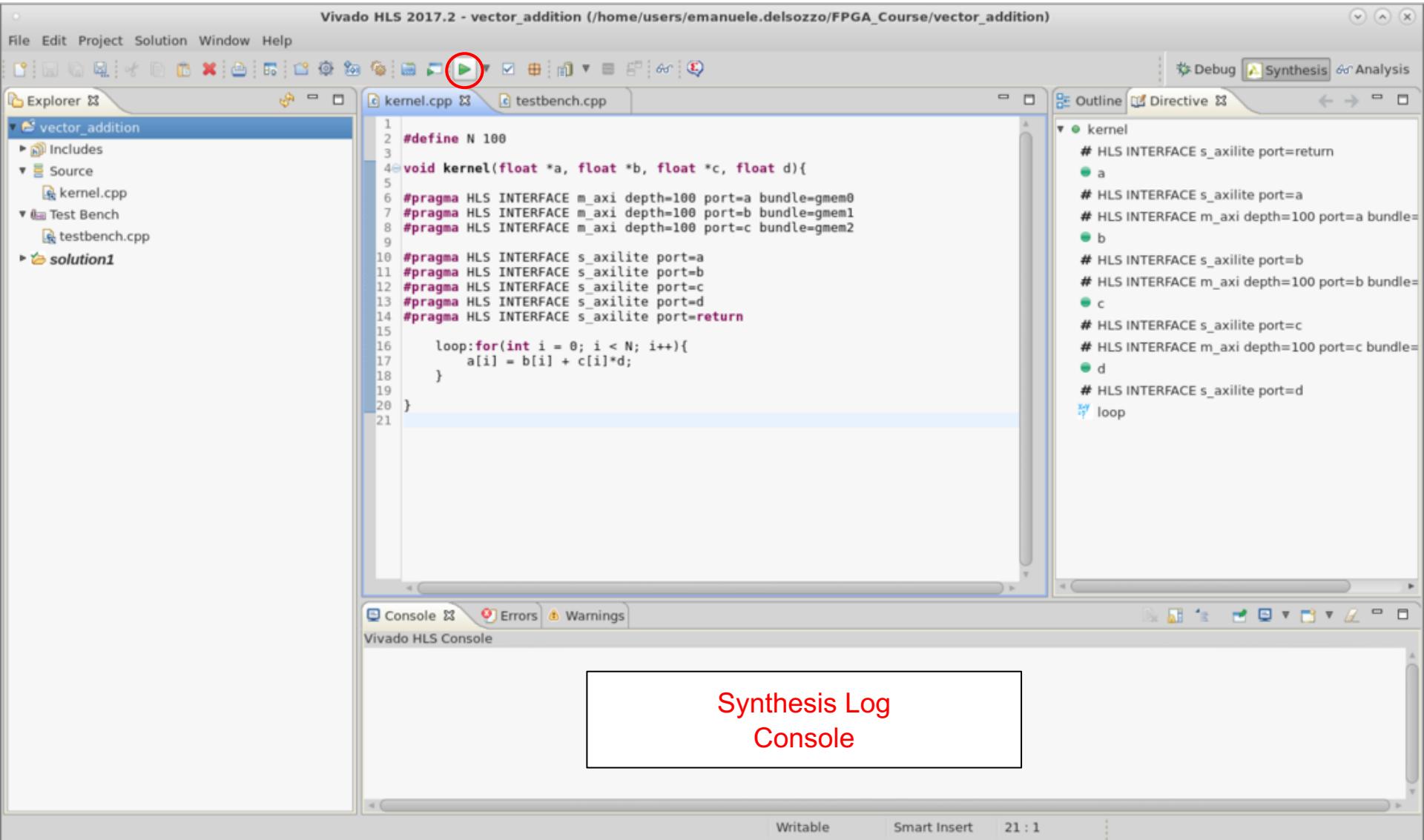
Outline Directive X Directive View is not available.

Console X Errors Warnings

Vivado HLS Console
Generating csim.exe
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
Finished C simulation.

A screenshot of the Vivado HLS 2017.2 software interface. The window title is "Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)". The menu bar includes File, Edit, Project, Solution, Window, and Help. The toolbar has various icons for file operations. The left sidebar shows a project tree with "vector_addition" expanded, containing "Includes", "Source" (with "kernel.cpp" selected), "Test Bench" (with "testbench.cpp" selected), and "solution1". The main workspace has three tabs: "kernel.cpp", "testbench.cpp", and "kernel_csim.log". The "kernel_csim.log" tab displays simulation logs with entries like "INFO: [SIM 2] ***** CSIM start *****", "Compiling ../../testbench.cpp in debug mode", "Generating csim.exe", and "INFO: [SIM 3] ***** CSIM finish *****". Below the workspace is a "Console" tab showing the command "Generating csim.exe" and simulation completion messages. The bottom right corner of the interface has the NECST laboratory logo.

Run Synthesis



Synthesis Report

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp Synthesis(solution1)

Debug Synthesis Analysis

Synthesis Report for 'kernel'

General Information

Date: Thu Jan 18 09:37:57 2018
 Version: 2017.2 (Build 1909853 on Thu Jun 15 18:55:24 MDT 2017)
 Project: vector_addition
 Solution: solution1
 Product family: virtex7
 Target device: xc7vx485tffg1761-2

Performance Estimates

- Timing (ns)**
 - Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25
- Latency (clock cycles)**
 - Summary**

Latency	Interval			
min	max	min	max	Type
812	812	813	813	none
 - Detail**
 - Instance**

Console Errors Warnings

Vivado HLS Console

```

INFO: [RTGEN 206-100] Finished creating RTL model for 'kernel'.
INFO: [HLS 200-111] Elapsed time: 0.05 seconds; current allocated memory: 60.821 MB.
INFO: [HLS 200-111] Finished generating all RTL models Time (s): cpu = 00:00:03 ; elapsed = 00:00:03 . Memory (MB): peak = 564.555 ;
INFO: [SYSC 207-301] Generating SystemC RTL for kernel.
INFO: [VHDL 208-304] Generating VHDL RTL for kernel.
INFO: [VLOG 209-307] Generating Verilog RTL for kernel.
INFO: [HLS 200-112] Total elapsed time: 2.52 seconds; peak allocated memory: 60.821 MB.
Finished C synthesis.
  
```

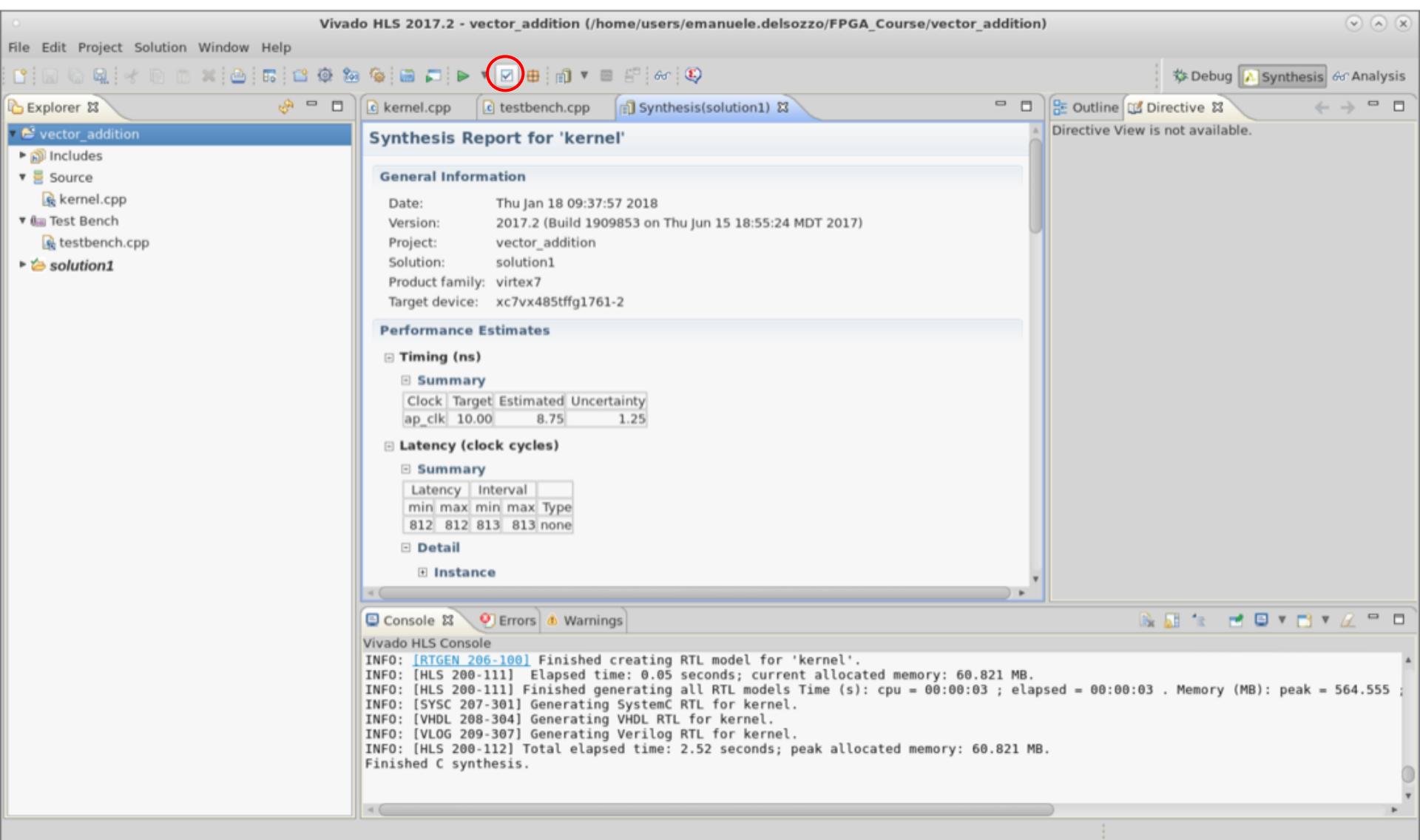
Naive Implementation Report

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

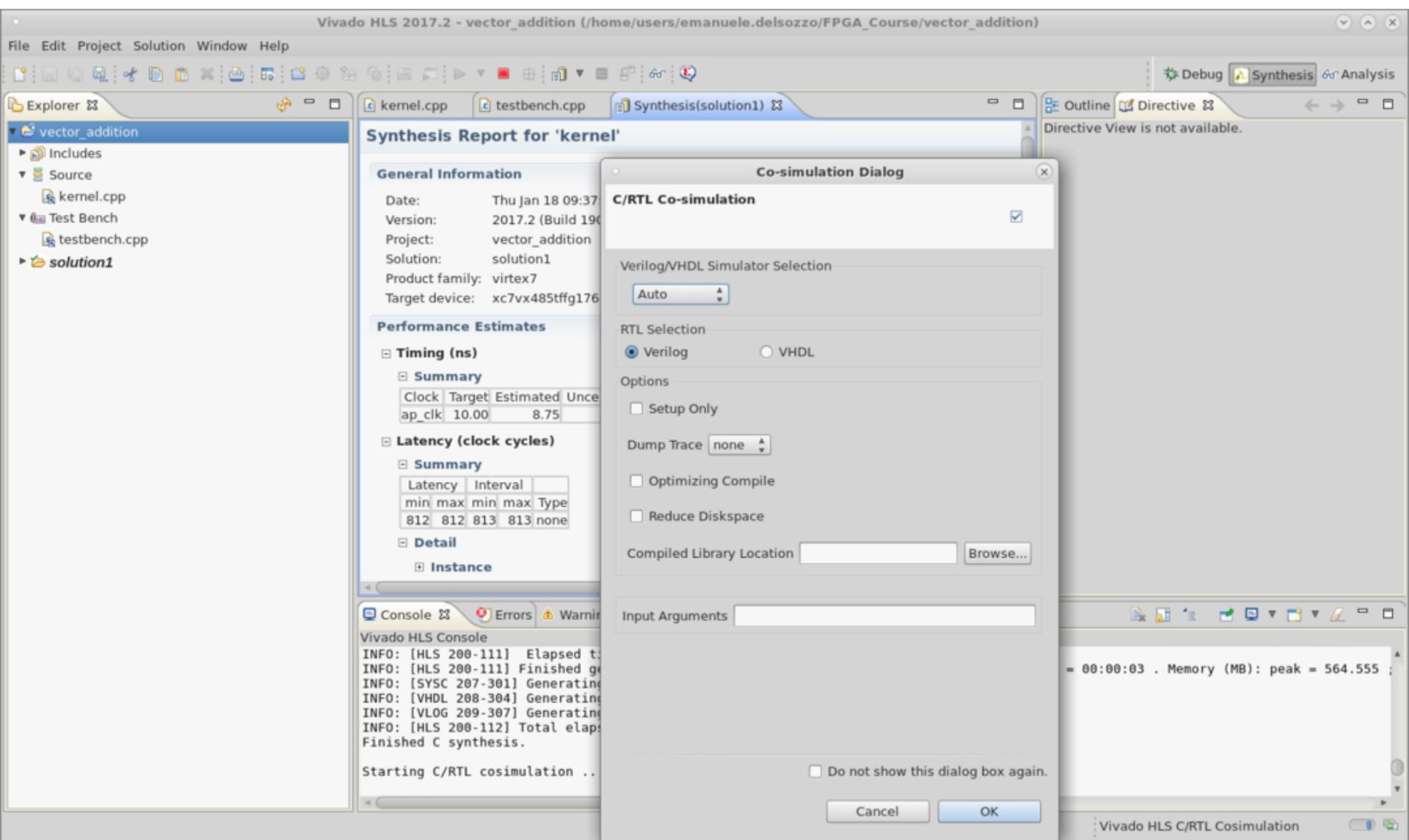
Latency		Interval		
min	max	min	max	Type
812	812	813	813	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	~0%	~0%	~0%	~0%

Run Co-Simulation



Co-Simulation Settings



Co-Simulation Results

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer Debug Synthesis Analysis

vector_addition

- Includes
- Source
 - kernel.cpp
- Test Bench
 - testbench.cpp
- solution1

kernel.cpp testbench.cpp Synthesis(solution) Simulation(solution)

Cosimulation Report for 'kernel'

Result

		Latency	Interval
RTL	Status	min avg max	min avg max
VHDL	NA NA NA NA NA NA NA NA		
Verilog	Pass 870 870 870 NA NA NA		

Export the report(.html) using the [Export Wizard](#)

Console Errors Warnings

Vivado HLS Console

```
////////////////////////////////////////////////////////////////
$finish called at time : 8885 ns : File "/home/users/emanuele.delozzo/FPGA_Course/vector_addition/solution1/sim/verilog/kernel.aut
## quit
INFO: [Common 17-206] Exiting xsim at Thu Jan 18 09:40:08 2018...
INFO: [COSIM_212-316] Starting C post checking ...
INFO: [COSIM_212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [COSIM 212-211] II is measurable only when transaction number is greater than 1 in RTL simulation. Otherwise, they will be ma
Finished C/RTL cosimulation.
```

V1 Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp Synthesis(solution1)

```

1 #include "string.h"
2
3 #define N 100
4
5 void kernel(float *a, float *b, float *c, float d){
6
7 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
8 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
9 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
10
11 #pragma HLS INTERFACE s_axilite port=a
12 #pragma HLS INTERFACE s_axilite port=b
13 #pragma HLS INTERFACE s_axilite port=c
14 #pragma HLS INTERFACE s_axilite port=d
15 #pragma HLS INTERFACE s_axilite port=return
16
17     float a_local[N], b_local[N], c_local[N], d_local;
18
19     memcpy(b_local, b, sizeof(float)*N);
20     memcpy(c_local, c, sizeof(float)*N);
21     d_local = d;
22
23     loop:for(int i = 0; i < N; i++){
24         a_local[i] = b_local[i] + c_local[i]*d_local;
25     }
26
27     memcpy(a, a_local, sizeof(float)*N);
28
29 }
30

```

Outline Directive

- kernel
 - # HLS INTERFACE s_axilite port=return
 - a
 - # HLS INTERFACE s_axilite port=a
 - # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
 - b
 - # HLS INTERFACE s_axilite port=b
 - # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
 - c
 - # HLS INTERFACE s_axilite port=c
 - # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
 - d
 - # HLS INTERFACE s_axilite port=d
 - a_local
 - b_local
 - c_local
 - loop

Console Errors Warnings

Vivado HLS Console

V1 Implementation Log

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delsozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Console Errors Warnings Debug Synthesis Analysis

Vivado HLS Console

```

INFO: [HLS 200-10] On os "CentOS Linux release 7.3.1611 (Core) "
INFO: [HLS 200-10] In directory '/home/users/emanuele.delsozzo/FPGA_Course'
INFO: [HLS 200-10] Opening project '/home/users/emanuele.delsozzo/FPGA_Course/vector_addition'.
INFO: [HLS 200-10] Adding design file 'vector_addition/kernel.cpp' to the project
INFO: [HLS 200-10] Adding test bench file 'vector_addition/testbench.cpp' to the project
INFO: [HLS 200-10] Opening solution '/home/users/emanuele.delsozzo/FPGA_Course/vector_addition/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-10] Setting target device to 'xc7vx48tffg1761-2'
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-10] Analyzing design file 'vector_addition/kernel.cpp' ...
INFO: [HLS 200-10] Validating synthesis directives ...
INFO: [HLS 200-111] Finished Checking Pragmas Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 436.555 ; gain = 12.797 ; free physical = 195981 ; free virtual = 195977
INFO: [HLS 200-111] Finished Linking Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 436.555 ; gain = 12.797 ; free physical = 195977 ; free virtual = 195975
INFO: [HLS 200-10] Starting code transformations ...
INFO: [HLS 200-111] Finished Standard Transforms Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 436.555 ; gain = 12.797 ; free physical = 195975 ; free virtual = 195973
INFO: [HLS 200-10] Checking synthesizability ...
INFO: [HLS 200-111] Finished Checking Synthesizability Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 436.555 ; gain = 12.797 ; free physical = 195973 ; free virtual = 195971
INFO: [HLS 200-111] Finished Pre-synthesis Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 564.551 ; gain = 140.793 ; free physical = 195909 ; free virtual = 195907
INFO: [XFORM 203-811] Inferring bus burst read of length 100 on port 'gmem1' (vector_addition/kernel.cpp:19:2).
INFO: [XFORM 203-811] Inferring bus burst read of length 100 on port 'gmem2' (vector_addition/kernel.cpp:20:2).
INFO: [XFORM 203-811] Inferring bus burst write of length 100 on port 'gmem0' (vector_addition/kernel.cpp:27:2).
INFO: [HLS 200-111] Finished Architecture Synthesis Time (s). cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 564.551 ; gain = 140.793 ; free physical = 195906 ; free virtual = 195904
INFO: [HLS 200-10] Starting hardware synthesis ...
INFO: [HLS 200-10] Synthesizing 'kernel' ...
INFO: [HLS 200-10] -----
INFO: [HLS 200-10] -- Implementing module 'kernel'
INFO: [HLS 200-10] -----
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-611] Pipelining loop 'memcpy.b_local.b'.
INFO: [SCHED 204-611] Pipelining result: Target II: 1, Final II: 1, Depth: 3.
INFO: [SCHED 204-611] Pipelining loop 'memcpy.c_local.c'.
INFO: [SCHED 204-611] Pipelining result: Target II: 1, Final II: 1, Depth: 3.
INFO: [SCHED 204-611] Pipelining loop 'memcpy.a.a_local.gep'.
INFO: [SCHED 204-611] Pipelining result: Target II: 1, Final II: 1, Depth: 3.
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111] Elapsed time: 2.24 seconds; current allocated memory: 60.052 MB.
INFO: [BIND 205-100] Starting micro-architecture generation ...
INFO: [BIND 205-101] Performing variable lifetime analysis.
INFO: [BIND 205-101] Exploring resource sharing.
INFO: [BIND 205-101] Binding ...
INFO: [BIND 205-100] Finished micro-architecture generation.
INFO: [HLS 200-111] Elapsed time: 0.12 seconds; current allocated memory: 60.553 MB.
INFO: [HLS 200-10] -----
INFO: [HLS 200-10] -- Generating RTL for module 'kernel'
INFO: [HLS 200-10] -----

```

V1 Implementation Report

68

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency		Interval		
min	max	min	max	Type
1227	1227	1228	1228	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	~0%	~0%	~0%	~0%

V1 Implementation Loops

69

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
memcpy b	101	101	3	1	1	100	yes
memcpy c	101	101	3	1	1	100	yes
loop	900	900	9	-	-	100	no
memcpy a	101	101	3	1	1	100	yes

V1 Analysis

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer **vector_addition**
Includes
Source kernel.cpp
Test Bench testbench.cpp
solution1

Synthesis Report for 'kernel'

General Information

Date: Thu Jan 18 10:25:30 2018
Version: 2017.2 (Build 1909853 on Thu Jun 15 18:55:24 MDT 2017)
Project: vector_addition
Solution: solution1
Product family: virtex7
Target device: xc7vx485tffg1761-2

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency (clock cycles)

Summary

Latency	Interval
min	max
1227	1227
1228	1228
Type	
none	

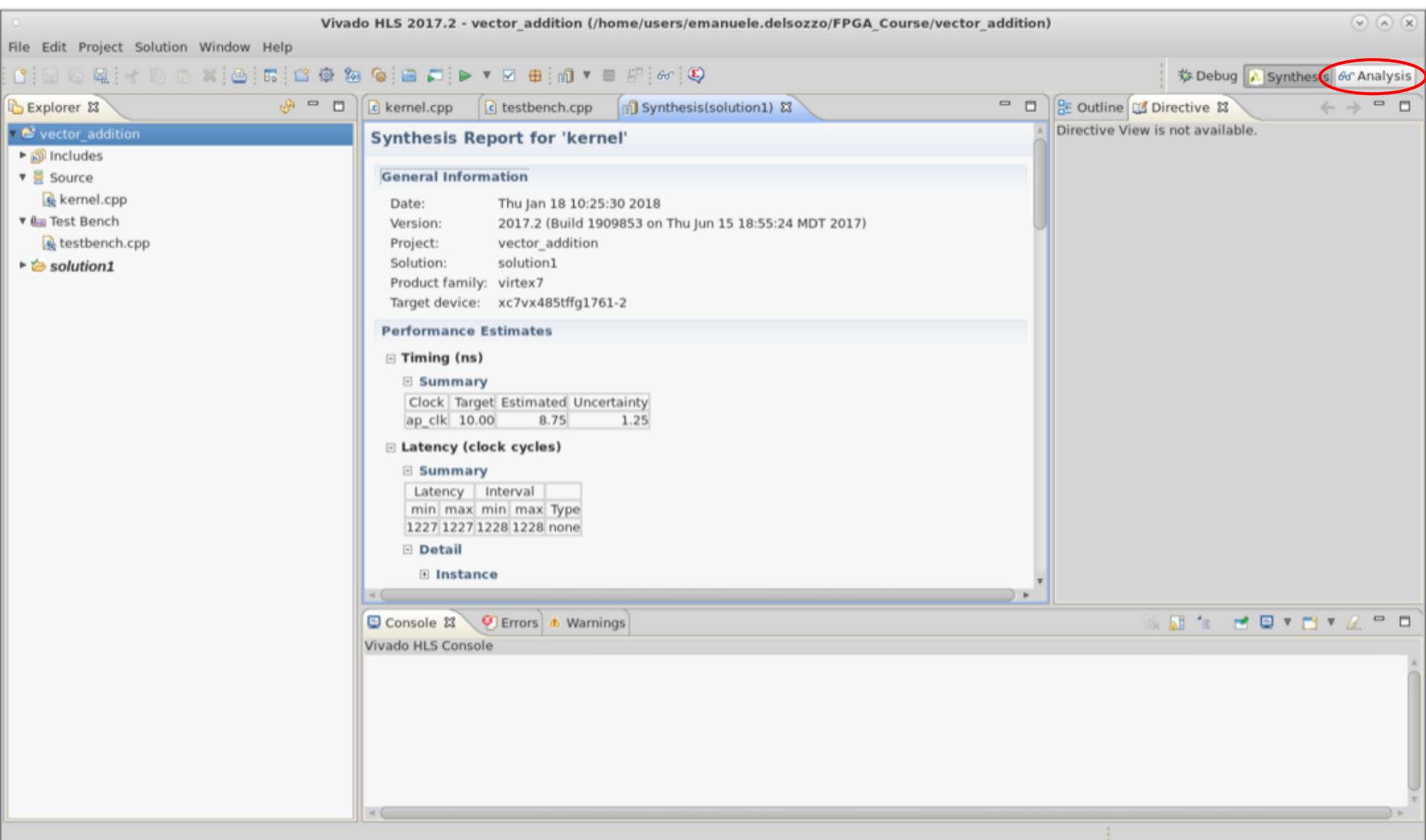
Detail

Instance

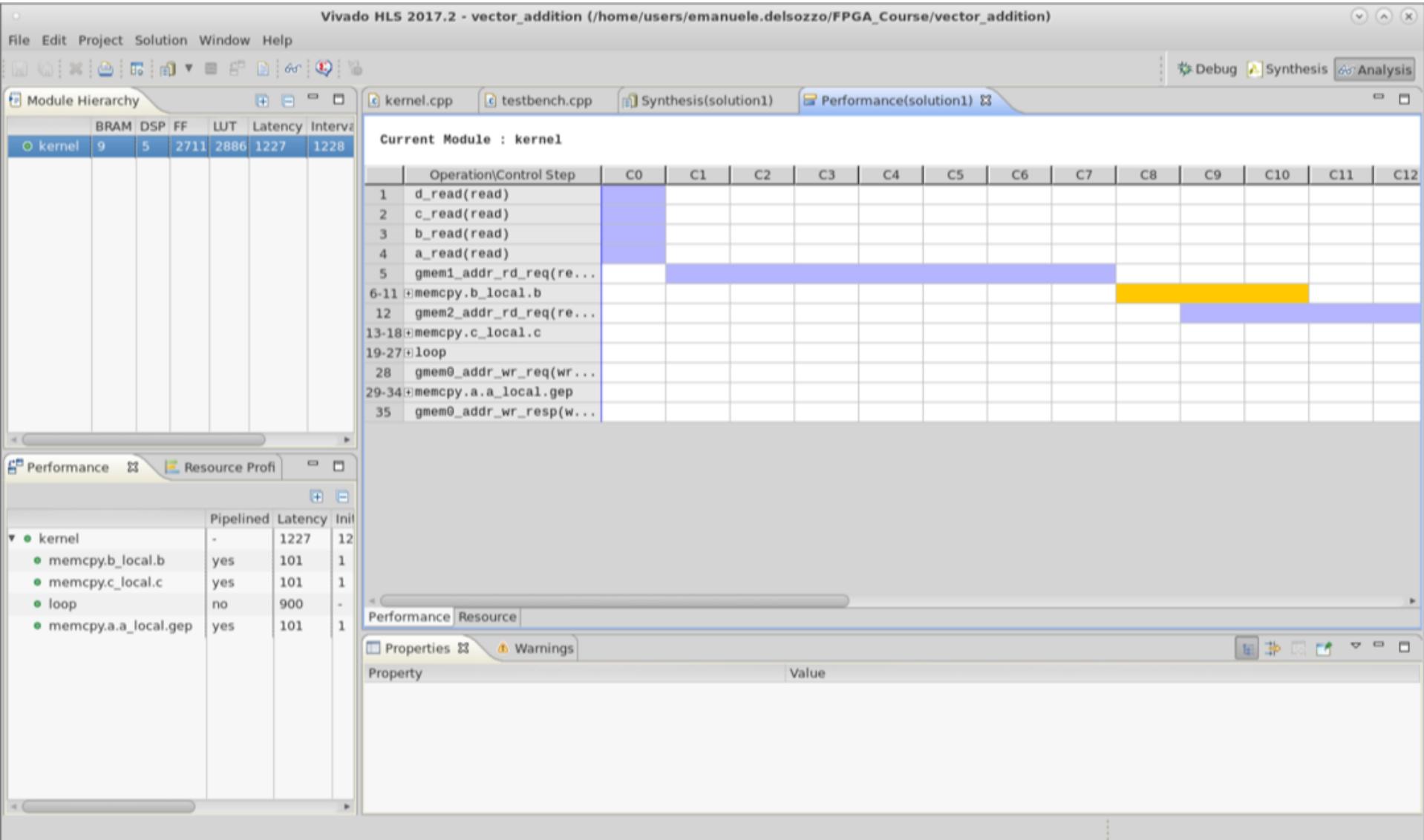
Console Errors Warnings
Vivado HLS Console

Debug Synthesis **Analysis**

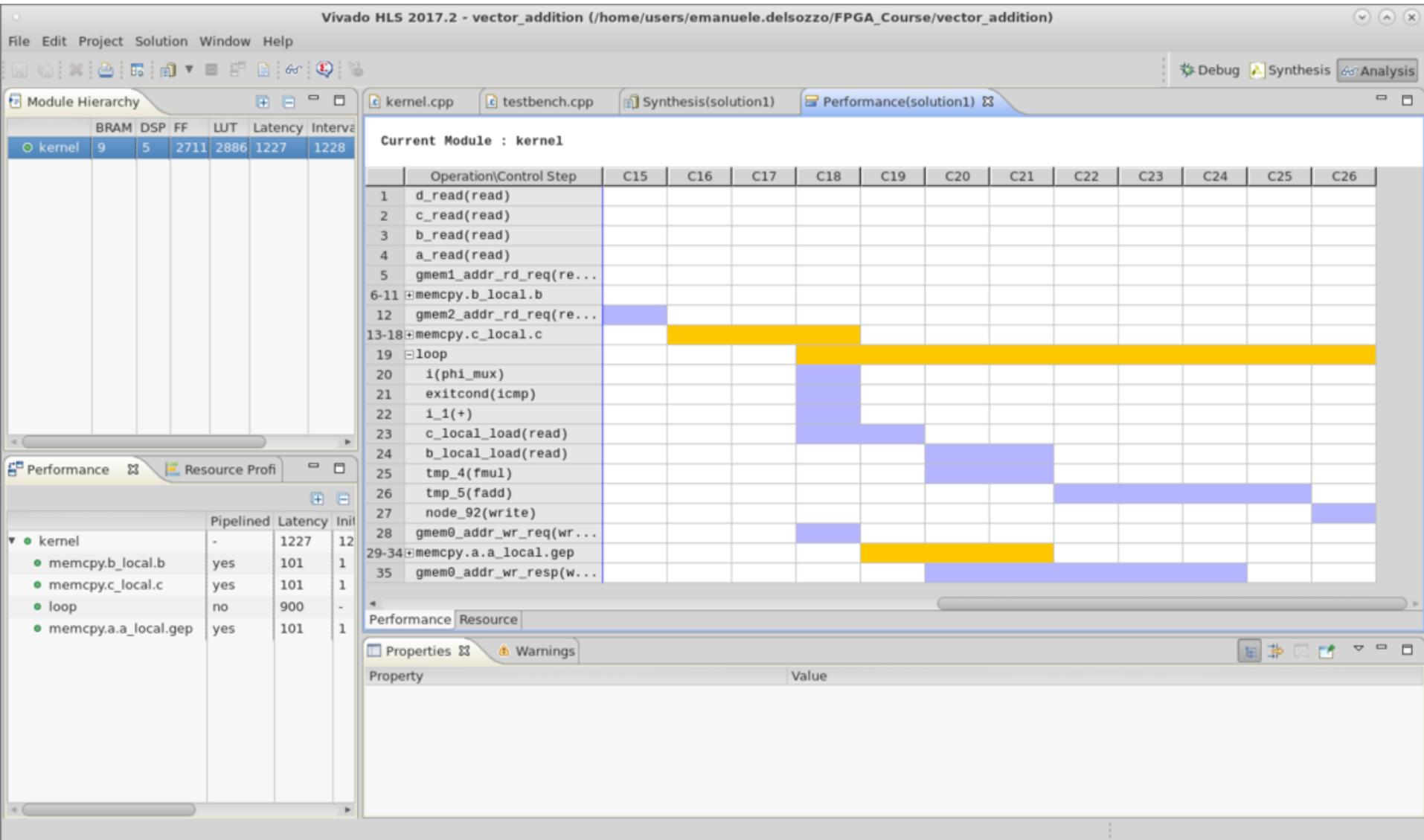
Outline Directive
Directive View is not available.



V1 Analysis



V1 Analysis



V2 Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp Synthesis(solutio... Performance(soluti...

```

vector_addition
  Includes
  Source
    kernel.cpp
  Test Bench
    testbench.cpp
solution1

1 #include "string.h"
2
3 #define N 100
4
5 void kernel(float *a, float *b, float *c, float d){
6
7 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
8 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
9 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
10
11 #pragma HLS INTERFACE s_axilite port=a
12 #pragma HLS INTERFACE s_axilite port=b
13 #pragma HLS INTERFACE s_axilite port=c
14 #pragma HLS INTERFACE s_axilite port=d
15 #pragma HLS INTERFACE s_axilite port=return
16
17     float a_local[N], b_local[N], c_local[N], d_local;
18
19     memcpy(b_local, b, sizeof(float)*N);
20     memcpy(c_local, c, sizeof(float)*N);
21     d_local = d;
22
23     loop:for(int i = 0; i < N; i++){
24 #pragma HLS PIPELINE
25         a_local[i] = b_local[i] + c_local[i]*d_local;
26     }
27
28     memcpy(a, a_local, sizeof(float)*N);
29
30 }
31

```

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 24 : 21

Outline Directive

kernel
 # HLS INTERFACE s_axilite port=return
 a
 # HLS INTERFACE s_axilite port=a
 # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
 b
 # HLS INTERFACE s_axilite port=b
 # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
 c
 # HLS INTERFACE s_axilite port=c
 # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
 d
 # HLS INTERFACE s_axilite port=d
 # HLS PIPELINE
 a_local
 b_local
 c_local
 loop
 # HLS PIPELINE

V2 Implementation Log

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delsozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Console Errors Warnings Debug Synthesis Analysis

Vivado HLS Console

Starting C synthesis ...

```
/xilinx/software/Vivado_HLS/2017.2/bin/vivado_hls /home/users/emanuele.delsozzo/FPGA_Course/vector_addition/solution1/csynth.tcl
INFO: [HLS 200-10] Running '/xilinx/software/Vivado HLS/2017.2/bin/unwrapped/lnx64.o/vivado_hls'
INFO: [HLS 200-10] For user 'emanuele.delsozzo' on host 'NAGS30' (Linux_x86_64 version 3.10.0-514.16.1.el7.x86_64) on Thu Jan 18 11:02:50 CET 2018
INFO: [HLS 200-10] On os "CentOS Linux release 7.3.1611 (Core) "
INFO: [HLS 200-10] In directory '/home/users/emanuele.delsozzo/FPGA_Course'
INFO: [HLS 200-10] Opening project '/home/users/emanuele.delsozzo/FPGA_Course/vector_addition'.
INFO: [HLS 200-10] Adding design file 'vector_addition/kernel.cpp' to the project
INFO: [HLS 200-10] Adding test bench file 'vector_addition/testbench.cpp' to the project
INFO: [HLS 200-10] Opening solution '/home/users/emanuele.delsozzo/FPGA_Course/vector_addition/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-10] Setting target device to 'xc7vx485tffg1761-2'
INFO: [HLS 200-10] Analyzing design file 'vector_addition/kernel.cpp' ...
INFO: [HLS 200-10] Validating synthesis directives ...
INFO: [HLS 200-111] Finished Checking Pragmas Time (s): cpu = 00:00:02 ; elapsed = 00:00:02 . Memory (MB): peak = 436.559 ; gain = 12.797 ; free physical = 194630 ; free virtual = 194626
INFO: [HLS 200-111] Finished Linking Time (s): cpu = 00:00:02 ; elapsed = 00:00:02 . Memory (MB): peak = 436.559 ; gain = 12.797 ; free physical = 194626 ; free virtual = 194626
INFO: [HLS 200-10] Starting code transformations ...
INFO: [HLS 200-111] Finished Standard Transforms Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 436.559 ; gain = 12.797 ; free physical = 194619 ; free virtual = 194619
INFO: [HLS 200-10] Checking synthesizability ...
INFO: [HLS 200-111] Finished Checking Synthesizability Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 436.559 ; gain = 12.797 ; free physical = 194619 ; free virtual = 194619
INFO: [HLS 200-111] Finished Pre-synthesis Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 564.555 ; gain = 140.793 ; free physical = 194599 ; free virtual = 194599
INFO: [XFORM 203-811] Inferring bus burst read of length 100 on port 'gmem1' (vector_addition/kernel.cpp:26:2).
INFO: [XFORM 203-811] Inferring bus burst read of length 100 on port 'gmem2' (vector_addition/kernel.cpp:27:2).
INFO: [XFORM 203-811] Inferring bus burst write of length 100 on port 'gmem0' (vector_addition/kernel.cpp:37:2).
INFO: [HLS 200-111] Finished Architecture Synthesis Time (s): cpu = 00:00:03 ; elapsed = 00:00:02 . Memory (MB): peak = 564.555 ; gain = 140.793 ; free physical = 194666 ; free virtual = 194666
INFO: [HLS 200-10] Starting hardware synthesis ...
INFO: [HLS 200-10] Synthesizing 'kernel' ...
INFO: [HLS 200-10] -----
INFO: [HLS 200-10] -- Implementing module 'kernel'
INFO: [HLS 200-10] -----
INFO: [SCHED 204-111] Starting scheduling ...
INFO: [SCHED 204-611] Pipelining loop 'memcpy.b_local.b'.
INFO: [SCHED 204-611] Pipelining result: Target II: 1, Final II: 1, Depth: 3.
INFO: [SCHED 204-611] Pipelining loop 'memcpy.c_local.c'.
INFO: [SCHED 204-611] Pipelining result: Target II: 1, Final II: 1, Depth: 3.
INFO: [SCHED 204-611] Pipelining loop 'loop'.
INFO: [SCHED 204-611] Pipelining result: Target II: 1, Final II: 1, Depth: 9.
INFO: [SCHED 204-611] Pipelining loop 'memcpy.a_a_local.gep'.
INFO: [SCHED 204-611] Pipelining result: Target II: 1, Final II: 1, Depth: 3.
INFO: [SCHED 204-111] Finished scheduling.
INFO: [HLS 200-111] Elapsed time: 2.13 seconds; current allocated memory: 60.078 MB.
INFO: [BIND 205-100] Starting micro-architecture generation ...
INFO: [BIND 205-101] Performing variable lifetime analysis.
INFO: [BIND 205-101] Exploring resource sharing.
INFO: [BIND 205-101] Binding ...
```

V2 Implementation Report

75

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency		Interval		
min	max	min	max	Type
435	435	436	436	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	~0%	~0%	~0%	~0%

V2 Implementation Loops

76

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
memcpy b	101	101	3	1	1	100	yes
memcpy c	101	101	3	1	1	100	yes
loop	107	107	9	1	1	100	yes
memcpy a	101	101	3	1	1	100	yes

V3 Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp

```

1 #include "string.h"
2
3 #define N 100
4
5 void myMemcpy(float *in, float*out){
6     for(int i = 0; i < N; i++){
7 #pragma HLS PIPELINE
8         out[i] = in[i];
9     }
10 }
11
12 void kernel(float *a, float *b, float *c, float d){
13
14 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
15 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
16 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
17
18 #pragma HLS INTERFACE s_axilite port=a
19 #pragma HLS INTERFACE s_axilite port=b
20 #pragma HLS INTERFACE s_axilite port=c
21 #pragma HLS INTERFACE s_axilite port=d
22 #pragma HLS INTERFACE s_axilite port=return
23
24     float a_local[N], b_local[N], c_local[N], d_local;
25
26     myMemcpy(b, b_local);
27     myMemcpy(c, c_local);
28     d_local = d;
29
30     loop:for(int i = 0; i < N; i++){
31 #pragma HLS PIPELINE
32         a_local[i] = b_local[i] + c_local[i]*d_local;
33     }
34
35     memcpy(a, a_local, sizeof(float)*N);
36
37 }
38

```

Outline Directive

- myMemcpy
- for Statement
- # HLS PIPELINE
- kernel
- # HLS INTERFACE s_axilite port=return
- a
- # HLS INTERFACE s_axilite port=a
- # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
- b
- # HLS INTERFACE s_axilite port=b
- # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
- c
- # HLS INTERFACE s_axilite port=c
- # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
- d
- # HLS INTERFACE s_axilite port=d
- a_local
- b_local
- c_local
- loop
- # HLS PIPELINE

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 38 : 1

V3 Implementation Report

78

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency		Interval		
min	max	min	max	Type
326	326	327	327	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	~0%	~0%	~0%	~0%

V3 Implementation Loops

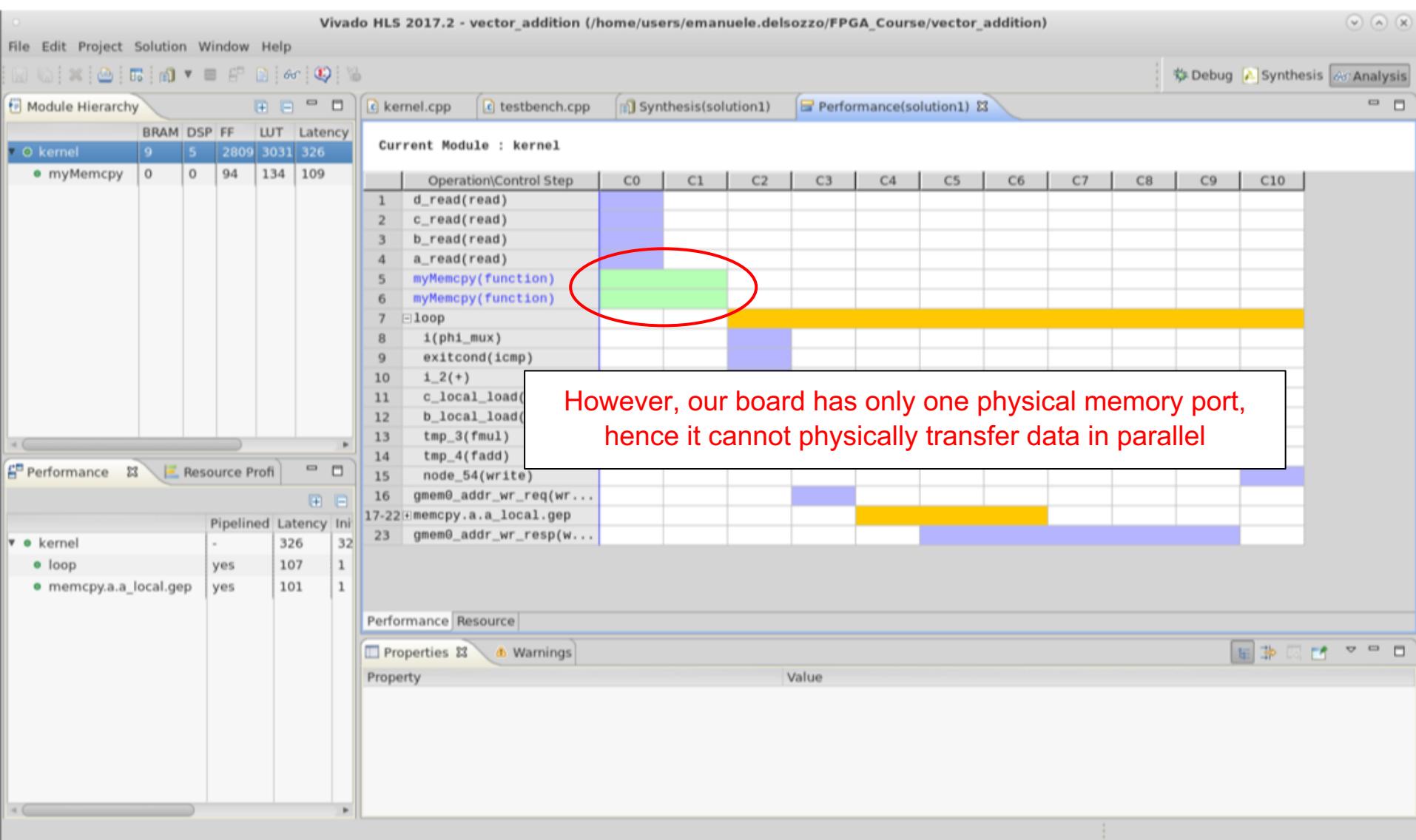
79

	Latency		Initiation Interval		
Module	min	max	min	max	Type
myMemcpy	109	109	109	109	None
myMemcpy	109	109	109	109	None

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
loop	107	107	9	1	1	100	yes
memcpy a	101	101	3	1	1	100	yes

V3 Implementation Analysis

80



V4 Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp Synthesis(solutio... Performance(soluti...

vector_addition

- Includes
- Source
 - kernel.cpp
- Test Bench
 - testbench.cpp
- solution1

```

1 #include "string.h"
2
3 #define N 100
4
5 void kernel(float *a, float *b, float *c, float d){
6
7 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
8 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
9 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
10
11 #pragma HLS INTERFACE s_axilite port=a
12 #pragma HLS INTERFACE s_axilite port=b
13 #pragma HLS INTERFACE s_axilite port=c
14 #pragma HLS INTERFACE s_axilite port=d
15 #pragma HLS INTERFACE s_axilite port=return
16
17     float a_local[N], b_local[N], c_local[N], d_local;
18
19     memcpy(b_local, b, sizeof(float)*N);
20     memcpy(c_local, c, sizeof(float)*N);
21     d_local = d;
22
23     loop:for(int i = 0; i < N; i++){
24 #pragma HLS UNROLL
25         a_local[i] = b_local[i] + c_local[i]*d_local;
26     }
27
28     memcpy(a, a_local, sizeof(float)*N);
29
30 }
31

```

Outline Directive Debug Synthesis Analysis

kernel

- # HLS INTERFACE s_axilite port=return
- a
- # HLS INTERFACE s_axilite port=a
- # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
- b
- # HLS INTERFACE s_axilite port=b
- # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
- c
- # HLS INTERFACE s_axilite port=c
- # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
- d
- # HLS INTERFACE s_axilite port=d
- a_local
- b_local
- c_local
- loop
- # HLS UNROLL

Console Errors Warnings

Vivado HLS Console

V4 Implementation Report

82

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency		Interval		
min	max	min	max	Type
373	373	374	374	none

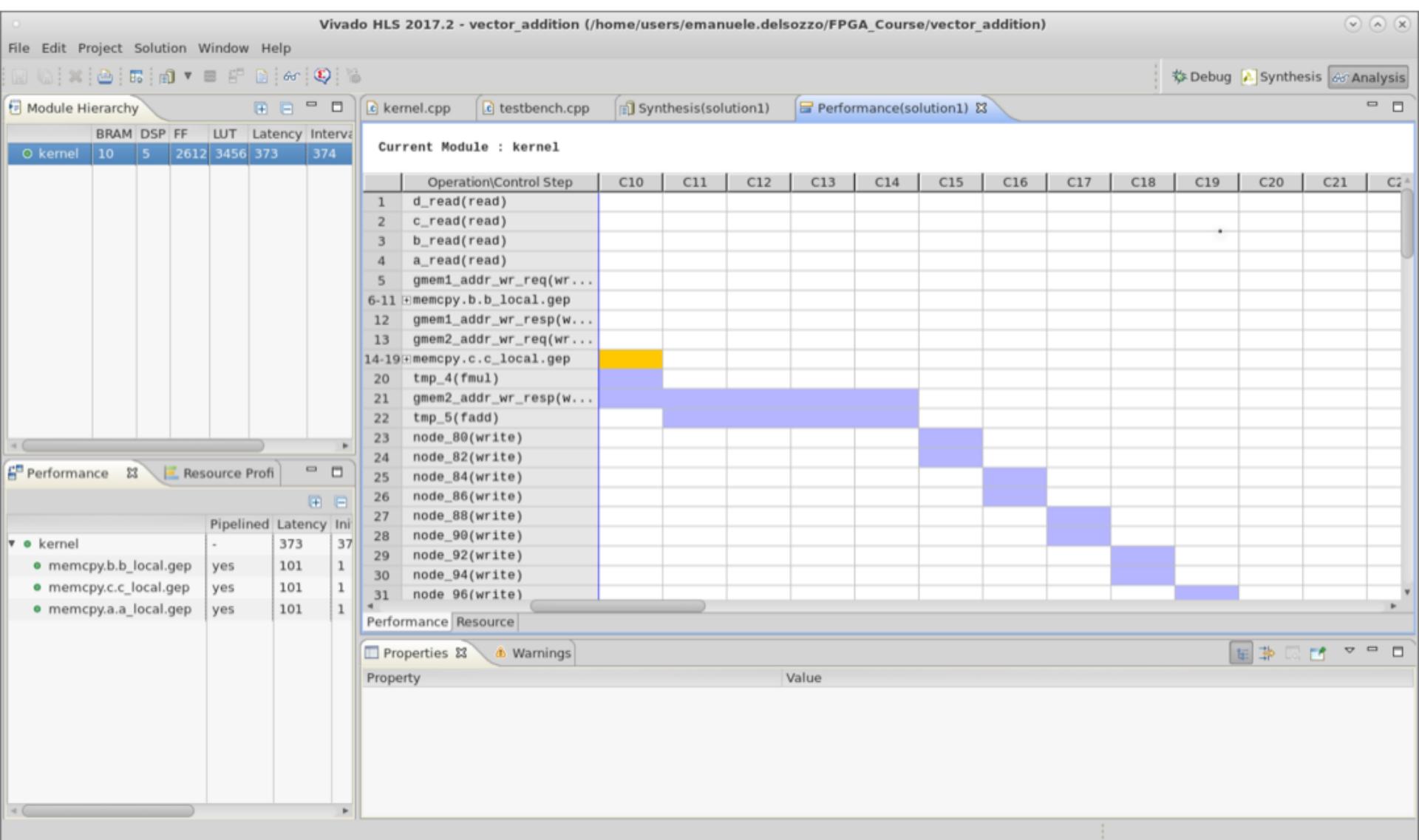
Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	~0%	~0%	~0%	1%

V4 Implementation Loops

83

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
memcpy b	101	101	3	1	1	100	yes
memcpy c	101	101	3	1	1	100	yes
memcpy a	101	101	3	1	1	100	yes

V4 Implementation Analysis



Insert Directive for a_local

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer X

- vector_addition
 - Includes
 - Source
 - kernel.cpp
 - Test Bench
 - testbench.cpp
- solution1

kernel.cpp X testbench.cpp Synthesis(solutio Performance(soluti

```

1 #include "string.h"
2
3 #define N 100
4
5 void kernel(float *a, float *b, float *c, float d){
6
7 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
8 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
9 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
10
11 #pragma HLS INTERFACE s_axilite port=a
12 #pragma HLS INTERFACE s_axilite port=b
13 #pragma HLS INTERFACE s_axilite port=c
14 #pragma HLS INTERFACE s_axilite port=d
15 #pragma HLS INTERFACE s_axilite port=return
16
17     float a_local[N], b_local[N], c_local[N], d_local;
18
19     memcpy(b_local, b, sizeof(float)*N);
20     memcpy(c_local, c, sizeof(float)*N);
21     d_local = d;
22
23     loop:for(int i = 0; i < N; i++){
24 #pragma HLS UNROLL
25         a_local[i] = b_local[i] + c_local[i]*d_local;
26     }
27
28     memcpy(a, a_local, sizeof(float)*N);
29
30 }
31

```

Outline X Directive X

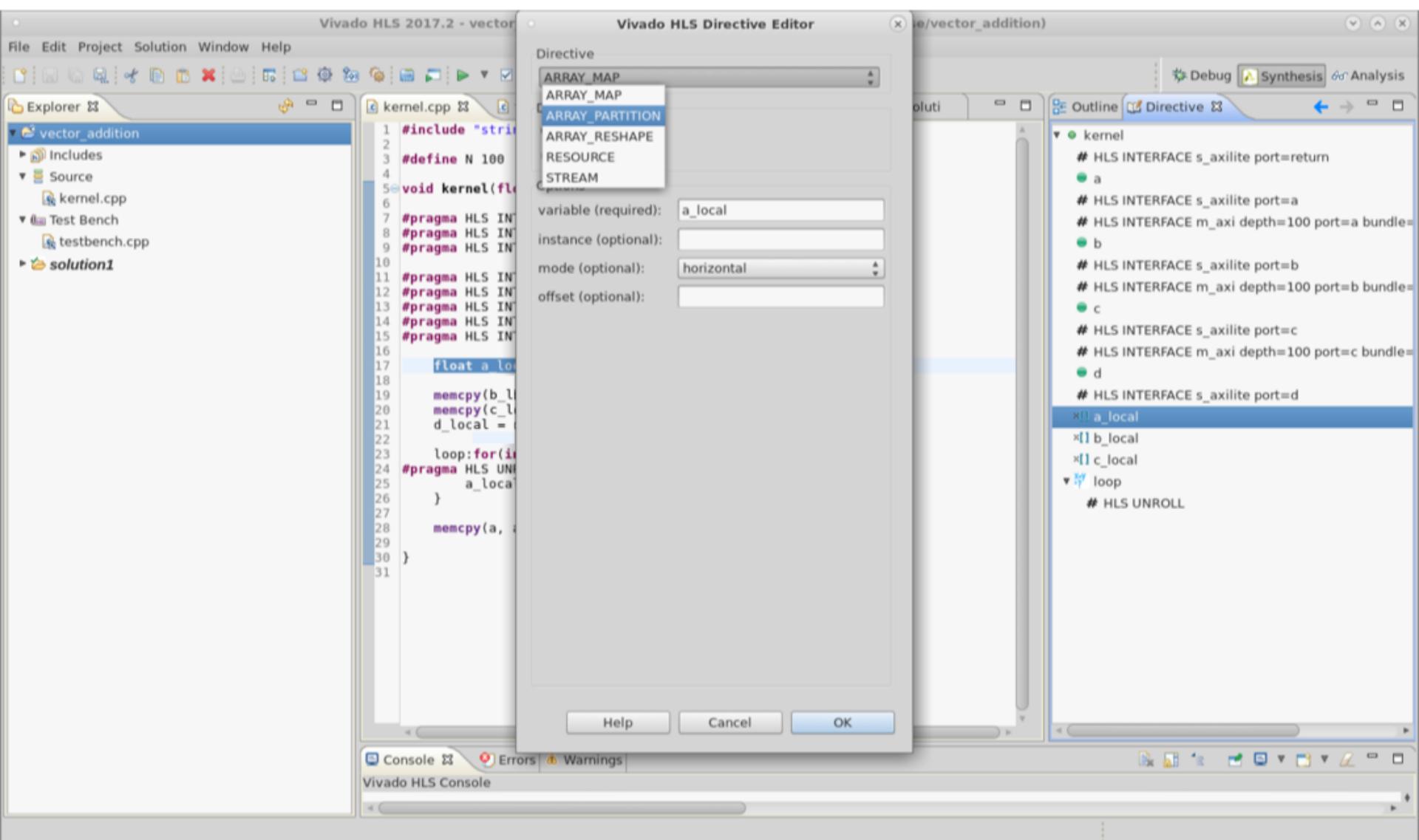
- kernel
 - # HLS INTERFACE s_axilite port=return
 - a
 - # HLS INTERFACE s_axilite port=a
 - # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
 - b
 - # HLS INTERFACE s_axilite port=b
 - # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
 - c
 - # HLS INTERFACE s_axilite port=c
 - # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
 - d
 - # HLS INTERFACE s_axilite port=d
 - a_local
 - b_local
 - c_local
 - loop
 - # HLS UNROLL

Insert Directive...

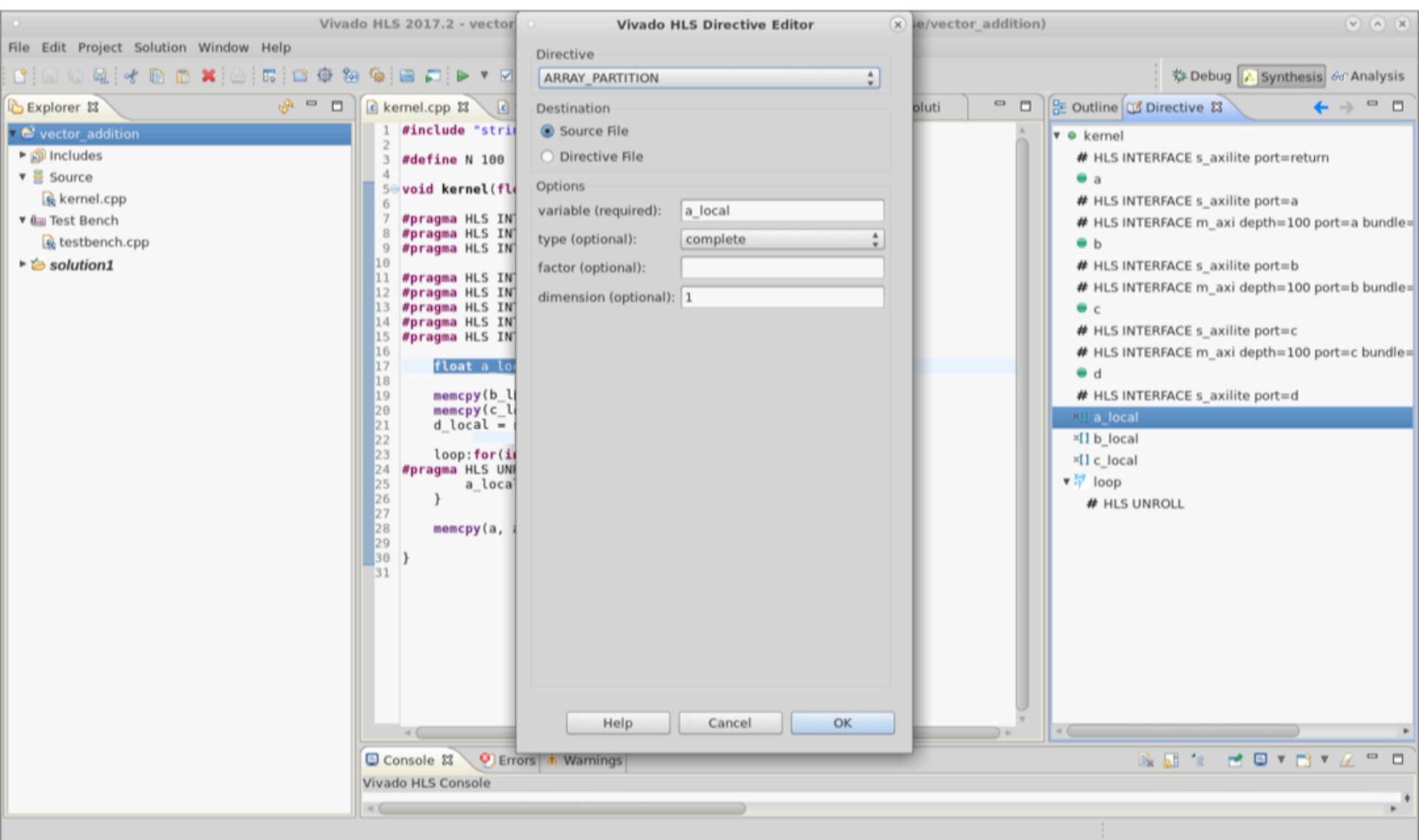
Console X Errors X Warnings X

Vivado HLS Console

Array Directives



Array Partition Directive



Array a_local directive

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp Synthesis(solutio... Performance(solut...

```

1 #include "string.h"
2
3 #define N 100
4
5 void kernel(float *a, float *b, float *c, float d){
6
7 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
8 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
9 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
10
11 #pragma HLS INTERFACE s_axilite port=a
12 #pragma HLS INTERFACE s_axilite port=b
13 #pragma HLS INTERFACE s_axilite port=c
14 #pragma HLS INTERFACE s_axilite port=d
15 #pragma HLS INTERFACE s_axilite port=return
16
17     float a_local[N], b_local[N], c_local[N], d_local;
18 #pragma HLS ARRAY_PARTITION variable=a_local complete dim=1
19
20     memcpy(b_local, b, sizeof(float)*N);
21     memcpy(c_local, c, sizeof(float)*N);
22     d_local = d;
23
24     loop:for(int i = 0; i < N; i++){
25 #pragma HLS UNROLL
26         a_local[i] = b_local[i] + c_local[i]*d_local;
27     }
28
29     memcpy(a, a_local, sizeof(float)*N);
30
31 }
32

```

Outline Directive

- kernel
 - # HLS INTERFACE s_axilite port=return
 - a
 - # HLS INTERFACE s_axilite port=a
 - # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
 - b
 - # HLS INTERFACE s_axilite port=b
 - # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
 - c
 - # HLS INTERFACE s_axilite port=c
 - # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
 - d
 - # HLS INTERFACE s_axilite port=d
 - a_local
 - # HLS ARRAY_PARTITION variable=a_local complete dim=1
 - b_local
 - c_local
 - loop
 - # HLS UNROLL

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 32 : 1

V5 Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp Synthesis(solut) kernel_csim.log

```

1 #include "string.h"
2
3 #define N 100
4
5 void kernel(float *a, float *b, float *c, float d){
6
7 #pragma HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
8 #pragma HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
9 #pragma HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
10
11 #pragma HLS INTERFACE s_axilite port=a
12 #pragma HLS INTERFACE s_axilite port=b
13 #pragma HLS INTERFACE s_axilite port=c
14 #pragma HLS INTERFACE s_axilite port=d
15 #pragma HLS INTERFACE s_axilite port=return
16
17     float a_local[N], b_local[N], c_local[N], d_local;
18 #pragma HLS ARRAY_PARTITION variable=a_local complete dim=1
19 #pragma HLS ARRAY_PARTITION variable=b_local complete dim=1
20 #pragma HLS ARRAY_PARTITION variable=c_local complete dim=1
21
22     memcpy(b_local, b, sizeof(float)*N);
23     memcpy(c_local, c, sizeof(float)*N);
24     d_local = d;
25
26     loop:for(int i = 0; i < N; i++){
27 #pragma HLS UNROLL
28         a_local[i] = b_local[i] + c_local[i]*d_local;
29     }
30
31     memcpy(a, a_local, sizeof(float)*N);
32
33 }
34

```

Outline Directive

kernel

- # HLS INTERFACE s_axilite port=return
- a
- # HLS INTERFACE s_axilite port=a
- # HLS INTERFACE m_axi depth=100 port=a bundle=gmem0
- b
- # HLS INTERFACE s_axilite port=b
- # HLS INTERFACE m_axi depth=100 port=b bundle=gmem1
- c
- # HLS INTERFACE s_axilite port=c
- # HLS INTERFACE m_axi depth=100 port=c bundle=gmem2
- d
- # HLS INTERFACE s_axilite port=d
- a_local
- # HLS ARRAY_PARTITION variable=a_local complete
- b_local
- # HLS ARRAY_PARTITION variable=b_local complete
- c_local
- # HLS ARRAY_PARTITION variable=c_local complete
- loop
- # HLS UNROLL

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 25 : 1

V5 Implementation Report

90

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency		Interval		
min	max	min	max	Type
320	320	321	321	none

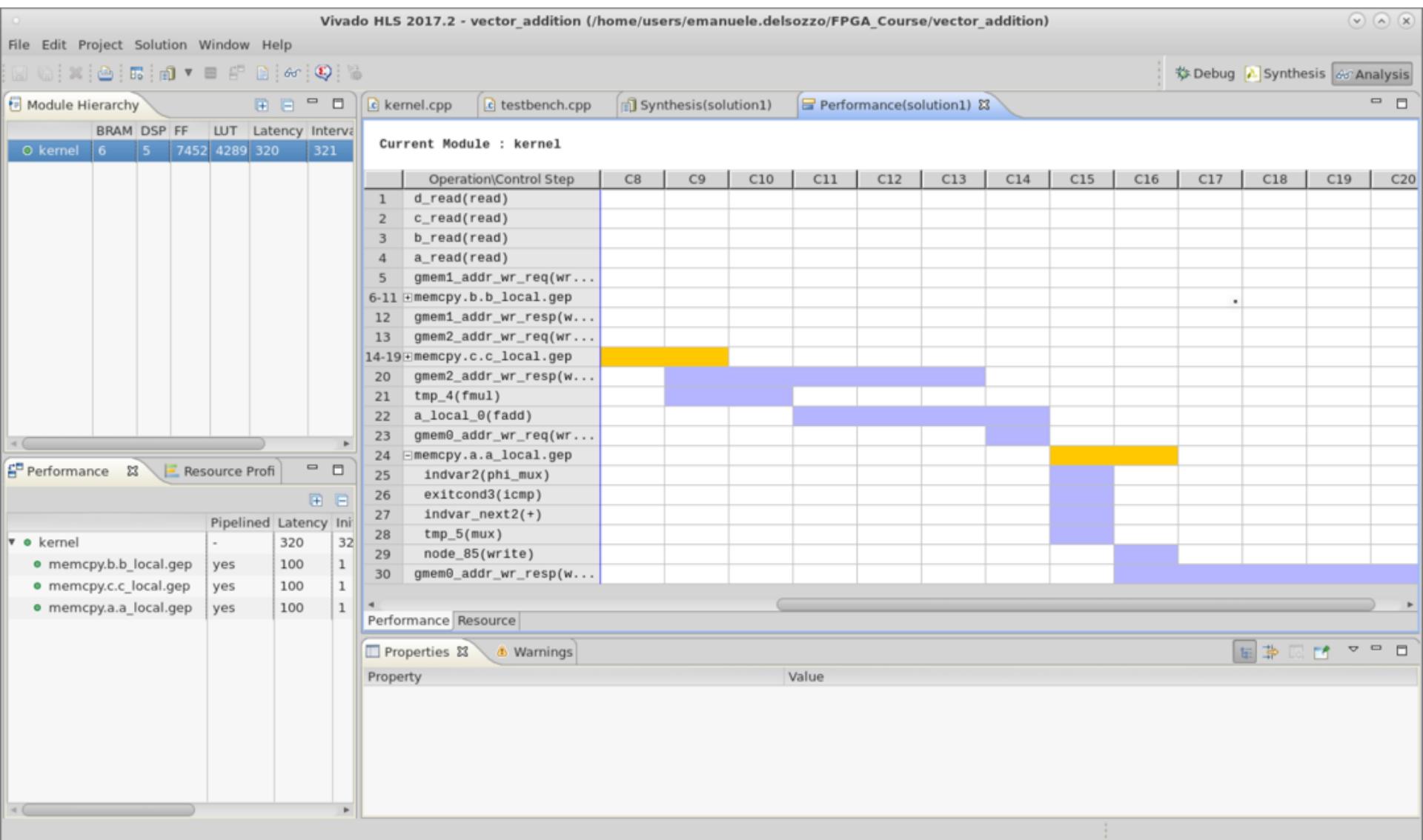
Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	~0%	17%	8%	12%

V5 Implementation Loops

91

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
memcpy b	101	101	3	1	1	100	yes
memcpy c	101	101	3	1	1	100	yes
memcpy a	101	101	3	1	1	100	yes

V5 Implementation Analysis



V5* Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delsozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp Synthesis(solut) kernel_csim.log 1

```

1 #include "string.h"
2
3 #define N 160
4
5 void kernel(float *a, float *b, float *c, float d){
6
7 #pragma HLS INTERFACE m_axi depth=160 port=a bundle=gmem0
8 #pragma HLS INTERFACE m_axi depth=160 port=b bundle=gmem1
9 #pragma HLS INTERFACE m_axi depth=160 port=c bundle=gmem2
10
11 #pragma HLS INTERFACE s_axilite port=a
12 #pragma HLS INTERFACE s_axilite port=b
13 #pragma HLS INTERFACE s_axilite port=c
14 #pragma HLS INTERFACE s_axilite port=d
15 #pragma HLS INTERFACE s_axilite port=return
16
17     float a_local[N], b_local[N], c_local[N], d_local;
18 #pragma HLS ARRAY_PARTITION variable=a_local complete dim=1
19 #pragma HLS ARRAY_PARTITION variable=b_local complete dim=1
20 #pragma HLS ARRAY_PARTITION variable=c_local complete dim=1
21
22     memcpy(b_local, b, sizeof(float)*N);
23     memcpy(c_local, c, sizeof(float)*N);
24     d_local = d;
25
26     loop:for(int i = 0; i < N; i++){
27 #pragma HLS UNROLL
28         a_local[i] = b_local[i] + c_local[i]*d_local;
29     }
30
31     memcpy(a, a_local, sizeof(float)*N);
32
33 }
34

```

Debug Synthesis Analysis

Outline Directive

kernel

- # HLS INTERFACE s_axilite port=return
- a
- # HLS INTERFACE s_axilite port=a
- # HLS INTERFACE m_axi depth=160 port=a bundle=gmem0
- b
- # HLS INTERFACE s_axilite port=b
- # HLS INTERFACE m_axi depth=160 port=b bundle=gmem1
- c
- # HLS INTERFACE s_axilite port=c
- # HLS INTERFACE m_axi depth=160 port=c bundle=gmem2
- d
- # HLS INTERFACE s_axilite port=d
- a_local
- # HLS ARRAY_PARTITION variable=a_local complete
- b_local
- # HLS ARRAY_PARTITION variable=b_local complete
- c_local
- # HLS ARRAY_PARTITION variable=c_local complete
- loop
- # HLS UNROLL

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 34 : 1

V5* Implementation Report

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency		Interval		
min	max	min	max	Type
508	508	509	509	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	~0%	28%	13%	19%

V5* Implementation Loops

95

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
memcpy b	160	160	2	1	1	100	yes
memcpy c	160	160	2	1	1	100	yes
memcpy a	160	160	2	1	1	100	yes

V6 Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp testbench.cpp kernel_csim.log Synthesis(solution1)

```

#include "string.h"
#include "ap_int.h"
#define N 160
#define N_AP (160/16)

void kernel(ap_uint<512> *a, ap_uint<512> *b, ap_uint<512> *c, float d){
#pragma HLS INTERFACE m_axi depth=10 port=a bundle=gmem0
#pragma HLS INTERFACE m_axi depth=10 port=b bundle=gmem1
#pragma HLS INTERFACE m_axi depth=10 port=c bundle=gmem2

#pragma HLS INTERFACE s_axilite port=a
#pragma HLS INTERFACE s_axilite port=b
#pragma HLS INTERFACE s_axilite port=c
#pragma HLS INTERFACE s_axilite port=d
#pragma HLS INTERFACE s_axilite port=return

ap_uint<512> a_local[N_AP], b_local[N_AP], c_local[N_AP];
float d_local;

memcpy(b_local, b, sizeof(ap_uint<512>)*N_AP);
memcpy(c_local, c, sizeof(ap_uint<512>)*N_AP);
d_local = d;

loop1:for(int i = 0; i < N_AP; i++){
#pragma HLS PIPELINE
    loop2:for(int j = 0; j < 16; j++){
        int upper = (j+1)*32 - 1;
        int lower = j*32;
        unsigned int b_tmp = b_local[i].range(upper, lower);
        unsigned int c_tmp = c_local[i].range(upper, lower);
        float b_val = *((float *)&b_tmp);
        float c_val = *((float *)&c_tmp);
        float a_val = b_val + c_val*d_local;
        a_local[i].range(upper, lower) = *((unsigned int *)&a_val);
    }
}
memcpy(a, a_local, sizeof(ap_uint<512>)*N_AP);
}
```

Outline Directive

- kernel
 - # HLS INTERFACE s_axilite port=return
 - a
 - # HLS INTERFACE s_axilite port=a
 - # HLS INTERFACE m_axi depth=10 port=a bundle=gmem0
 - b
 - # HLS INTERFACE s_axilite port=b
 - # HLS INTERFACE m_axi depth=10 port=b bundle=gmem1
 - c
 - # HLS INTERFACE s_axilite port=c
 - # HLS INTERFACE m_axi depth=10 port=c bundle=gmem2
 - d
 - # HLS INTERFACE s_axilite port=d
 - a_local
 - b_local
 - c_local
 - loop1
 - # HLS PIPELINE
 - loop2

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 41 : 1

V6 Implementation Report

97

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.75	1.25

Latency		Interval		
min	max	min	max	Type
75	75	76	76	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	6%	2%	2%	3%

V6 Implementation Loops

98

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
memcpy b	11	11	3	1	1	10	yes
memcpy c	11	11	3	1	1	10	yes
loop1	17	17	9	1	1	10	yes
memcpy a	11	11	3	1	1	10	yes

V7 Implementation

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp kernel.hpp

```

1 #include "kernel.hpp"
2
3 void kernel(myStream &a, myStream &b, myStream &c, myStream &d){
4 #pragma HLS INTERFACE axis register both port=a
5 #pragma HLS INTERFACE axis register both port=b
6 #pragma HLS INTERFACE axis register both port=c
7 #pragma HLS INTERFACE axis register both port=d
8
9 #pragma HLS INTERFACE ap_ctrl_none port=return
10 #pragma DATAFLOW
11
12     streamType d_tmp = d.read();
13     float d_val = d_tmp.data;
14
15     loop:for(int i = 0; i < N; i++){
16 #pragma HLS PIPELINE
17         streamType b_tmp = b.read();
18         float b_val = b_tmp.data;
19         streamType c_tmp = c.read();
20         float c_val = c_tmp.data;
21         float a_val = b_val + c_val * d_val;
22         streamType a_tmp;
23         a_tmp.data = a_val;
24         a_tmp.last = i == N - 1;
25         a.write(a_tmp);
26     }
27
28 }
29

```

Outline Directive

- kernel
 - # HLS INTERFACE ap_ctrl_none port=return
 - # DATAFLOW
 - a
 - # HLS INTERFACE axis register both port=a
 - b
 - # HLS INTERFACE axis register both port=b
 - c
 - # HLS INTERFACE axis register both port=c
 - d
 - # HLS INTERFACE axis register both port=d
 - d_tmp
- loop
 - # HLS PIPELINE
 - b_tmp
 - c_tmp
 - a_tmp

Console Errors Warnings

Vivado HLS Console

Streaming Interface

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer □

- vector_addition
 - Includes
 - Source
 - kernel.cpp
 - kernel.hpp
 - old.cpp
 - Test Bench
 - testbench.cpp
 - solution1

kernel.cpp kernel.hpp

```
1 #ifndef KERNEL_HPP
2 #define KERNEL_HPP
3
4 #define N 160
5 #include "hls_stream.h"
6 #include "ap_int.h"
7
8 template<class DT, int D,int U,int TI,int TD>
9 struct my_ap_axis{
10     DT      data;
11     ap_uint<(D+7)/8> keep;
12     ap_uint<(D+7)/8> strb;
13     ap_uint<U>       user;
14     ap_uint<1>       last;
15     ap_uint<TI>      id;
16     ap_uint<TD>      dest;
17 };
18
19 typedef my_ap_axis<float, 32, 1, 1, 1> streamType;
20 typedef hls::stream<streamType> myStream;
21
22 #endif
23 |
```

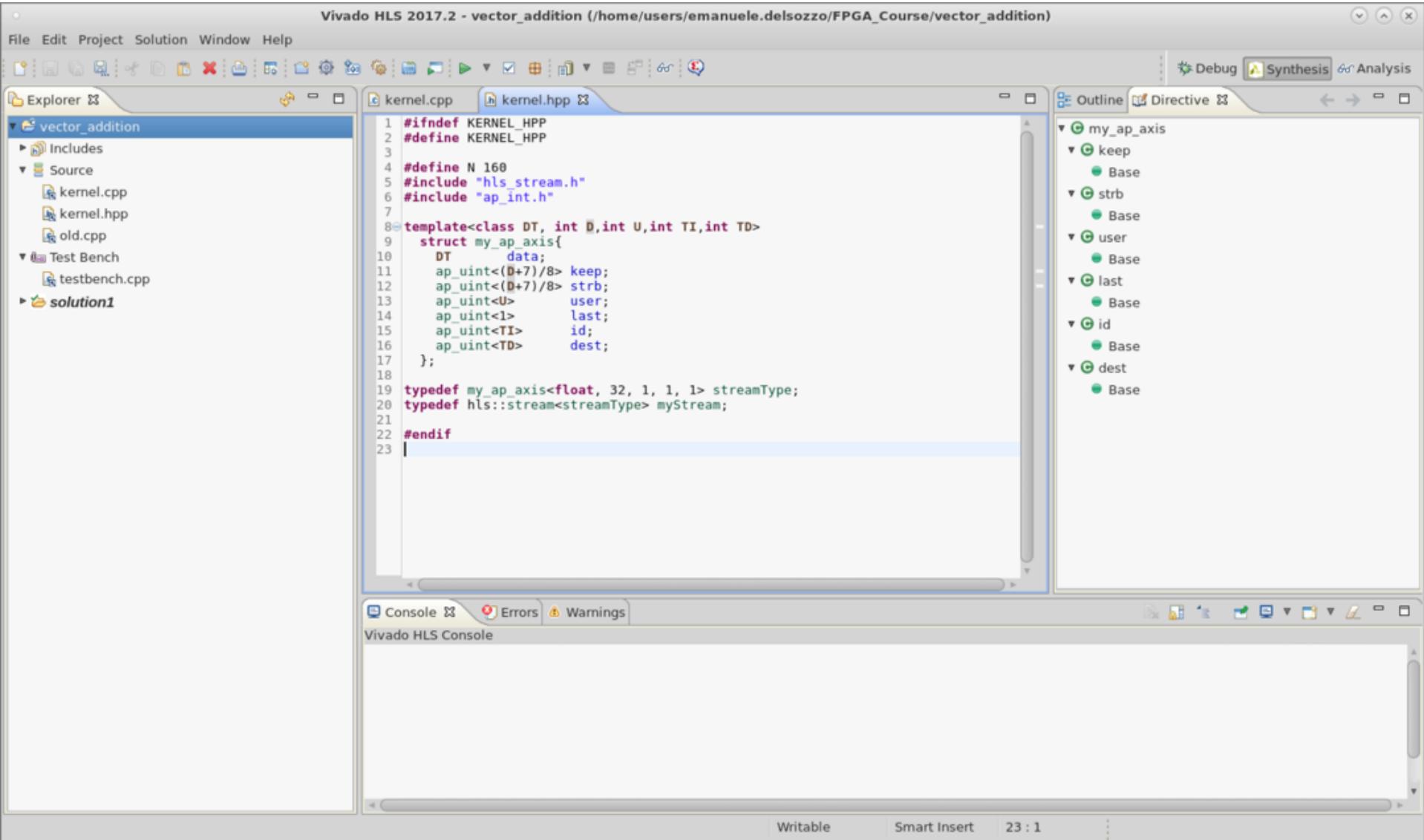
Outline Directive

- my_ap_axis
 - keep
 - Base
 - strb
 - Base
 - user
 - Base
 - last
 - Base
 - id
 - Base
 - dest
 - Base

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 23 : 1



V7 Testbench

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Debug Synthesis Analysis

```

kernel.cpp kernel.hpp *testbench.cpp
3 #include "kernel.hpp"
4 #define X 20
5
6 void kernel(myStream &a, myStream &b, myStream &c, myStream &d);
7
8 int main(){
9
10    float aCPU[N], aFPGA[N], b[N], c[N], d;
11    myStream aStream, bStream, cStream, dStream;
12    streamType aType, bType, cType, dType;
13
14    srand(0);
15    d = (float) rand() / ((float) (RAND_MAX/X));
16    for(int i = 0; i < N; i++){
17        b[i] = (float) rand() / ((float) (RAND_MAX/X));
18        c[i] = (float) rand() / ((float) (RAND_MAX/X));
19    }
20    for(int i = 0; i < N; i++){
21        aCPU[i] = b[i] + c[i]*d;
22    }
23    for(int i = 0; i < N; i++){
24        bType.data = b[i];
25        bType.last = i == N - 1;
26        bStream.write(bType);
27        cType.data = c[i];
28        cType.last = i == N - 1;
29        cStream.write(cType);
30    }
31    dType.data = d;
32    dType.last = 1;
33    dStream.write(dType);
34
35    kernel(aStream, bStream, cStream, dStream);
36
37    for(int i = 0; i < N; i++){
38        aType = aStream.read();
39        aFPGA[i] = aType.data;
40        if(aCPU[i] != aFPGA[i]){
41            printf("Error at index %d: %f != %f\n", i, aCPU[i], aFPGA[i]);
42            return 1;
43        }
44    }
45
46    return 0;
47
48 }
```

Writable Smart Insert 45 : 1

V7 Implementation Report

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.46	1.25

Latency		Interval		
min	max	min	max	Type
169	169	170	170	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	0%	~0%	~0%	~0%

V7 Implementation Loops

103

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
loop	166	166	8	1	1	160	yes

V8 Implementation

104

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer kernel.cpp kernel.hpp

```
1 #include "kernel.hpp"
2
3 void kernel(myStreamAP &a, myStreamAP &b, myStreamAP &c, myStream &d){
4 #pragma HLS INTERFACE axis register both port=a
5 #pragma HLS INTERFACE axis register both port=b
6 #pragma HLS INTERFACE axis register both port=c
7 #pragma HLS INTERFACE axis register both port=d
8
9 #pragma HLS INTERFACE ap_ctrl_none port=return
# pragma DATAFLOW
10
11     streamType d_tmp = d.read();
12     float d_val = d_tmp.data;
13
14     loop:for(int i = 0; i < N_AP; i++){
15 #pragma HLS PIPELINE
16         ap_uint<512> a_val;
17         streamTypeAP b_tmp = b.read();
18         ap_uint<512> b_val = b_tmp.data;
19         streamTypeAP c_tmp = c.read();
20         ap_uint<512> c_val = c_tmp.data;
21         for(int j = 0; j < 16; j++){
22             int upper = (j+1)*32 - 1;
23             int lower = j*32;
24             unsigned int b_tmp_uint = b_val.range(upper, lower);
25             unsigned int c_tmp_uint = c_val.range(upper, lower);
26             float b_valF = *((float *)&b_tmp_uint);
27             float c_valF = *((float *)&c_tmp_uint);
28             float a_valF = b_valF + c_valF*d_val;
29             a_val.range(upper, lower) = *((unsigned int *)&a_valF);
30         }
31         streamTypeAP a_tmp;
32         a_tmp.data = a_val;
33         a_tmp.last = i == N - 1;
34         a.write(a_tmp);
35     }
36 }
37
38 }
```

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 39 : 1

Outline Directive Debug Synthesis Analysis

kernel

- # HLS INTERFACE ap_ctrl_none port=return
- # DATAFLOW
- a
- # HLS INTERFACE axis register both port=a
- b
- # HLS INTERFACE axis register both port=b
- c
- # HLS INTERFACE axis register both port=c
- d
- # HLS INTERFACE axis register both port=d
- d_tmp
- loop
- # HLS PIPELINE
- a_val
 - Base
 - b_tmp
- b_val
 - Base
 - c_tmp
- c_val
 - Base
- for Statement
- a_tmp

V8 Streaming Interface

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer □ kernel.cpp kernel.hpp

```
1 #ifndef KERNEL_HPP
2 #define KERNEL_HPP
3
4 #define N 160
5 #define N_AP (N/16)
6 #include "hls_stream.h"
7 #include "ap_int.h"
8
9 template<class DT, int D,int U,int TI,int TD>
10 struct my_ap_axis{
11     DT data;
12     ap_uint<(D+7)/8> keep;
13     ap_uint<(D+7)/8> strb;
14     ap_uint<U> user;
15     ap_uint<1> last;
16     ap_uint<TI> id;
17     ap_uint<TD> dest;
18 };
19
20 typedef my_ap_axis<float, 32, 1, 1, 1> streamType;
21 typedef hls::stream<streamType> myStream;
22 typedef ap_uint<512> ap_512;
23 typedef my_ap_axis<ap_512, 512, 1, 1, 1> streamTypeAP;
24 typedef hls::stream<streamTypeAP> myStreamAP;
25
26 #endif
27
```

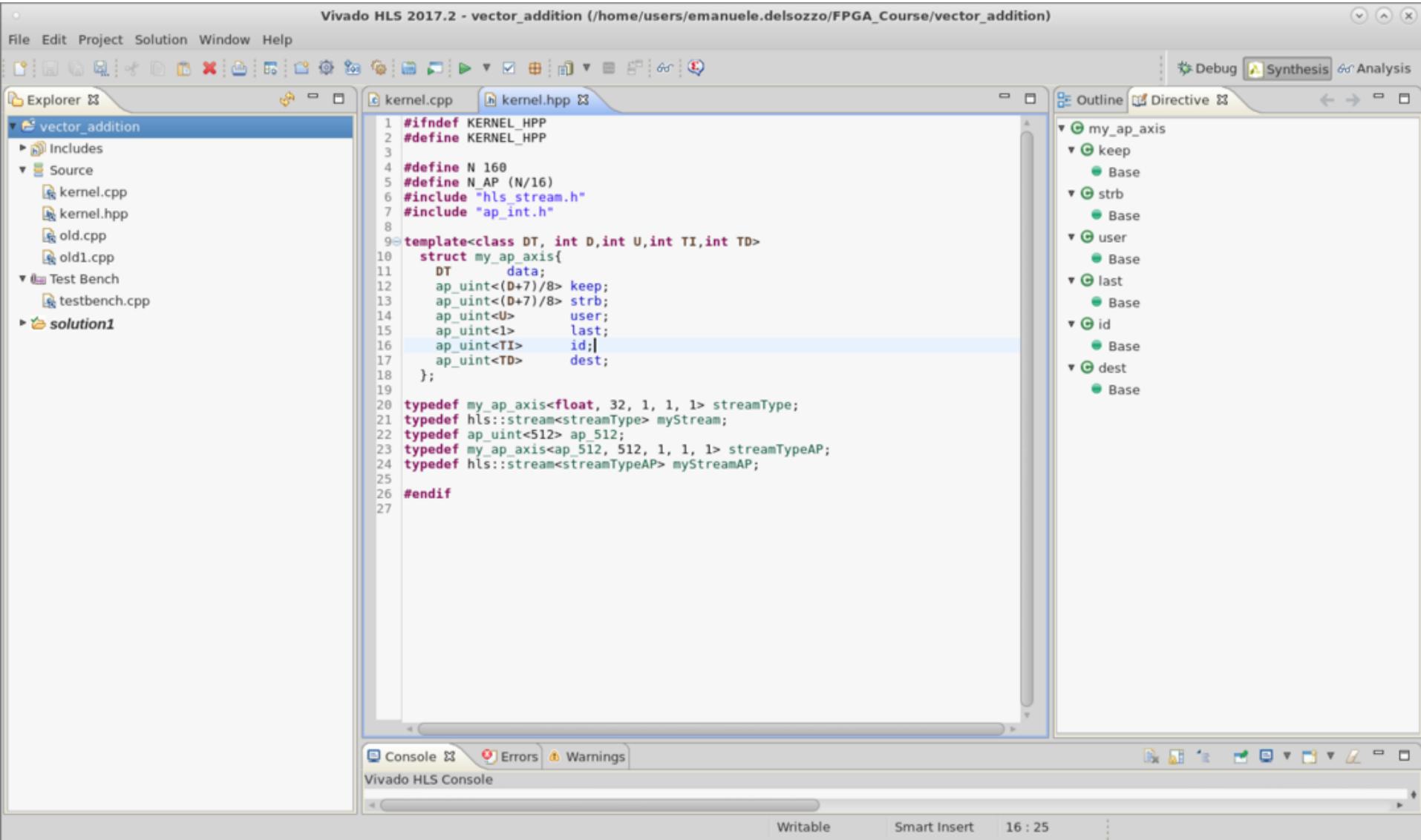
Outline Directive

- my_ap_axis
 - keep
 - Base
 - strb
 - Base
 - user
 - Base
 - last
 - Base
 - id
 - Base
 - dest
 - Base

Console □ Errors □ Warnings

Vivado HLS Console

Writable Smart Insert 16 : 25



V8 Testbench

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Debug Synthesis Analysis

kernel.cpp kernel.hpp testbench.cpp

```

5 void kernel(myStreamAP &a, myStreamAP &b, myStreamAP &c, myStream &d);
6
7
8 int main(){
9
10    float aCPU[N], aFPGA[N], b[N], c[N], d;
11    myStreamAP aStream, bStream, cStream;
12    myStream dStream;
13    streamType dType;
14    streamTypeAP aType, bType, cType;
15
16    srand(0);
17    d = (float) rand() / ((float) (RAND_MAX/X));
18    for(int i = 0; i < N; i++){
19        b[i] = (float) rand() / ((float) (RAND_MAX/X));
20        c[i] = (float) rand() / ((float) (RAND_MAX/X));
21    }
22    for(int i = 0; i < N; i++){
23        aCPU[i] = b[i] + c[i]*d;
24    }
25    for(int i = 0; i < N_AP; i++){
26        ap_uint<512> bAP;
27        ap_uint<512> cAP;
28        for(int j = 0; j < 16; j++){
29            int upper = (j+1)*32 - 1;
30            int lower = j*32;
31            float bVal = b[i*16 + j];
32            float cVal = c[i*16 + j];
33            bAP.range(upper, lower) = *((unsigned int *)&bVal);
34            cAP.range(upper, lower) = *((unsigned int *)&cVal);
35        }
36        bType.data = bAP;
37        bType.last = i == N_AP - 1;
38        bStream.write(bType);
39        cType.data = cAP;
40        cType.last = i == N_AP - 1;
41        cStream.write(cType);
42    }
43    dType.data = d;
44    dType.last = 1;
45    dStream.write(dType);
46
47    kernel(aStream, bStream, cStream, dStream);
48
49    for(int i = 0; i < N_AP; i++){
50        aType = aStream.read();
51    }

```

Writable Smart Insert 38 : 30

V8 Implementation Report

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.46	1.25

Latency		Interval		
min	max	min	max	Type
19	19	20	20	none

Name	BRAM_18K	DSP48E	FF	LUT
Available	2060	2800	2393	2618
Utilization	0%	2%	1%	1%

V8 Implementation Loops

108

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
loop	16	16	8	1	1	10	yes

Export IP

109

Vivado HLS 2017.2 - vector_addition (/home/users/emanuele.delozzo/FPGA_Course/vector_addition)

File Edit Project Solution Window Help

Explorer           

kernel.cpp kernel.hpp testbench.cpp Synthesis(solution1)

vector_addition

Includes

Source

kernel.cpp

kernel.hpp

old.cpp

old1.cpp

Test Bench

testbench.cpp

solution1

#include "kernel.hpp"

void kernel(myStreamAP &a, myStreamAP &b, myStreamAP &c, myStream &d){

#pragma HLS INTERFACE axis register both port=a

#pragma HLS INTERFACE axis register both port=b

#pragma HLS INTERFACE axis register both port=c

#pragma HLS INTERFACE axis register both port=d

#pragma HLS INTERFACE ap_ctrl_none port=return

#pragma DATAFLOW

streamType d_tmp = d.read();

float d_val = d_tmp.data;

loop:for(int i = 0; i < N_AP; i++){

#pragma HLS PIPELINE

ap_uint<512> a_val;

streamTypeAP b_tmp = b.read();

ap_uint<512> b_val = b_tmp.data;

streamTypeAP c_tmp = c.read();

ap_uint<512> c_val = c_tmp.data;

for(int j = 0; j < 16; j++){

int upper = (j+1)*32 - 1;

int lower = j*32;

unsigned int b_tmp_uint = b_val.range(upper, lower);

unsigned int c_tmp_uint = c_val.range(upper, lower);

float b_valF = *((float *)&b_tmp_uint);

float c_valF = *((float *)&c_tmp_uint);

float a_valF = b_valF + c_valF*d_val;

a_val.range(upper, lower) = *((unsigned int *)&a_valF);

}

streamTypeAP a_tmp;

a_tmp.data = a_val;

Console Errors Warnings

Vivado HLS Console

Writable Smart Insert 5 : 48

Outline Directive   

kernel

HLS INTERFACE axis register both port=b%

HLS INTERFACE ap_ctrl_none port=return

DATAFLOW

a

HLS INTERFACE axis register both port=a

b

c

HLS INTERFACE axis register both port=c

d

HLS INTERFACE axis register both port=d

d_tmp

loop

HLS PIPELINE

a_val

Base

b_tmp

b_val

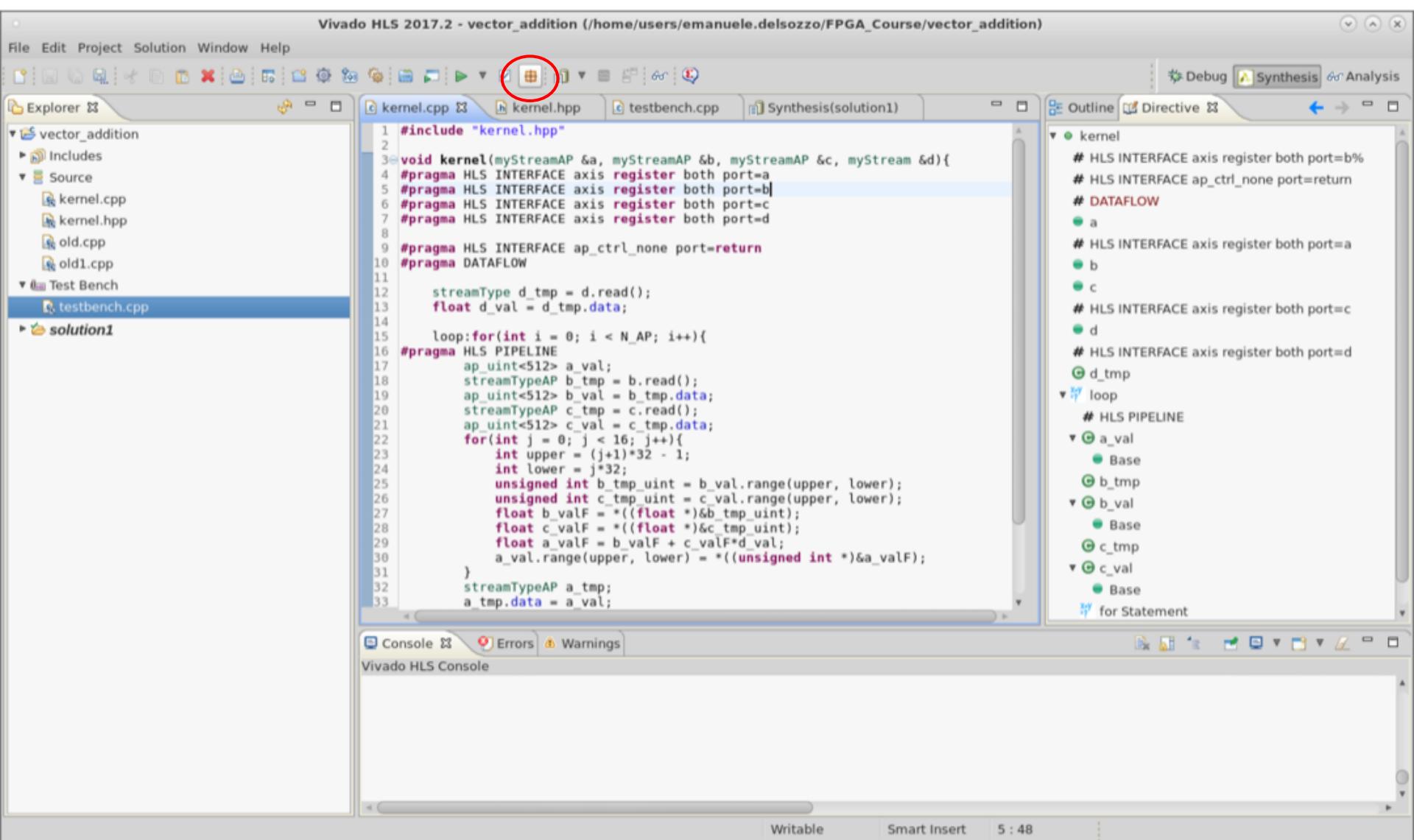
Base

c_tmp

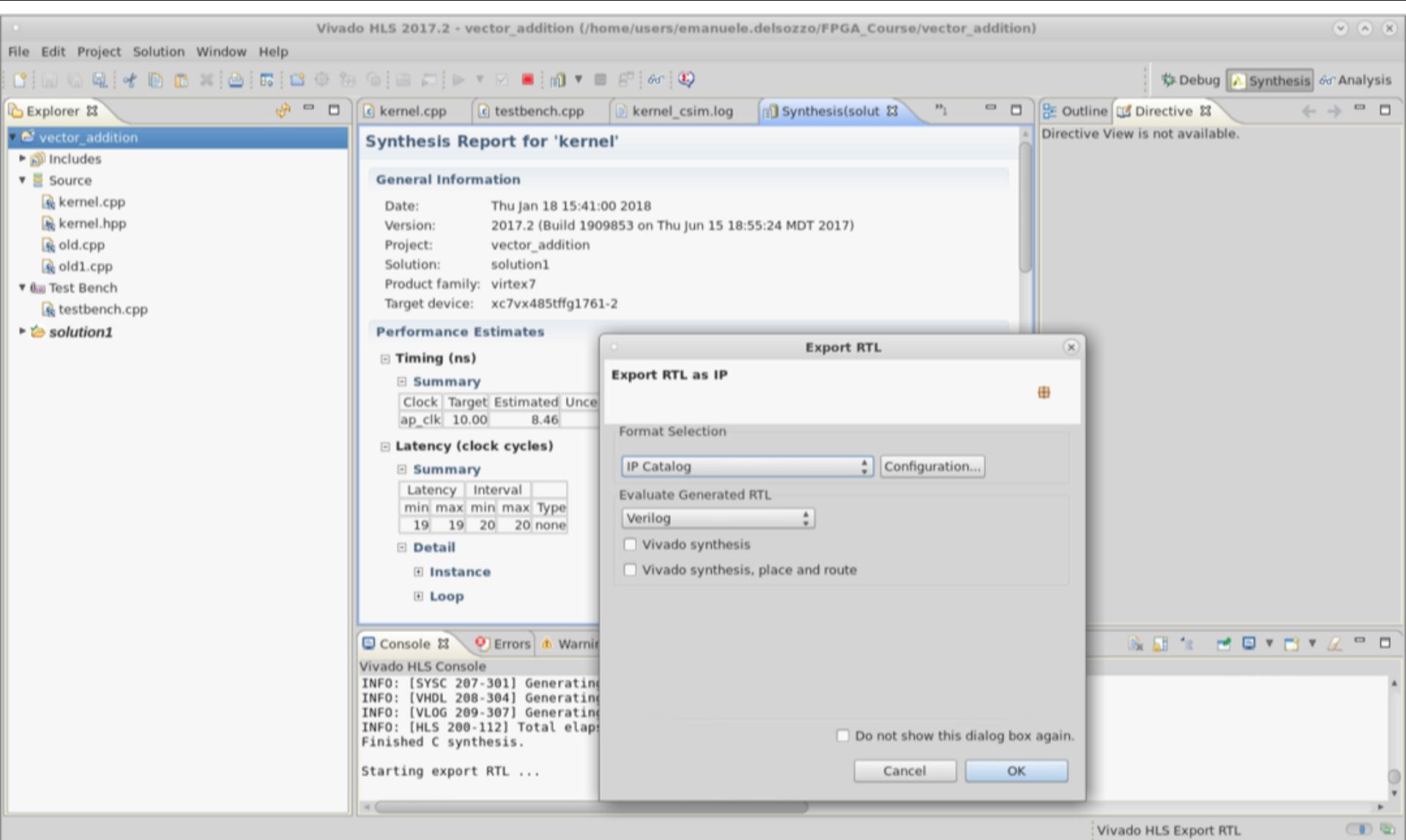
c_val

Base

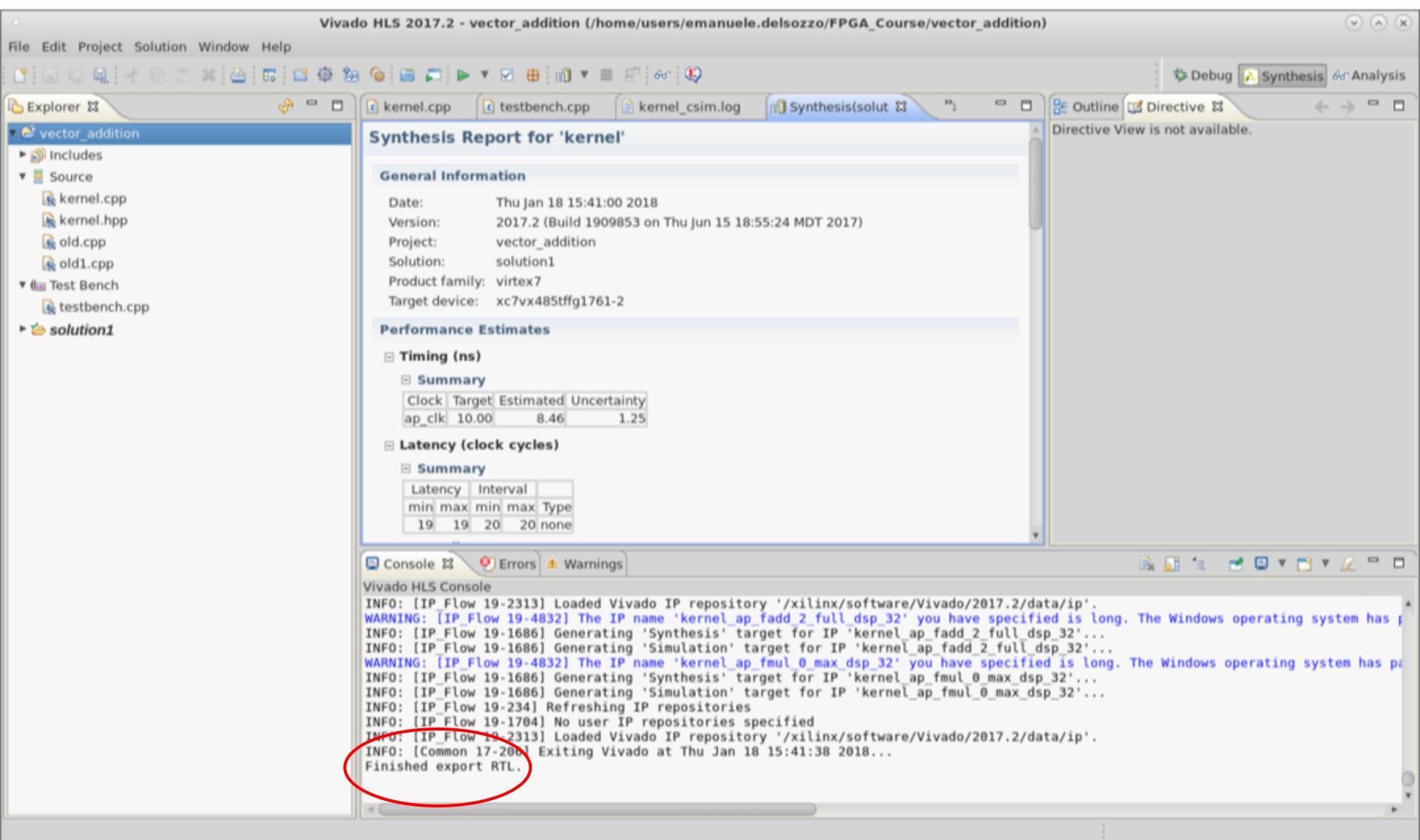
for Statement



Export IP Settings



Export IP Done



This is only the beginning!!

112

For more information, read Vivado HLS manual

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_2/ug902-vivado-high-level-synthesis.pdf

Feedbacks

- We are working at improving this course, would you share your feedback for this lesson?

<https://goo.gl/tLcWQj>



Thank You for the Attention!



Lorenzo Di Tucci

lorenzo.ditucci@polimi.it

Emanuele Del Sozzo

emanuele.delsozzo@polimi.it

Marco D. Santambrogio

marco.santambrogio@polimi.it