Turun yliopisto
University of Turku

EXERCISE 5
DESIGNING WITH VITIS HLS: C++ VS. VHDL

OBJECTIVES

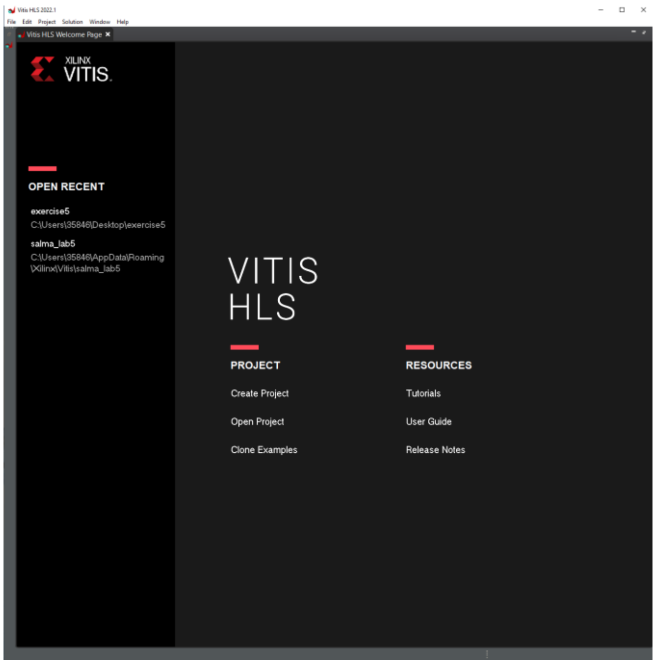
Upon completion of this lab, you will be able to:

- ✓ Create a new project using Vitis HLS
- ✓ Simulate a design
- ✓ Synthesize a design
- ✓ Implement a design
- ✓ Perform design analysis within Vitis HLS

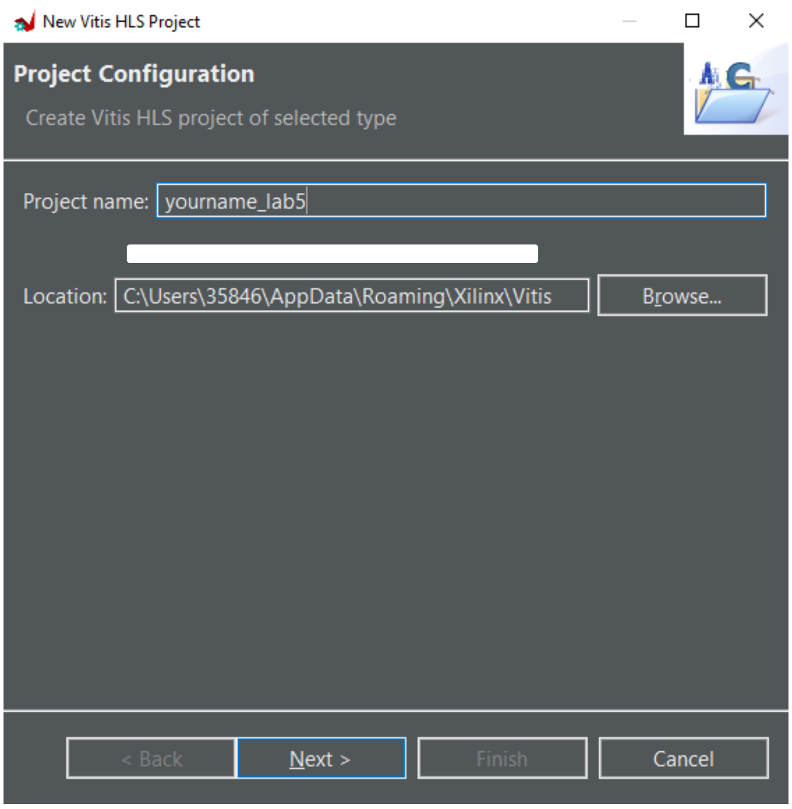
NEW PROJECT AND SIMULATION

Follow the instructions below to create and simulate a project in Vitis HLS.

1. Create a new project in Vitis HLS



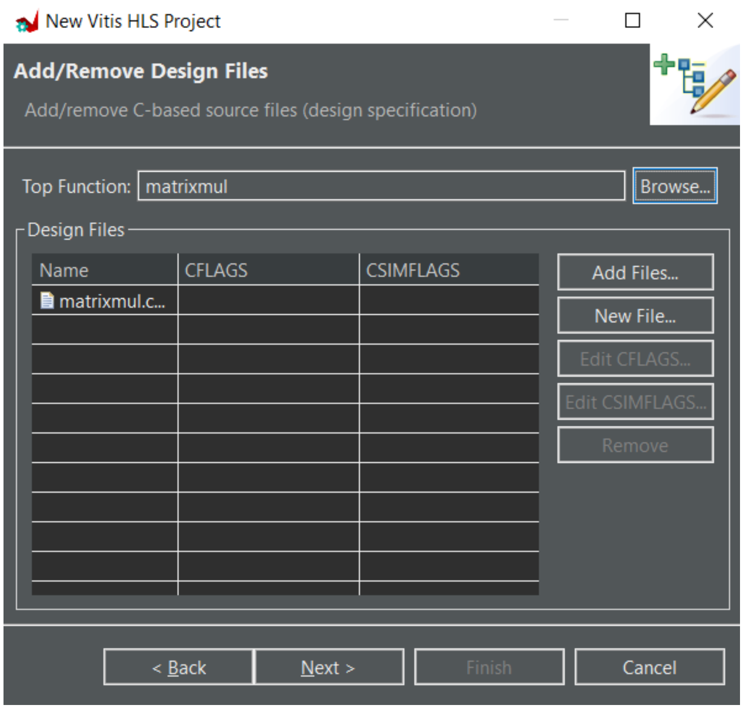
2. For the project name, type *yourname_lab5*



Project name: yourname_lab5

Location: C:\Users\35846\AppData\Roaming\Xilinx\Vitis

3. In the Add/Remove Files window, type matrixmul as the Top Function name (the provided source file contains the function, called matrixmul, to be synthesized). Then add the matrixmul.cpp file from the lab1 folder in moodle.


New Vitis HLS Project

Add/Remove Design Files

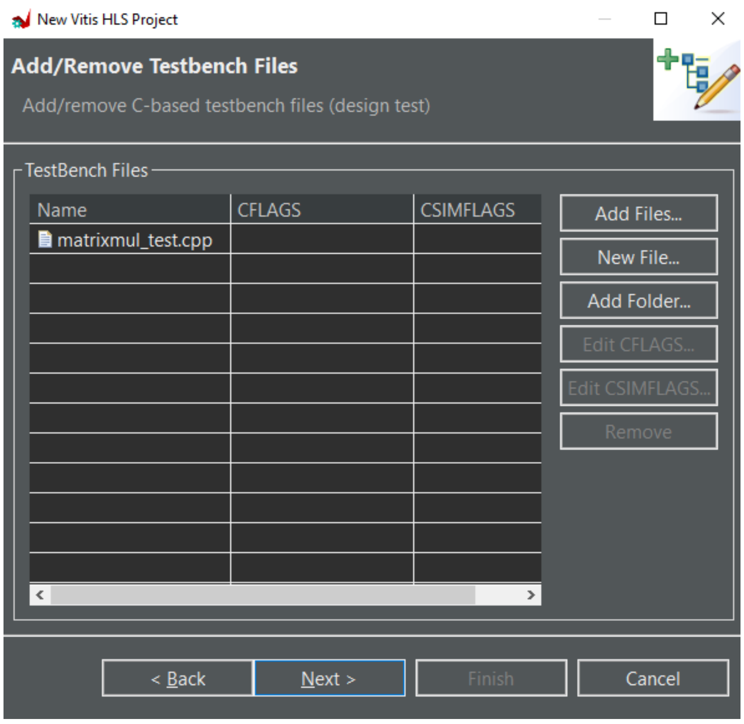
Add/remove C-based source files (design specification)

Top Function: matrixmul

Design Files

Name	CFLAGS	CSIMFLAGS
matrixmul.c...		

4. Click *Next*, then again the *Add Files…* button, select matrixmul_test.cpp file from the lab1 folder in moodle.


New Vitis HLS Project

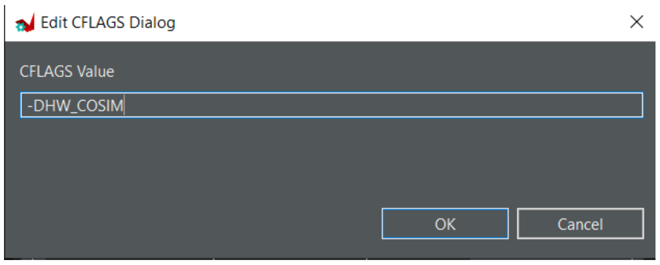
Add/Remove Testbench Files

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

TestBench Files

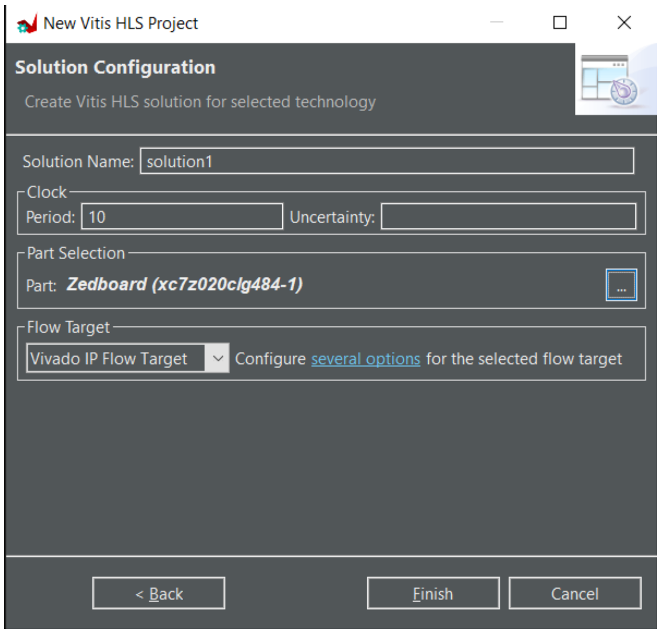
Name	CFLAGS	CSIMFLAGS
matrixmul_test.cpp		

5. Click the *Edit CFLAG…* button, type -DHW_COSIM, and click OK (this defines a compiler flag that will be used later)



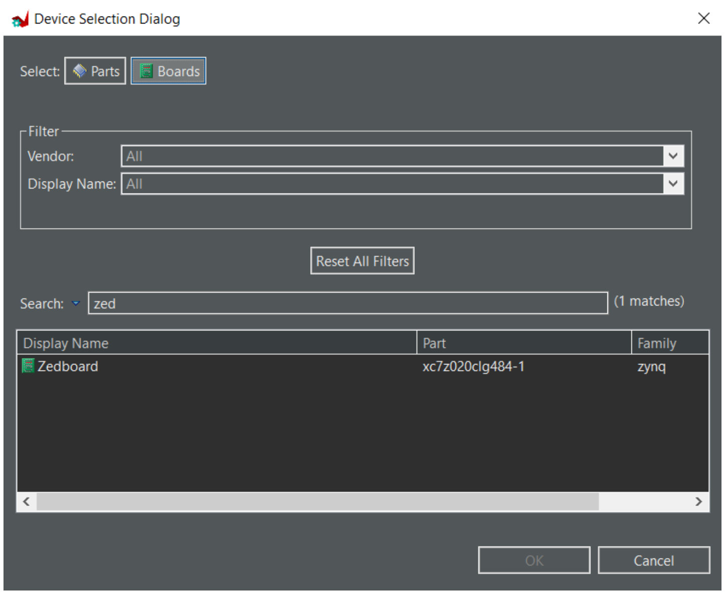
CFLAGS Value: -DHW_COSIM

6. In the *Solution Configuration* page, leave *Solution Name* field as solution1 and set the *clock period* as 10.



Solution Name: solution1
Clock
Period: 10 Uncertainty:
Part Selection
Part: Zedboard (xc7z020clg484-1)
Flow Target
Vivado IP Flow Target Configure several options for the selected flow target

7. Click the … button of the *Part Selection* section, choose *Boards* and then find the entry for the *Zedboard*.

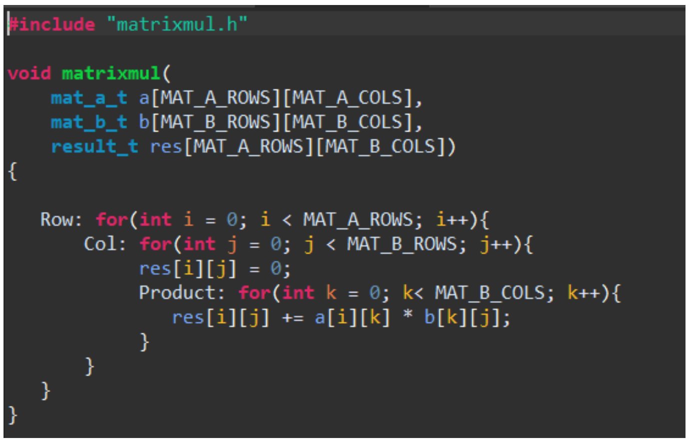


Filter
Vendor: All
Display Name: All

Search: zed (1 matches)

Display Name	Part	Family
Zedboard	xc7z020clg484-1	zynq

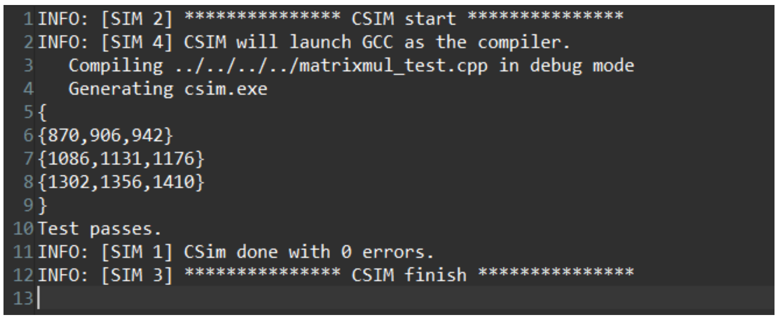
8. In the *Source* entry of the *Explorer* project directory, double-click to open the **matrixmul.cpp** file.



```
#include "matrixmul.h"

void matrixmul(
    mat_a_t a[MAT_A_ROWS][MAT_A_COLS],
    mat_b_t b[MAT_B_ROWS][MAT_B_COLS],
    result_t res[MAT_A_ROWS][MAT_B_COLS])
{

   Row: for(int i = 0; i < MAT_A_ROWS; i++){
       Col: for(int j = 0; j < MAT_B_ROWS; j++){
            res[i][j] = 0;
            Product: for(int k = 0; k< MAT_B_COLS; k++){
               res[i][j] += a[i][k] * b[k][j];
            }
       }
   }
}
```

You can see that the code implements a matrix multiplication with three nested loops. The product loop (inner most loop) is where the actual product and sum happens.

9. Select *Project* and *Run C Simulation*, click OK without making any changes.

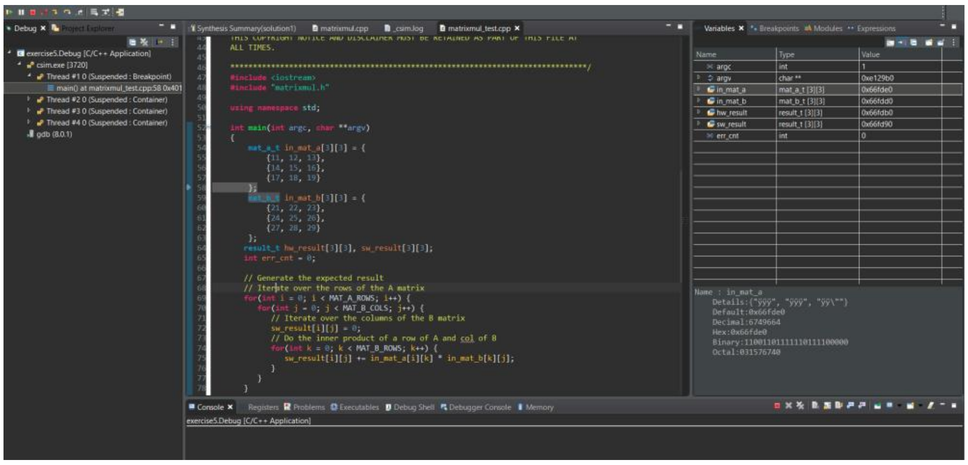


```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../matrixmul_test.cpp in debug mode
   Generating csim.exe
{
{870,906,942}
{1086,1131,1176}
{1302,1356,1410}
}
Test passes.
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************

```

10. Now go to the **matrixmul_test.cpp** file to see its content.

    You will find two matrices initialized at the beginning of the *main* function. If HW_COSIM is defined (as was done during the project set-up) then the matrixmul function is called and compares the output of the computed result with the one returned from the called function, and prints *Test passed* if the results match.

11. Now simulate the design again, but this time choose the Launch Debugger option.



```
exercise5.Debug [C/C++ Application]
  csim.exe [3720]
    Thread #1 0 (Suspended : Breakpoint)
      main() at matrixmul_test.cpp:58 0x401
    Thread #2 0 (Suspended : Container)
    Thread #3 0 (Suspended : Container)
    Thread #4 0 (Suspended : Container)
  gdb (8.0.1)
```

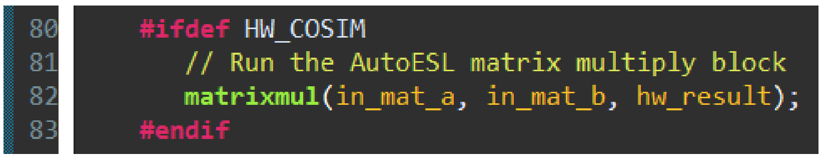
```cpp
THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS PART OF THIS FILE AT
ALL TIMES.

*******************************************************************************/
#include <iostream>
#include "matrixmul.h"

using namespace std;

int main(int argc, char **argv)
{
    mat_a_t in_mat_a[3][3] = {
        {11, 12, 13},
        {14, 15, 16},
        {17, 18, 19}
    };
    mat_b_t in_mat_b[3][3] = {
        {21, 22, 23},
        {24, 25, 26},
        {27, 28, 29}
    };
   result_t hw_result[3][3], sw_result[3][3];
   int err_cnt = 0;

   // Generate the expected result
   // Iterate over the rows of the A matrix
   for(int i = 0; i < MAT_A_ROWS; i++) {
      for(int j = 0; j < MAT_B_COLS; j++) {
         // Iterate over the columns of the B matrix
         sw_result[i][j] = 0;
         // Do the inner product of a row of A and col of B
         for(int k = 0; k < MAT_B_ROWS; k++) {
            sw_result[i][j] += in_mat_a[i][k] * in_mat_b[k][j];
         }
      }
   }
```

Name	Type	Value
argc	int	1
argv	char **	0xe129b0
in_mat_a	mat_a_t [3][3]	0x66fde0
in_mat_b	mat_b_t [3][3]	0x66fdd0
hw_result	result_t [3][3]	0x66fdb0
sw_result	result_t [3][3]	0x66fd90
err_cnt	int	0

```
Name : in_mat_a
    Details:{"ÿÿÿ", "ÿÿÿ", "ÿÿ\""}
    Default:0x66fde0
    Decimal:6749664
    Hex:0x66fde0
    Binary:11001101111110111100000
    Octal:031576740
```

```
exercise5.Debug [C/C++ Application]
```

12. Set breakpoint in line 82.



```cpp
80   #ifdef HW_COSIM
81      // Run the AutoESL matrix multiply block
82      matrixmul(in_mat_a, in_mat_b, hw_result);
83   #endif
```

13. Using the Step Over (F6) button several times, observe the execution progress, and observe the variable values updating, as well as computed software result.

14. Now click the Resume button or press F8 to complete the software computation and stop at line 101.

Task 5.1

Change the values of the matrices initialized in lines 73 and 78. Then calculate the value of the multiplication by yourself and initialize the result as well instead of calculating it. Do two tests, one with the correct result and one with the wrong result, to make sure that you get the error message. Include the values of your matrices in your report as well as a screenshot of both the success and error messages.

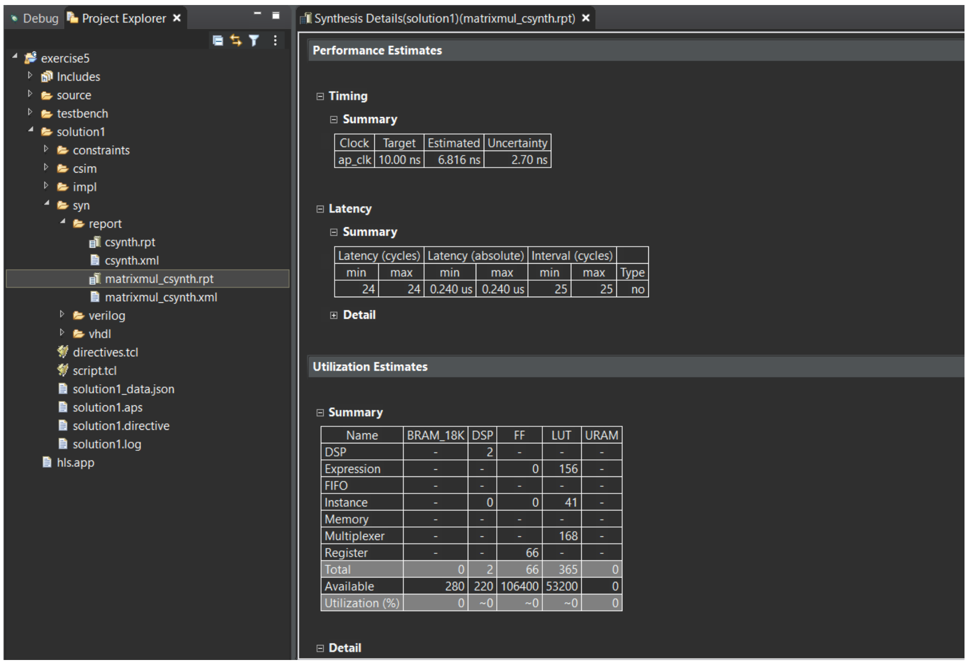
Turun yliopisto
University of Turku

Synthesis of the Design

Follow the instructions below to create and simulate a project in Vitis HLS.

1. Select Solution > Run C Synthesis > Active Solution

2. When the synthesis process is completed, the synthesis results will be displayed:



Debug | Project Explorer | Synthesis Details(solution1)(matrixmul_csynth.rpt)

```
exercise5
  Includes
  source
  testbench
  solution1
    constraints
    csim
    impl
    syn
      report
        csynth.rpt
        csynth.xml
        matrixmul_csynth.rpt
        matrixmul_csynth.xml
      verilog
      vhdl
    directives.tcl
    script.tcl
    solution1_data.json
    solution1.aps
    solution1.directive
    solution1.log
  hls.app
```

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.816 ns	2.70 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
24	24	0.240 us	0.240 us	25	25	no

Detail

Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	2	-	-	-
Expression	-	-	0	156	-
FIFO	-	-	-	-	-
Instance	-	0	0	41	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	168	-
Register	-	-	66	-	-
Total	0	2	66	365	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0

Detail

3. If you expand **solution1** in *Explorer*, several generated files will be available. These include reports but also the implementation in Verilog and VHDL.

Task 5.2

Answer the following questions:

- ✓ Estimated clock period
- ✓ Worst case latency
- ✓ Number of DSP48E utilized in the implementation
- ✓ Number of FFs utilized in the implementation
- ✓ Number of LUTs utilized in the implementation

Task 5.3

Write one or two paragraphs (4-10 lines) explaining your understanding of the benefits, drawbacks and challenges of working with FPGAs writing code in VHDL or in C++ and utilizing High Level Synthesis to automatically generate the VHDL code. How readable do you find the automatic VHDL code in Vitis HLS? Is there any part that you do not understand at all?