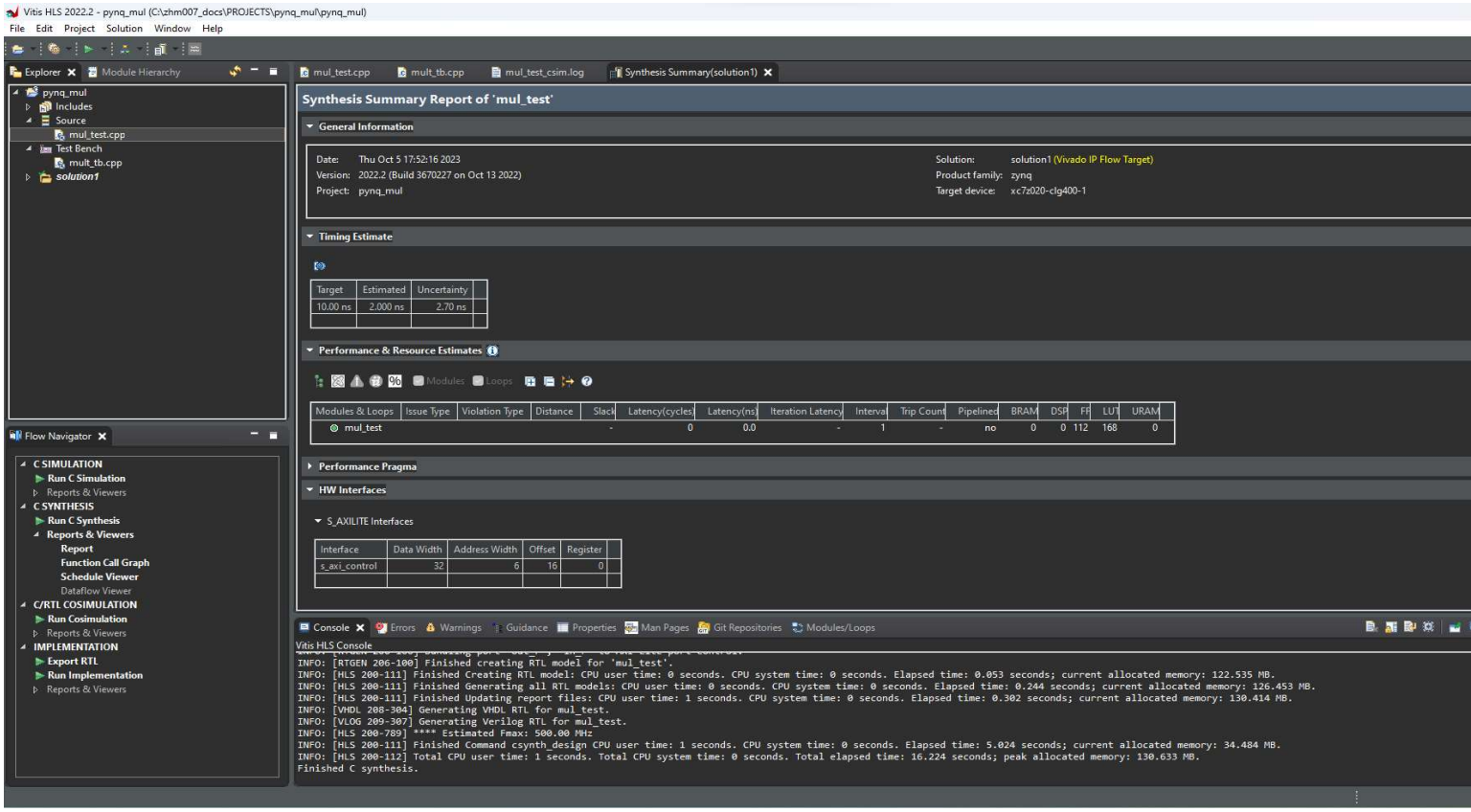
WES 237C Hardware for Embedded Systems Kastner Section 1: Lab1 Section 2: Project 1, FIR	**Ricardo Lizárraga** rilizarraga@ucsd.edu, **PID: A69028483**	***Submit to:*** *Ryan Charles Kastner* *(TA) Zhenghua Ma* *(TA) Abarajithan Gnaneswaran*

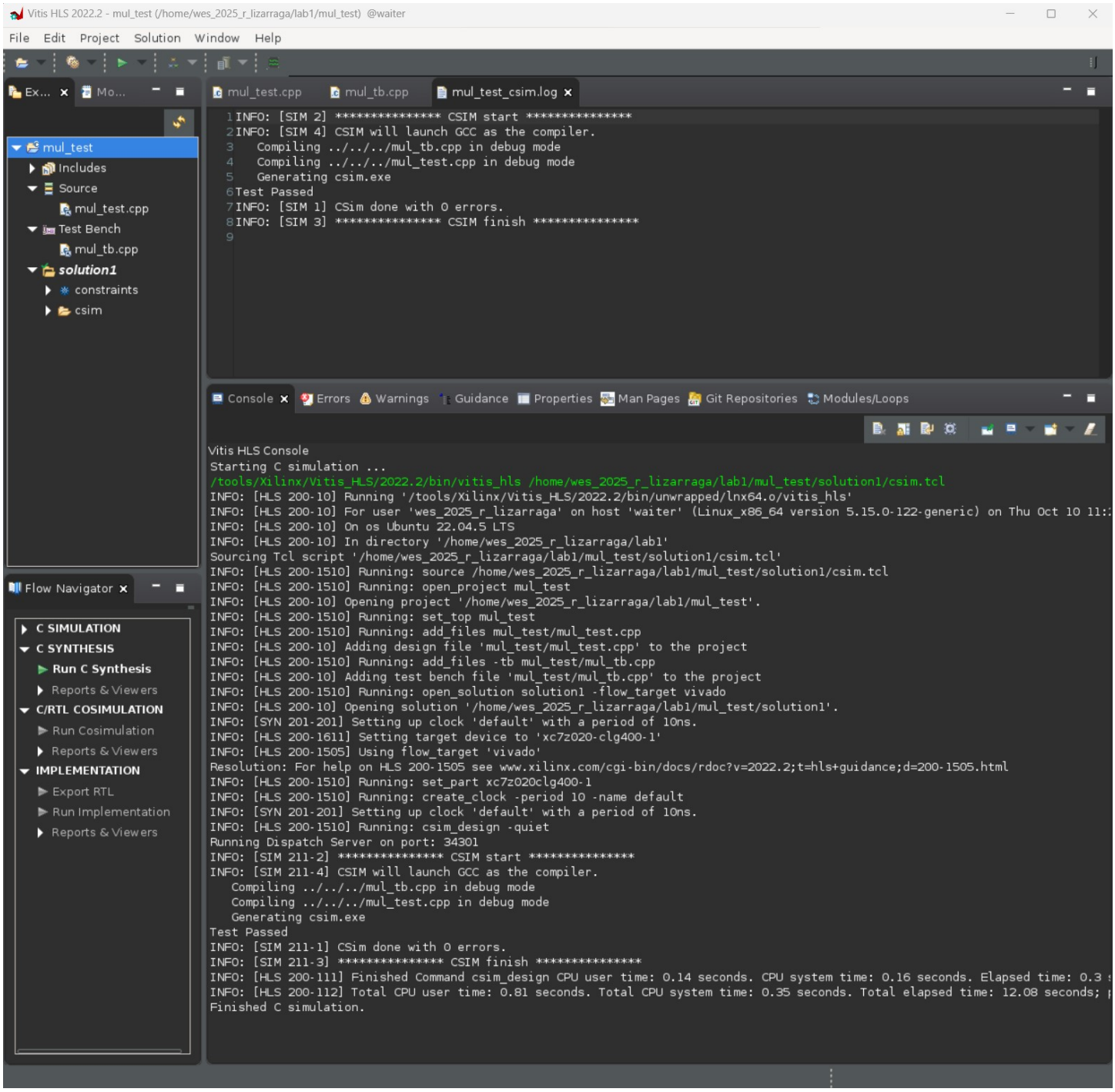
Section 1: Lab1

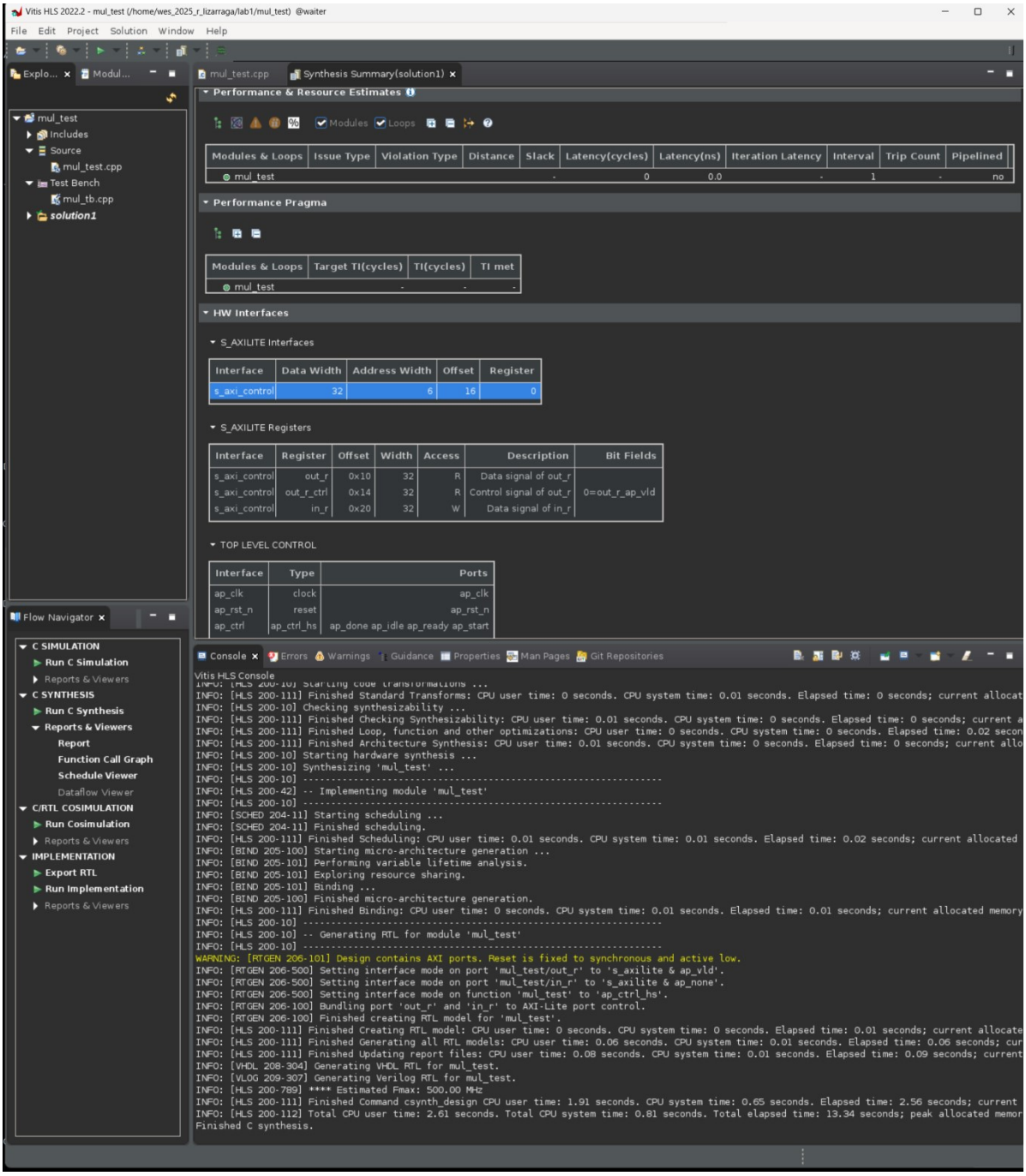
https://pp4fpgas.readthedocs.io/en/latest/PYNQ-example.html
https://pp4fpgas.readthedocs.io/en/latest/project1.html
should look like this:



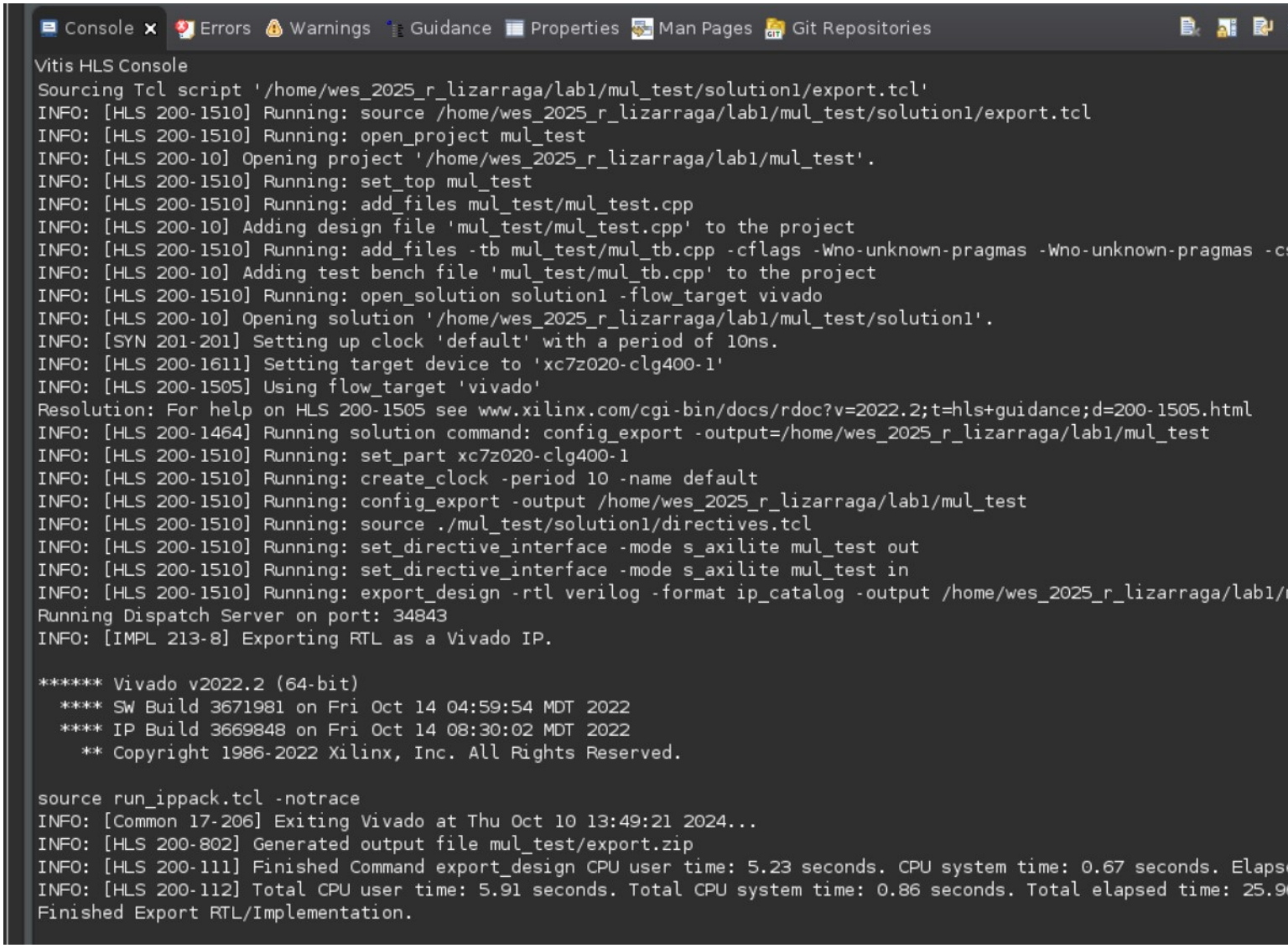
mul_test.cpp mul_tb.cpp mul_test_csim.log

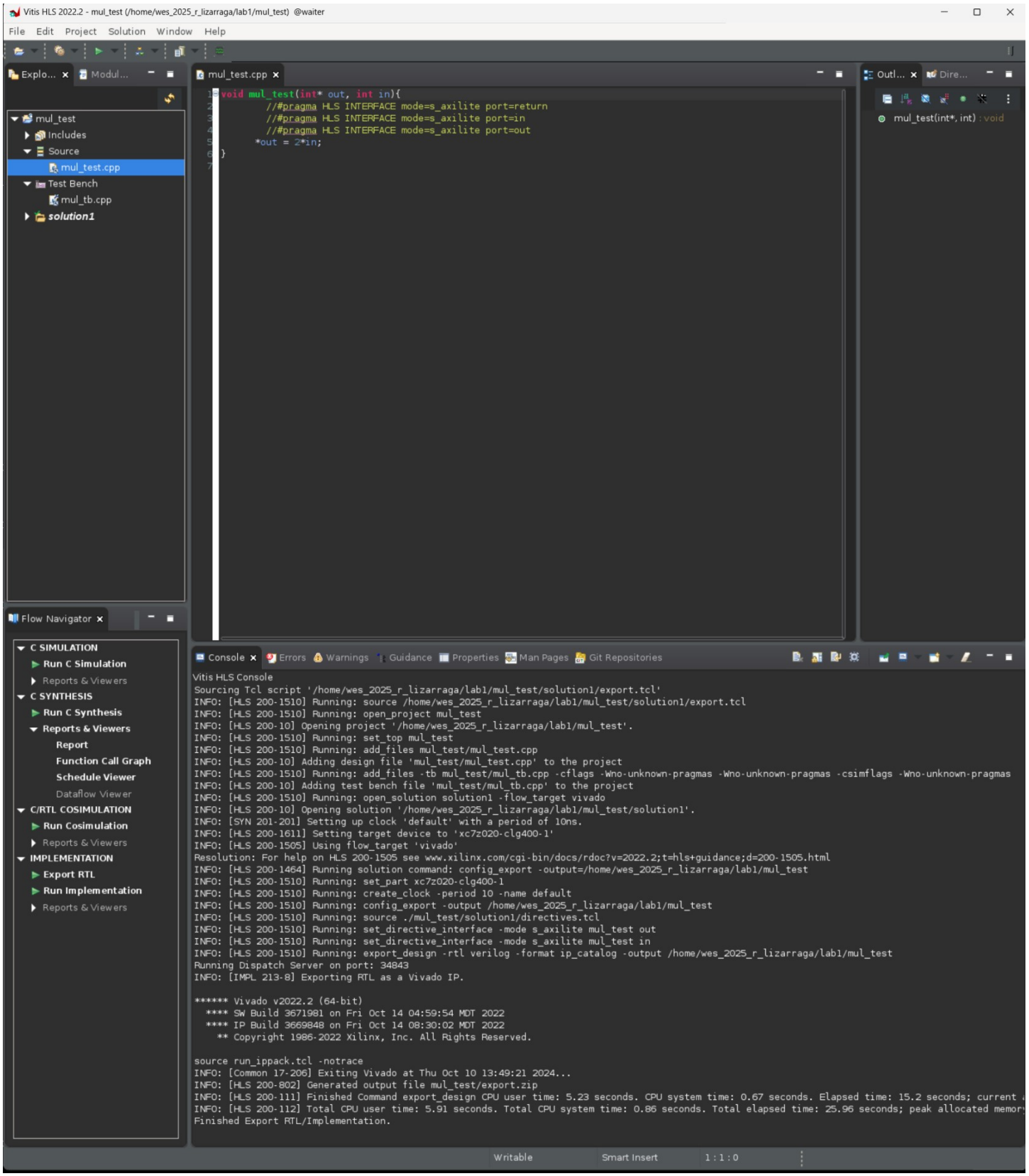
```
1 INFO: [SIM 2] ***************** CSIM start *****************
2 INFO: [SIM 4] CSIM will launch GCC as the compiler.
3    Compiling ../../../mul_tb.cpp in debug mode
4    Compiling ../../../mul_test.cpp in debug mode
5    Generating csim.exe
6 Test Passed
7 INFO: [SIM 1] CSim done with 0 errors.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

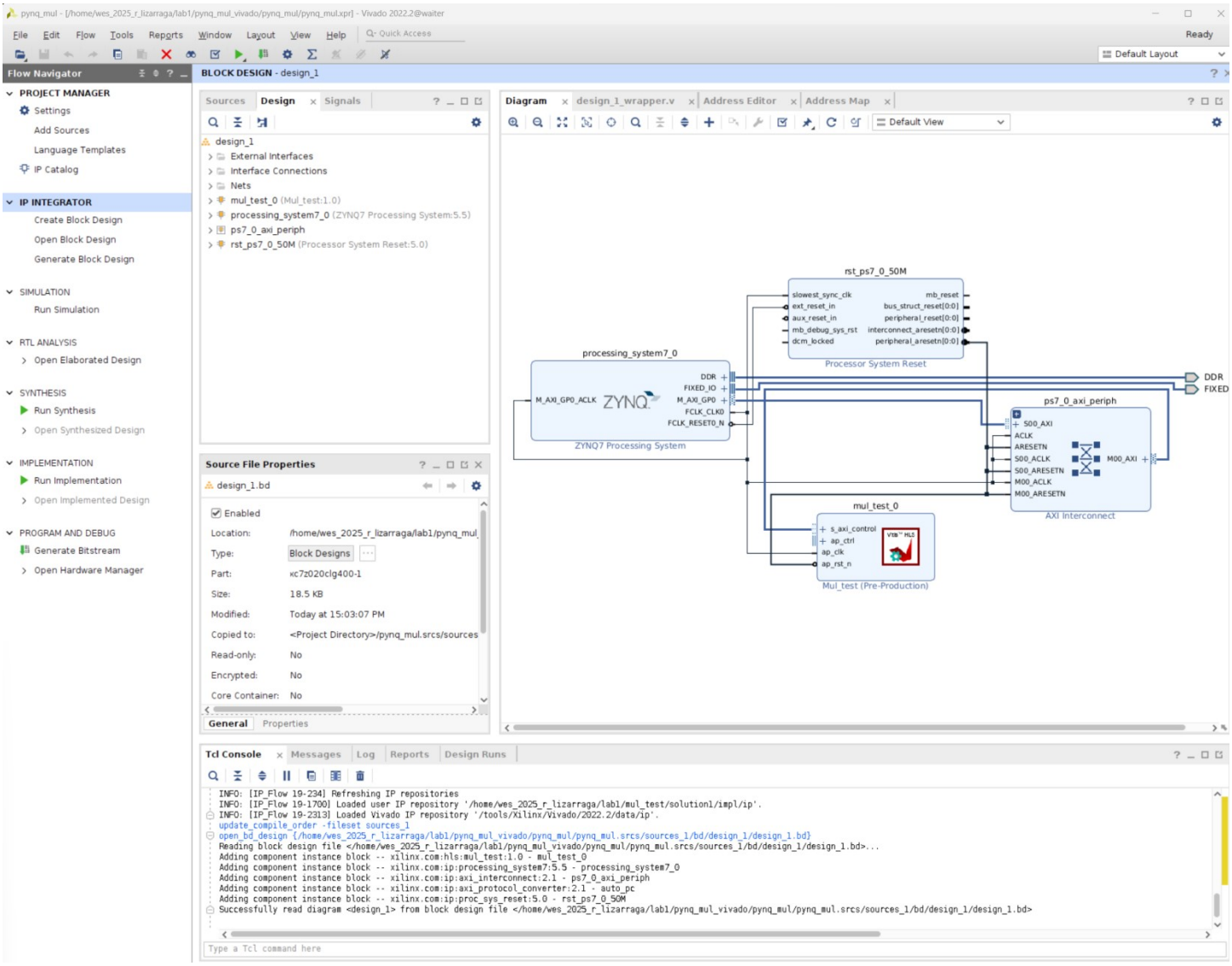

Vitis HLS 2022.2 - mul_test (/home/wes_2025_r_lizarraga/lab1/mul_test) @waiter
File Edit Project Solution Window Help
mul_test
Includes
Source
mul_test.cpp
Test Bench
mul_tb.cpp
solution1
constraints
csim
mul_test.cpp
mul_tb.cpp
mul_test_csim.log
1 INFO: [SIM 2] *************** CSIM start ***************
2 INFO: [SIM 4] CSIM will launch GCC as the compiler.
3 Compiling ../../../mul_tb.cpp in debug mode
4 Compiling ../../../mul_test.cpp in debug mode
5 Generating csim.exe
6 Test Passed
7 INFO: [SIM 1] CSim done with 0 errors.
8 INFO: [SIM 3] *************** CSIM finish ***************
9
Console
Errors
Warnings
Guidance
Properties
Man Pages
Git Repositories
Modules/Loops
Vitis HLS Console
Starting C simulation ...
/tools/Xilinx/Vitis_HLS/2022.2/bin/vitis_hls /home/wes_2025_r_lizarraga/lab1/mul_test/solution1/csim.tcl
INFO: [HLS 200-10] Running '/tools/Xilinx/Vitis_HLS/2022.2/bin/unwrapped/lnx64.o/vitis_hls'
INFO: [HLS 200-10] For user 'wes_2025_r_lizarraga' on host 'waiter' (Linux_x86_64 version 5.15.0-122-generic) on Thu Oct 10 11:
INFO: [HLS 200-10] On os Ubuntu 22.04.5 LTS
INFO: [HLS 200-10] In directory '/home/wes_2025_r_lizarraga/lab1'
Sourcing Tcl script '/home/wes_2025_r_lizarraga/lab1/mul_test/solution1/csim.tcl'
INFO: [HLS 200-1510] Running: source /home/wes_2025_r_lizarraga/lab1/mul_test/solution1/csim.tcl
INFO: [HLS 200-1510] Running: open_project mul_test
INFO: [HLS 200-10] Opening project '/home/wes_2025_r_lizarraga/lab1/mul_test'.
INFO: [HLS 200-1510] Running: set_top mul_test
INFO: [HLS 200-1510] Running: add_files mul_test/mul_test.cpp
INFO: [HLS 200-10] Adding design file 'mul_test/mul_test.cpp' to the project
INFO: [HLS 200-1510] Running: add_files -tb mul_test/mul_tb.cpp
INFO: [HLS 200-10] Adding test bench file 'mul_test/mul_tb.cpp' to the project
INFO: [HLS 200-1510] Running: open_solution solution1 -flow_target vivado
INFO: [HLS 200-10] Opening solution '/home/wes_2025_r_lizarraga/lab1/mul_test/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1611] Setting target device to 'xc7z020-clg400-1'
INFO: [HLS 200-1505] Using flow_target 'vivado'
Resolution: For help on HLS 200-1505 see www.xilinx.com/cgi-bin/docs/rdoc?v=2022.2;t=hls+guidance;d=200-1505.html
INFO: [HLS 200-1510] Running: set_part xc7z020clg400-1
INFO: [HLS 200-1510] Running: create_clock -period 10 -name default
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1510] Running: csim_design -quiet
Running Dispatch Server on port: 34301
INFO: [SIM 211-2] *************** CSIM start ***************
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
Compiling ../../../mul_tb.cpp in debug mode
Compiling ../../../mul_test.cpp in debug mode
Generating csim.exe
Test Passed
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: 0.14 seconds. CPU system time: 0.16 seconds. Elapsed time: 0.3
INFO: [HLS 200-112] Total CPU user time: 0.81 seconds. Total CPU system time: 0.35 seconds. Total elapsed time: 12.08 seconds;
Finished C simulation.
Flow Navigator
C SIMULATION
C SYNTHESIS
Run C Synthesis
Reports & Viewers
C/RTL COSIMULATION
Run Cosimulation
Reports & Viewers
IMPLEMENTATION
Export RTL
Run Implementation
Reports & Viewers



Finished Export RTL/Implementation:

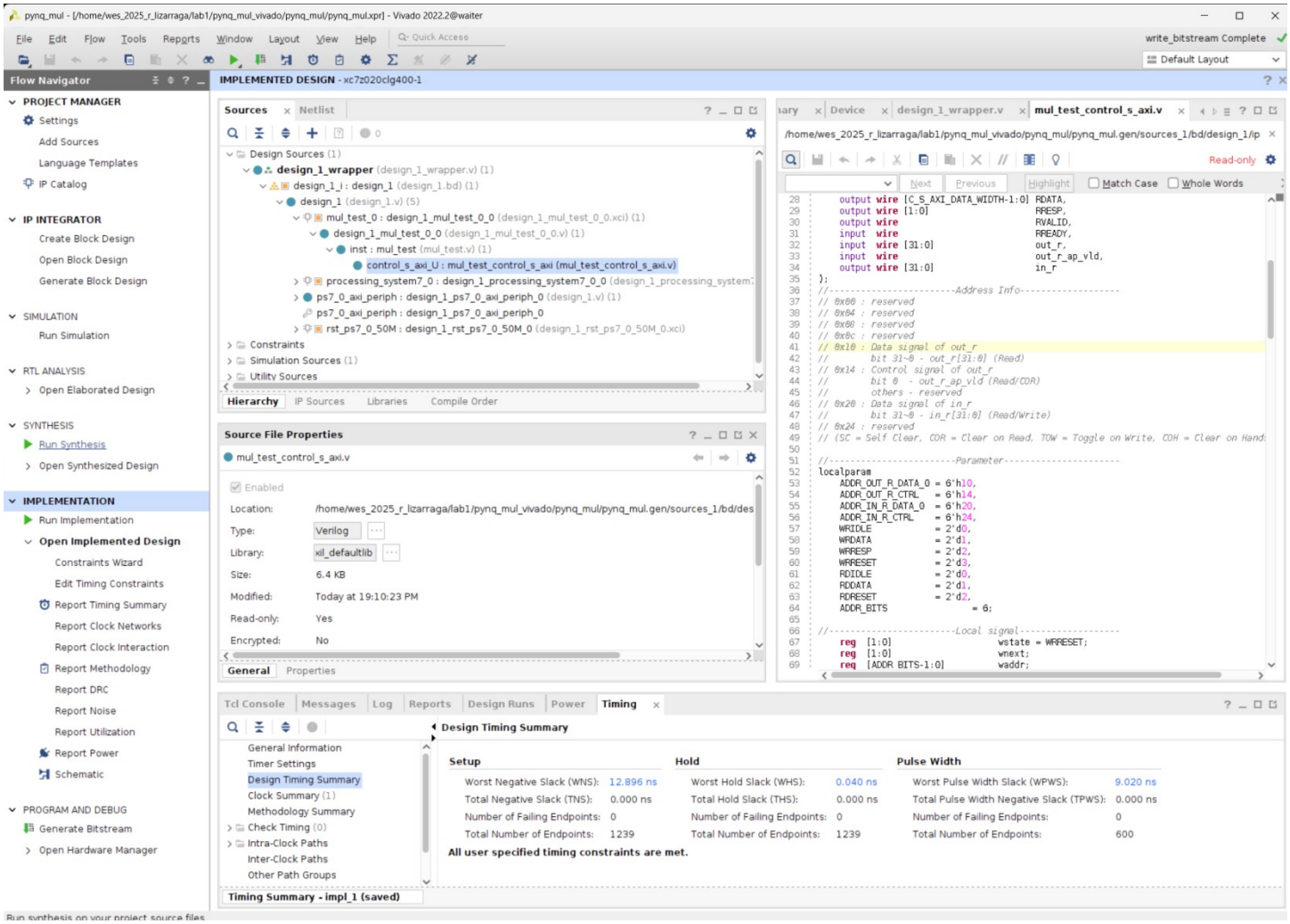

Console
Errors
Warnings
Guidance
Properties
Man Pages
Git Repositories
Vitis HLS Console
Sourcing Tcl script '/home/wes_2025_r_lizarraga/lab1/mul_test/solution1/export.tcl'
INFO: [HLS 200-1510] Running: source /home/wes_2025_r_lizarraga/lab1/mul_test/solution1/export.tcl
INFO: [HLS 200-1510] Running: open_project mul_test
INFO: [HLS 200-10] Opening project '/home/wes_2025_r_lizarraga/lab1/mul_test'.
INFO: [HLS 200-1510] Running: set_top mul_test
INFO: [HLS 200-1510] Running: add_files mul_test/mul_test.cpp
INFO: [HLS 200-10] Adding design file 'mul_test/mul_test.cpp' to the project
INFO: [HLS 200-1510] Running: add_files -tb mul_test/mul_tb.cpp -cflags -Wno-unknown-pragmas -Wno-unknown-pragmas -c
INFO: [HLS 200-10] Adding test bench file 'mul_test/mul_tb.cpp' to the project
INFO: [HLS 200-1510] Running: open_solution solution1 -flow_target vivado
INFO: [HLS 200-10] Opening solution '/home/wes_2025_r_lizarraga/lab1/mul_test/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1611] Setting target device to 'xc7z020-clg400-1'
INFO: [HLS 200-1505] Using flow_target 'vivado'
Resolution: For help on HLS 200-1505 see www.xilinx.com/cgi-bin/docs/rdoc?v=2022.2;t=hls+guidance;d=200-1505.html
INFO: [HLS 200-1464] Running solution command: config_export -output=/home/wes_2025_r_lizarraga/lab1/mul_test
INFO: [HLS 200-1510] Running: set_part xc7z020-clg400-1
INFO: [HLS 200-1510] Running: create_clock -period 10 -name default
INFO: [HLS 200-1510] Running: config_export -output /home/wes_2025_r_lizarraga/lab1/mul_test
INFO: [HLS 200-1510] Running: source ./mul_test/solution1/directives.tcl
INFO: [HLS 200-1510] Running: set_directive_interface -mode s_axilite mul_test out
INFO: [HLS 200-1510] Running: set_directive_interface -mode s_axilite mul_test in
INFO: [HLS 200-1510] Running: export_design -rtl verilog -format ip_catalog -output /home/wes_2025_r_lizarraga/lab1/
Running Dispatch Server on port: 34843
INFO: [IMPL 213-8] Exporting RTL as a Vivado IP.
****** Vivado v2022.2 (64-bit)
**** SW Build 3671981 on Fri Oct 14 04:59:54 MDT 2022
**** IP Build 3669848 on Fri Oct 14 08:30:02 MDT 2022
** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
source run_ippack.tcl -notrace
INFO: [Common 17-206] Exiting Vivado at Thu Oct 10 13:49:21 2024...
INFO: [HLS 200-802] Generated output file mul_test/export.zip
INFO: [HLS 200-111] Finished Command export_design CPU user time: 5.23 seconds. CPU system time: 0.67 seconds. Elaps
INFO: [HLS 200-112] Total CPU user time: 5.91 seconds. Total CPU system time: 0.86 seconds. Total elapsed time: 25.9
Finished Export RTL/Implementation.


Vitis HLS 2022.2 - mul_test (/home/wes_2025_r_lizarraga/lab1/mul_test) @waiter
File Edit Project Solution Window Help
mul_test
Includes
Source
mul_test.cpp
Test Bench
mul_tb.cpp
solution1
mul_test.cpp
void mul_test(int* out, int in){
//#pragma HLS INTERFACE mode=s_axilite port=return
//#pragma HLS INTERFACE mode=s_axilite port=in
//#pragma HLS INTERFACE mode=s_axilite port=out
*out = 2*in;
}
mul_test(int*, int) : void
Flow Navigator
C SIMULATION
Run C Simulation
Reports & Viewers
C SYNTHESIS
Run C Synthesis
Reports & Viewers
Report
Function Call Graph
Schedule Viewer
Dataflow Viewer
C/RTL COSIMULATION
Run Cosimulation
Reports & Viewers
IMPLEMENTATION
Export RTL
Run Implementation
Reports & Viewers
Console
Errors
Warnings
Guidance
Properties
Man Pages
Git Repositories
Vitis HLS Console
Sourcing Tcl script '/home/wes_2025_r_lizarraga/lab1/mul_test/solution1/export.tcl'
INFO: [HLS 200-1510] Running: source /home/wes_2025_r_lizarraga/lab1/mul_test/solution1/export.tcl
INFO: [HLS 200-1510] Running: open_project mul_test
INFO: [HLS 200-10] Opening project '/home/wes_2025_r_lizarraga/lab1/mul_test'.
INFO: [HLS 200-1510] Running: set_top mul_test
INFO: [HLS 200-1510] Running: add_files mul_test/mul_test.cpp
INFO: [HLS 200-10] Adding design file 'mul_test/mul_test.cpp' to the project
INFO: [HLS 200-1510] Running: add_files -tb mul_test/mul_tb.cpp -cflags -Wno-unknown-pragmas -Wno-unknown-pragmas -csimflags -Wno-unknown-pragmas
INFO: [HLS 200-10] Adding test bench file 'mul_test/mul_tb.cpp' to the project
INFO: [HLS 200-1510] Running: open_solution solution1 -flow_target vivado
INFO: [HLS 200-10] Opening solution '/home/wes_2025_r_lizarraga/lab1/mul_test/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1611] Setting target device to 'xc7z020-clg400-1'
INFO: [HLS 200-1505] Using flow_target 'vivado'
Resolution: For help on HLS 200-1505 see www.xilinx.com/cgi-bin/docs/rdoc?v=2022.2;t=hls+guidance;d=200-1505.html
INFO: [HLS 200-1464] Running solution command: config_export -output=/home/wes_2025_r_lizarraga/lab1/mul_test
INFO: [HLS 200-1510] Running: set_part xc7z020-clg400-1
INFO: [HLS 200-1510] Running: create_clock -period 10 -name default
INFO: [HLS 200-1510] Running: config_export -output /home/wes_2025_r_lizarraga/lab1/mul_test
INFO: [HLS 200-1510] Running: source ./mul_test/solution1/directives.tcl
INFO: [HLS 200-1510] Running: set_directive_interface -mode s_axilite mul_test out
INFO: [HLS 200-1510] Running: set_directive_interface -mode s_axilite mul_test in
INFO: [HLS 200-1510] Running: export_design -rtl verilog -format ip_catalog -output /home/wes_2025_r_lizarraga/lab1/mul_test
Running Dispatch Server on port: 34843
INFO: [IMPL 213-8] Exporting RTL as a Vivado IP.
****** Vivado v2022.2 (64-bit)
**** SW Build 3671981 on Fri Oct 14 04:59:54 MDT 2022
**** IP Build 3669848 on Fri Oct 14 08:30:02 MDT 2022
** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
source run_ippack.tcl -notrace
INFO: [Common 17-206] Exiting Vivado at Thu Oct 10 13:49:21 2024...
INFO: [HLS 200-802] Generated output file mul_test/export.zip
INFO: [HLS 200-111] Finished Command export_design CPU user time: 5.23 seconds. CPU system time: 0.67 seconds. Elapsed time: 15.2 seconds; current
INFO: [HLS 200-112] Total CPU user time: 5.91 seconds. Total CPU system time: 0.86 seconds. Total elapsed time: 25.96 seconds; peak allocated memory
Finished Export RTL/Implementation.
Writable
Smart Insert
1 : 1 : 0

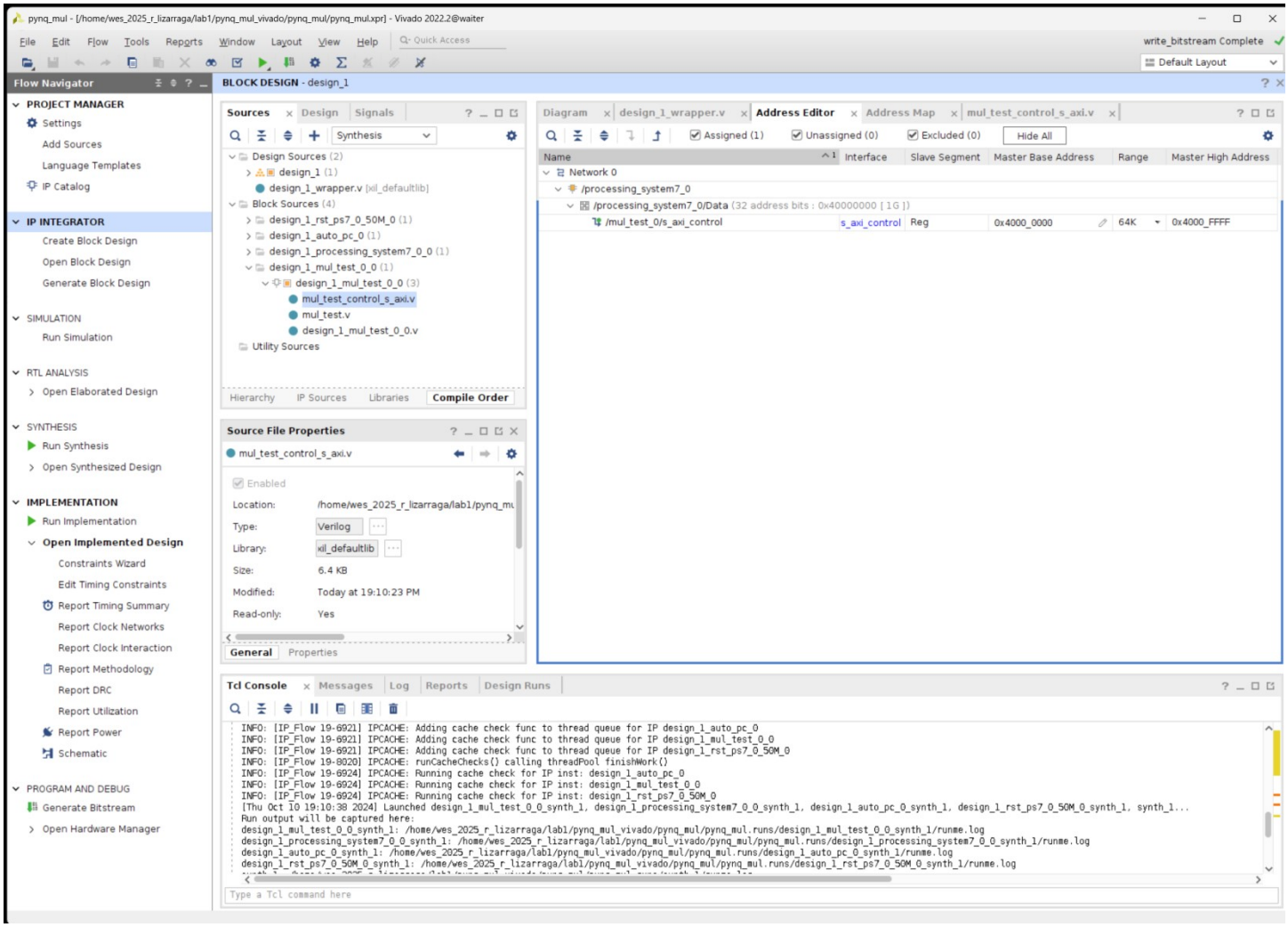


note addresses for in and out ports:

```
//------------------------Address Info-------------------
// 0x00 : reserved
// 0x04 : reserved
// 0x08 : reserved
// 0x0c : reserved
// 0x10 : Data signal of out_r
//        bit 31~0 - out_r[31:0] (Read)
// 0x14 : Control signal of out_r
//        bit 0  - out_r_ap_vld (Read/COR)
//        others - reserved
// 0x20 : Data signal of in_r
//        bit 31~0 - in_r[31:0] (Read/Write)
// 0x24 : reserved
// (SC = Self Clear, COR = Clear on Read, TOW = Toggle on Wri
```


pynq_mul - [/home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.xpr] - Vivado 2022.2@waiter
File Edit Flow Tools Reports Window Layout View Help
write_bitstream Complete
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
IMPLEMENTED DESIGN - xc7z020clg400-1
Sources
Netlist
Design Sources (1)
design_1_wrapper (design_1_wrapper.v) (1)
design_1_i : design_1 (design_1.bd) (1)
design_1 (design_1.v) (5)
mul_test_0 : design_1_mul_test_0_0 (design_1_mul_test_0_0.xci) (1)
design_1_mul_test_0_0 (design_1_mul_test_0_0.v) (1)
inst : mul_test (mul_test.v) (1)
control_s_axi_U : mul_test_control_s_axi (mul_test_control_s_axi.v)
processing_system7_0 : design_1_processing_system7_0_0 (design_1_processing_system7
ps7_0_axi_periph : design_1_ps7_0_axi_periph_0 (design_1.v) (1)
ps7_0_axi_periph : design_1_ps7_0_axi_periph_0
rst_ps7_0_50M : design_1_rst_ps7_0_50M_0 (design_1_rst_ps7_0_50M_0.xci)
Constraints
Simulation Sources (1)
Utility Sources
Hierarchy
IP Sources
Libraries
Compile Order
Source File Properties
mul_test_control_s_axi.v
Enabled
Location: /home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.gen/sources_1/bd/des
Type: Verilog
Library: xil_defaultlib
Size: 6.4 KB
Modified: Today at 19:10:23 PM
Read-only: Yes
Encrypted: No
General
Properties
Device
design_1_wrapper.v
mul_test_control_s_axi.v
/home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.gen/sources_1/bd/design_1/ip
Read-only
Next
Previous
Highlight
Match Case
Whole Words
output wire [C_S_AXI_DATA_WIDTH-1:0] RDATA,
output wire [1:0] RRESP,
output wire RVALID,
input wire RREADY,
input wire [31:0] out_r,
input wire out_r_ap_vld,
output wire [31:0] in_r
);
//------------------------Address Info-------------------
// 0x00 : reserved
// 0x04 : reserved
// 0x08 : reserved
// 0x0c : reserved
// 0x10 : Data signal of out_r
// bit 31~0 - out_r[31:0] (Read)
// 0x14 : Control signal of out_r
// bit 0 - out_r_ap_vld (Read/COR)
// others - reserved
// 0x20 : Data signal of in_r
// bit 31~0 - in_r[31:0] (Read/Write)
// 0x24 : reserved
// (SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Hand
//------------------------Parameter----------------------
localparam
ADDR_OUT_R_DATA_0 = 6'h10,
ADDR_OUT_R_CTRL = 6'h14,
ADDR_IN_R_DATA_0 = 6'h20,
ADDR_IN_R_CTRL = 6'h24,
WRIDLE = 2'd0,
WRDATA = 2'd1,
WRRESP = 2'd2,
WRRESET = 2'd3,
RDIDLE = 2'd0,
RDDATA = 2'd1,
RDRESET = 2'd2,
ADDR_BITS = 6;
//------------------------Local signal-------------------
reg [1:0] wstate = WRRESET;
reg [1:0] wnext;
reg [ADDR_BITS-1:0] waddr;
Tcl Console
Messages
Log
Reports
Design Runs
Power
Timing
Design Timing Summary
General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Methodology Summary
Check Timing (0)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
Setup
Worst Negative Slack (WNS): 12.896 ns
Total Negative Slack (TNS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 1239
Hold
Worst Hold Slack (WHS): 0.040 ns
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 1239
Pulse Width
Worst Pulse Width Slack (WPWS): 9.020 ns
Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 600
All user specified timing constraints are met.
Timing Summary - impl_1 (saved)
Run synthesis on your project source files


Name
Interface
Slave Segment
Master Base Address
Range
Master High Address
Network 0
/processing_system7_0
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])
/mul_test_0/s_axi_control
s_axi_control
Reg
0x4000_0000
64K
0x4000_FFFF


pynq_mul - [/home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.xpr] - Vivado 2022.2@waiter
File Edit Flow Tools Reports Window Layout View Help
write_bitstream Complete
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Utilization
Report Power
Schematic
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
BLOCK DESIGN - design_1
Sources
Design
Signals
Synthesis
Design Sources (2)
design_1 (1)
design_1_wrapper.v [xil_defaultlib]
Block Sources (4)
design_1_rst_ps7_0_50M_0 (1)
design_1_auto_pc_0 (1)
design_1_processing_system7_0_0 (1)
design_1_mul_test_0_0 (1)
design_1_mul_test_0_0 (3)
mul_test_control_s_axi.v
mul_test.v
design_1_mul_test_0_0.v
Utility Sources
Hierarchy
IP Sources
Libraries
Compile Order
Source File Properties
mul_test_control_s_axi.v
Enabled
Location: /home/wes_2025_r_lizarraga/lab1/pynq_mu
Type: Verilog
Library: xil_defaultlib
Size: 6.4 KB
Modified: Today at 19:10:23 PM
Read-only: Yes
General
Properties
Diagram
design_1_wrapper.v
Address Editor
Address Map
mul_test_control_s_axi.v
Assigned (1)
Unassigned (0)
Excluded (0)
Hide All
Name
Interface
Slave Segment
Master Base Address
Range
Master High Address
Network 0
/processing_system7_0
/processing_system7_0/Data (32 address bits : 0x40000000 [1G])
/mul_test_0/s_axi_control
s_axi_control
Reg
0x4000_0000
64K
0x4000_FFFF
Tcl Console
Messages
Log
Reports
Design Runs
INFO: [IP_Flow 19-6921] IPCACHE: Adding cache check func to thread queue for IP design_1_auto_pc_0
INFO: [IP_Flow 19-6921] IPCACHE: Adding cache check func to thread queue for IP design_1_mul_test_0_0
INFO: [IP_Flow 19-6921] IPCACHE: Adding cache check func to thread queue for IP design_1_rst_ps7_0_50M_0
INFO: [IP_Flow 19-8020] IPCACHE: runCacheChecks() calling threadPool finishWork()
INFO: [IP_Flow 19-6924] IPCACHE: Running cache check for IP inst: design_1_auto_pc_0
INFO: [IP_Flow 19-6924] IPCACHE: Running cache check for IP inst: design_1_mul_test_0_0
INFO: [IP_Flow 19-6924] IPCACHE: Running cache check for IP inst: design_1_rst_ps7_0_50M_0
[Thu Oct 10 19:10:38 2024] Launched design_1_mul_test_0_0_synth_1, design_1_processing_system7_0_0_synth_1, design_1_auto_pc_0_synth_1, design_1_rst_ps7_0_50M_0_synth_1, synth_1...
Run output will be captured here:
design_1_mul_test_0_0_synth_1: /home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.runs/design_1_mul_test_0_0_synth_1/runme.log
design_1_processing_system7_0_0_synth_1: /home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.runs/design_1_processing_system7_0_0_synth_1/runme.log
design_1_auto_pc_0_synth_1: /home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.runs/design_1_auto_pc_0_synth_1/runme.log
design_1_rst_ps7_0_50M_0_synth_1: /home/wes_2025_r_lizarraga/lab1/pynq_mul_vivado/pynq_mul/pynq_mul.runs/design_1_rst_ps7_0_50M_0_synth_1/runme.log
Type a Tcl command here

jupyter Lab1 Last Checkpoint: 11/02/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code Voilà

In [1]:

```
from pynq import Overlay
from pynq import MMIO

ol = Overlay("./design_1_wrapper.bit") # designate a bitstream to be flashed to the FPGA
ol.download() # flash the FPGA

mul_ip = MMIO(0x40000000, 0x10000) # (IP_BASE_ADDRESS, ADDRESS_RANGE), told to us in Vivado
inp = 6 # number we want to double

mul_ip.write(0x20, inp) # write input value to input address in fabric
print("input:", mul_ip.read(0x20)) # confirm that our value was written correctly to the fabri
mul_ip.write(0x00, 1) # set ap_start to 1 which initiates the process we wrote to the fabric
print("output:", mul_ip.read(0x10)) # read corresponding output value from the output address
```

```
input: 6
output: 12
```

In [1]:

```
from pynq import Overlay
from pynq import MMIO

ol = Overlay("./design_1_wrapper.bit") # designa
ol.download() # flash the FPGA

mul_ip = MMIO(0x40000000, 0x10000) # (IP_BASE_AD
inp = 123456789 # number we want to double

mul_ip.write(0x20, inp) # write input value to i
print("input:", mul_ip.read(0x20)) # confirm tha
mul_ip.write(0x00, 1) # set ap_start to 1 which
print("output:", mul_ip.read(0x10)) # read corre
```

```
input: 123456789
output: 246913578
```

Section 2: Project: FIR Filter Design

1) Introduction

The goal of this project is to learn how the basics of an HLS tool. The learning outcomes are to gain a basic understanding of how the Vivado HLS tool works, to get exposed to HLS optimizations, to perform a guided design space exploration to obtain architectures with different tradeoffs in performance and resource usage, to generate a high-quality FIR architecture, and to demonstrate the integration of that FIR on the Zynq FPGA using the Pynq infrastructure.
This project is designed to be paired with Chapter 2 from Parallel Programming for FPGAs book. The book covers many aspects of the optimizations in this project, and we strongly recommend this as a reference.
The project is divided into three parts:

- Design an 11-tap FIR filter
- Design and optimize a 128-tap FIR filter
- Prototype an FIR filter architecture on a Zynq FPGA

You should start this assignment by understanding the 11-tap FIR filter and implementing a functionally correct design. Next, you modify the code and experiment with different optimizations specified in the questions. The 128-tap FIR filter is more complex and may have different trade-offs, and in the final report, you need to answer the questions about the 128-tap filter. Your answers should show that you understand different optimization and their effects on throughput, latency, and area. Finally, you will take one of your FIR filter designs, program that onto a Zynq FPGA, and demonstrate its functionality with the Pynq infrastructure.

2) Preparation

Before you start, we strongly suggest you familiarize yourself with the high-level synthesis tool.

- Vitis HLS: A good option is this VitisHLS User Guide. You do not need to go through the optimization steps, though they provide a good preview of the optimizations you will find in future projects.
- Vivado: Xilinx tool for RTL, SoC design (excluding firmware), and FPGA prototyping. It is not required for this project if you are not planning to prototype on Zynq FPGA. Vivado User Guide: Getting Started

You can follow the lab tutorials step by step (up to C synthesis and exporting RTL) to complete the Vitis HLS enviroment setup.

3) Materials

You can download the project files here:

- project1.zip

This contains:

- fir11 folder: 11 tap fir filter
    - fir.cpp - Implements top-level function
    - fir.h - header file
    - fir_test.cpp - test bench
    - input.dat - input chirp signal

    - out.gold.dat - “Golden” output. When the testbench (from fir_test.cpp) is run through the file fir.cpp it should generate this result. If it does not, you did something wrong.
- fir128 folder: 128 tap fir filter
    - fir.cpp - Implements top-level function
    - fir.h - header file
    - fir_test.cpp - test bench
    - input.dat - input chirp signal
    - out.gold.dat - “Golden” output. When the testbench (from fir_test.cpp) is run through the file fir.cpp it should generate this result. If it does not, you did something wrong.
- Tutorial folder:
    - ug871-vivado-high-level-synthesis-tutorial.pdf - Various tutorials for Vivado HLS
- demo folder: Demo folder for 11-tap filter
    - input.dat - input chirp signal

Target Board: xc7z020clg400-1
Software: Vitis HLS 2022.2 (recommended)
Clock Period: 10 ns or 100MHz

4) Project Goal

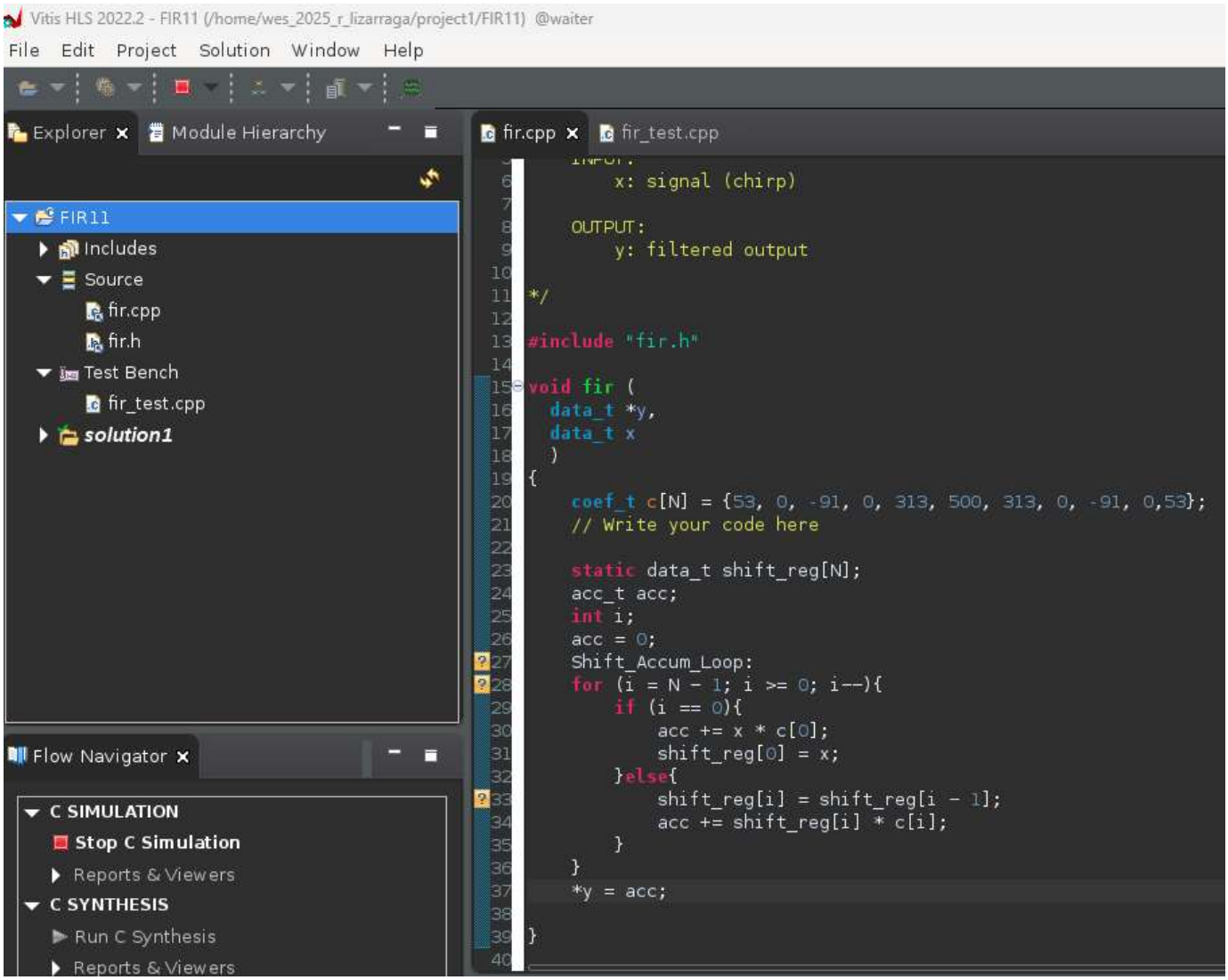
The first goal of this project is to generate a functionally correct HLS design for an 11-tap FIR filter. Also, you should start to gain an understanding of different HLS optimizations. For FIR128, you should modify the code to generate several optimized designs. Your goal is to create designs that provide tradeoffs between resource usage and execution time. This will require you to rewrite the code and insert pragmas. More specifically, you must do the following:

- Design an 11-tap FIR filter with HLS. In the rest of this document, we use the term FIR11 to refer to this task.
- Design a 128-tap FIR filter with HLS and optimize it. We call this subtask FIR128.

5) FIR11

$ cd /home/wes_2025_r_lizarraga/project1/
$ vitis_hls -f script.tcl

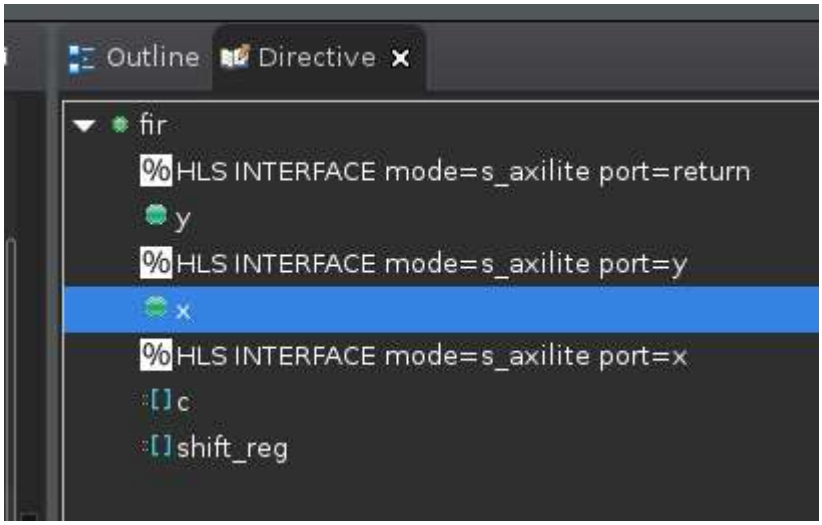
The first step for the project is to get a functionally correct design working for an 11-tap FIR filter. For this, you must use the Vivado HLS tool and finish the function body of *void fir()* in the file fir.cpp to implement the filter. You can test the correctness of your code by using the provided testbench. This code does not need to be highly optimized; you will work on creating optimized code later. It just needs to work correctly. **Use the provided script.tcl to create your project**, or manually add source & testbench files and set the top function.



Run C simulation

```
Starting C simulation ...
Starting C simulation ...
/tools/Xilinx/Vitis_HLS/2022.2/bin/vitis_hls /home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl
/tools/Xilinx/Vitis_HLS/2022.2/bin/vitis_hls /home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl
INFO: [HLS 200-10] Running '/tools/Xilinx/Vitis_HLS/2022.2/bin/unwrapped/lnx64.o/vitis_hls'
INFO: [HLS 200-10] For user 'wes_2025_r_lizarraga' on host 'waiter' (Linux_x86_64 version 5.15.0-122-generic) on Wed Oct 16 23:57:36 PDT
2024
INFO: [HLS 200-10] On os Ubuntu 22.04.5 LTS
INFO: [HLS 200-10] In directory '/home/wes_2025_r_lizarraga/project1'
Sourcing Tcl script '/home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl'
INFO: [HLS 200-1510] Running: source /home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl
/home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl can't be opened.
couldn't read file "/home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl": no such file or directory
while executing
"source /home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl"
invoked from within
"hls::main /home/wes_2025_r_lizarraga/project1/FIR11/solution1/csim.tcl"
("uplevel" body line 1)
invoked from within
"uplevel 1 hls::main {*}$newargs"
(procedure "hls_proc" line 16)
invoked from within
"hls_proc [info nameofexecutable] $argv"
INFO: [HLS 200-112] Total CPU user time: 0.33 seconds. Total CPU system time: 0.1 seconds. Total elapsed time: 0.32 seconds; peak allocated
memory: 98.898 MB.
Finished C simulation.
```

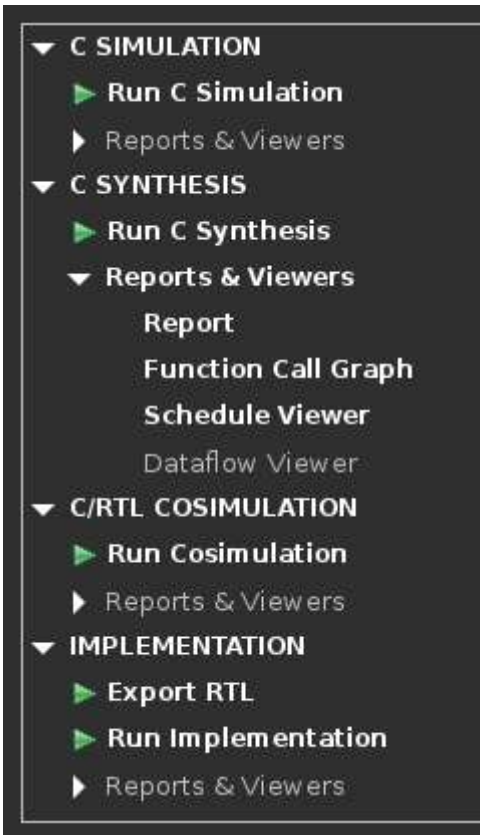
HLS INTERFACES



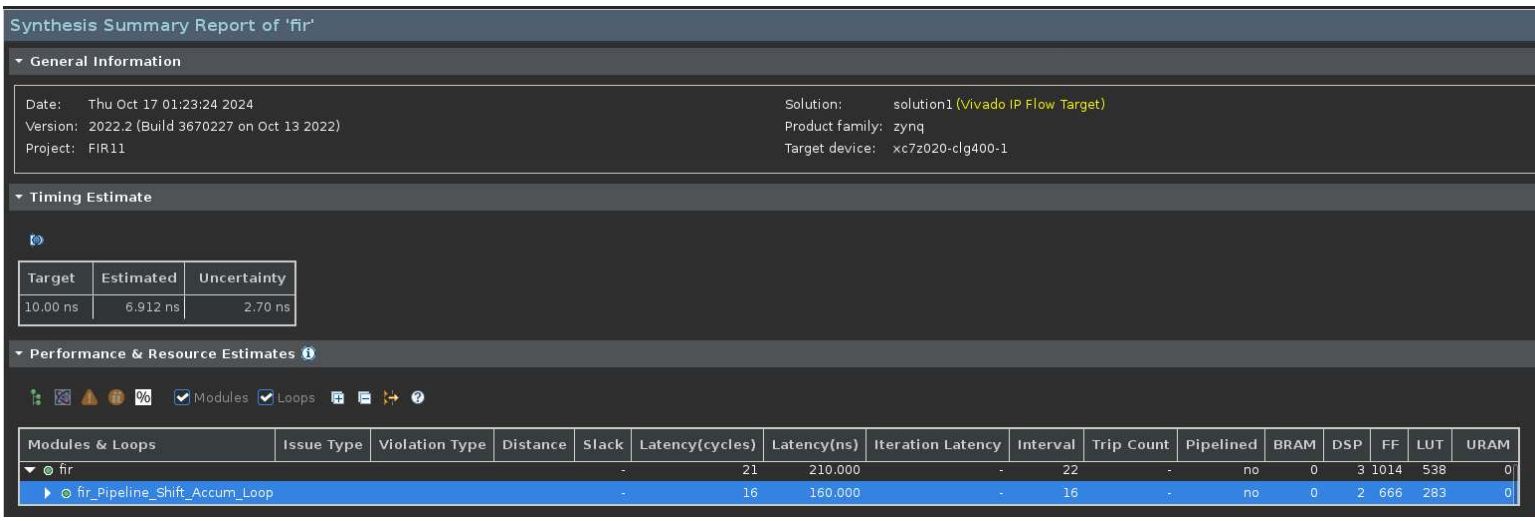
C SYNTESIS

```
Starting C synthesis ...
/tools/Xilinx/Vitis_HLS/2022.2/bin/vitis_hls /home/wes_2025_r_lizarraga/project1/FIR11/solution1/csynth.tcl
INFO: [HLS 200-10] Running '/tools/Xilinx/Vitis_HLS/2022.2/bin/unwrapped/lnx64.o/vitis_hls'
INFO: [HLS 200-10] For user 'wes_2025_r_lizarraga' on host 'waiter' (Linux_x86_64 version 5.15.0-122-generic) on Thu Oct 17 01:23:11 PDT
2024
INFO: [HLS 200-10] On os Ubuntu 22.04.5 LTS
INFO: [HLS 200-10] In directory '/home/wes_2025_r_lizarraga/project1'
Sourcing Tcl script '/home/wes_2025_r_lizarraga/project1/FIR11/solution1/csynth.tcl'
INFO: [HLS 200-1510] Running: source /home/wes_2025_r_lizarraga/project1/FIR11/solution1/csynth.tcl
INFO: [HLS 200-1510] Running: open_project FIR11
INFO: [HLS 200-10] Opening project '/home/wes_2025_r_lizarraga/project1/FIR11'.
INFO: [HLS 200-1510] Running: set_top fir
INFO: [HLS 200-1510] Running: add_files fir.cpp
INFO: [HLS 200-10] Adding design file 'fir.cpp' to the project
INFO: [HLS 200-1510] Running: add_files fir.h
INFO: [HLS 200-10] Adding design file 'fir.h' to the project


INFO: [HLS 200-1510] Running: add_files -tb fir_test.cpp
INFO: [HLS 200-10] Adding test bench file 'fir_test.cpp' to the project
INFO: [HLS 200-1510] Running: open_solution solution1 -flow_target vivado
INFO: [HLS 200-10] Opening solution '/home/wes_2025_r_lizarraga/project1/FIR11/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1611] Setting target device to 'xc7z020-clg400-1'
INFO: [HLS 200-1505] Using flow_target 'vivado'
Resolution: For help on HLS 200-1505 see www.xilinx.com/cgi-bin/docs/rdoc?v=2022.2;t=hls+guidance;d=200-1505.html
INFO: [HLS 200-1510] Running: set_part xc7z020clg400-1
INFO: [HLS 200-1510] Running: create_clock -period 10 -name default
INFO: [HLS 200-1510] Running: source ./FIR11/solution1/directives.tcl
INFO: [HLS 200-1510] Running: set_directive_interface -mode s_axilite fir
INFO: [HLS 200-1510] Running: set_directive_interface -mode s_axilite fir y
INFO: [HLS 200-1510] Running: set_directive_interface -mode s_axilite fir x
INFO: [HLS 200-1510] Running: csynth_design
Running Dispatch Server on port: 40583
INFO: [HLS 200-111] Finished File checks and directory preparation: CPU user time: 0.01 seconds. CPU system time: 0.01 seconds. Elapsed
time: 10.01 seconds; current allocated memory: 214.242 MB.
INFO: [HLS 200-10] Analyzing design file 'fir.cpp' ...
INFO: [HLS 200-111] Finished Source Code Analysis and Preprocessing: CPU user time: 0.11 seconds. CPU system time: 0.28 seconds. Elapsed
time: 0.4 seconds; current allocated memory: 214.242 MB.
INFO: [HLS 200-777] Using interface defaults for 'Vivado' flow target.
INFO: [HLS 200-111] Finished Compiling Optimization and Transform: CPU user time: 1.54 seconds. CPU system time: 0.37 seconds. Elapsed time:
1.91 seconds; current allocated memory: 214.570 MB.
INFO: [HLS 200-111] Finished Checking Pragmas: CPU user time: 0 seconds. CPU system time: 0 seconds. Elapsed time: 0 seconds; current
allocated memory: 214.570 MB.
INFO: [HLS 200-10] Starting code transformations ...
INFO: [HLS 200-111] Finished Standard Transforms: CPU user time: 0 seconds. CPU system time: 0.01 seconds. Elapsed time: 0.01 seconds;
current allocated memory: 215.191 MB.
INFO: [HLS 200-10] Checking synthesizability ...
INFO: [HLS 200-111] Finished Checking Synthesizability: CPU user time: 0.01 seconds. CPU system time: 0 seconds. Elapsed time: 0.01 seconds;
current allocated memory: 215.250 MB.
INFO: [XFORM 203-510] Pipelining loop 'Shift_Accum_Loop' (fir.cpp:24) in function 'fir' automatically.
INFO: [HLS 200-111] Finished Loop, function and other optimizations: CPU user time: 0.01 seconds. CPU system time: 0.01 seconds. Elapsed
time: 0.02 seconds; current allocated memory: 236.707 MB.
```

```
INFO: [HLS 200-472] Inferring partial write operation for 'shift_reg' (fir.cpp:31:17)
INFO: [HLS 200-472] Inferring partial write operation for 'shift_reg' (fir.cpp:33:17)
INFO: [HLS 200-111] Finished Architecture Synthesis: CPU user time: 0 seconds. CPU system time: 0 seconds. Elapsed time: 0.01 seconds;
current allocated memory: 236.707 MB.
INFO: [HLS 200-10] Starting hardware synthesis ...
INFO: [HLS 200-10] Synthesizing 'fir' ...
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'fir_Pipeline_Shift_Accum_Loop'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-61] Pipelining loop 'Shift_Accum_Loop'.
INFO: [HLS 200-1470] Pipelining result : Target II = NA, Final II = 1, Depth = 5, loop 'Shift_Accum_Loop'
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111] Finished Scheduling: CPU user time: 0.01 seconds. CPU system time: 0.03 seconds. Elapsed time: 0.03 seconds; current
allocated memory: 237.992 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] Finished Binding: CPU user time: 0.02 seconds. CPU system time: 0 seconds. Elapsed time: 0.02 seconds; current allocated
memory: 237.992 MB.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'fir'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111] Finished Scheduling: CPU user time: 0.02 seconds. CPU system time: 0 seconds. Elapsed time: 0.02 seconds; current
allocated memory: 238.184 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] Finished Binding: CPU user time: 0.01 seconds. CPU system time: 0 seconds. Elapsed time: 0.02 seconds; current allocated
memory: 238.184 MB.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-10] -- Generating RTL for module 'fir_Pipeline_Shift_Accum_Loop'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-1030] Apply Unified Pipeline Control on module 'fir_Pipeline_Shift_Accum_Loop' pipeline 'Shift_Accum_Loop' pipeline type
'loop pipeline'
INFO: [RTGEN 206-100] Generating core module 'mul_11s_32s_32_2_1': 1 instance(s).
INFO: [RTGEN 206-100] Finished creating RTL model for 'fir_Pipeline_Shift_Accum_Loop'.
INFO: [RTMG 210-278] Implementing memory 'fir_fir_Pipeline_Shift_Accum_Loop_shift_reg_RAM_AUTO_1R1W' using auto RAMs.
INFO: [RTMG 210-279] Implementing memory 'fir_fir_Pipeline_Shift_Accum_Loop_fir_int_int_c_ROM_AUTO_1R' using auto ROMs.
INFO: [HLS 200-111] Finished Creating RTL model: CPU user time: 0.02 seconds. CPU system time: 0 seconds. Elapsed time: 0.02 seconds;
current allocated memory: 238.184 MB.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-10] -- Generating RTL for module 'fir'
INFO: [HLS 200-10] ----------------------------------------------------------------
WARNING: [RTGEN 206-101] Design contains AXI ports. Reset is fixed to synchronous and active low.
INFO: [RTGEN 206-500] Setting interface mode on port 'fir/y' to 's_axilite & ap_vld'.
INFO: [RTGEN 206-500] Setting interface mode on port 'fir/x' to 's_axilite & ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on function 'fir' to 's_axilite & ap_ctrl_hs'.
WARNING: [RTGEN 206-101] Global array 'fir_int_int_c' will not be exposed as RTL port.
INFO: [RTGEN 206-100] Bundling port 'y', 'x' and 'return' to AXI-Lite port control.
INFO: [RTGEN 206-100] Generating core module 'mul_32s_7ns_32_2_1': 1 instance(s).
INFO: [RTGEN 206-100] Finished creating RTL model for 'fir'.
INFO: [HLS 200-111] Finished Creating RTL model: CPU user time: 0.04 seconds. CPU system time: 0 seconds. Elapsed time: 0.04 seconds;
current allocated memory: 239.223 MB.
INFO: [HLS 200-111] Finished Generating all RTL models: CPU user time: 0.09 seconds. CPU system time: 0 seconds. Elapsed time: 0.09 seconds;
current allocated memory: 242.922 MB.
INFO: [HLS 200-111] Finished Updating report files: CPU user time: 0.1 seconds. CPU system time: 0.02 seconds. Elapsed time: 0.12 seconds;
current allocated memory: 250.723 MB.
INFO: [VHDL 208-304] Generating VHDL RTL for fir.
INFO: [VLOG 209-307] Generating Verilog RTL for fir.
INFO: [HLS 200-790] **** Loop Constraint Status: All loop constraints were satisfied.
INFO: [HLS 200-789] **** Estimated Fmax: 144.68 MHz
INFO: [HLS 200-111] Finished Command csynth_design CPU user time: 2 seconds. CPU system time: 0.73 seconds. Elapsed time: 2.73 seconds;
current allocated memory: 36.625 MB.
INFO: [HLS 200-112] Total CPU user time: 2.6 seconds. Total CPU system time: 0.94 seconds. Total elapsed time: 13.45 seconds; peak allocated
memory: 250.867 MB.
Finished C synthesis.
```



Back on vitis_hls to run C-Synthesis



Synthesis Summary Report of 'fir'

General Information

Date: Thu Oct 17 01:23:24 2024
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: FIR11
Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	6.912 ns	2.70 ns

Performance & Resource Estimates

Modules Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fir				-	21	210.000	-	22	-	no	0	3	1014	538	0
fir_Pipeline_Shift_Accum_Loop				-	16	160.000	-	16	-	no	0	2	666	283	0

6) FIR128 Instructions

You must complete the following tasks:

1. Implement a functionally correct, but not optimized, 128-tap FIR filter. This is your baseline implementation. Use the provided script.tcl to create your project. As you attempt each optimization according to the questions below, think about what other optimizations would work well in conjunction with them.

wes_2025_r_lizarraga@waiter:~/project1/FIR128$ ls -l
total 24
-rw-rw-r-- 1 wes_2025_r_lizarraga wes_2025_r_lizarraga 836 Oct 18 09:39 fir.cpp
-rw-rw-r-- 1 wes_2025_r_lizarraga wes_2025_r_lizarraga 265 Oct 18 09:39 fir.h
-rw-rw-r-- 1 wes_2025_r_lizarraga wes_2025_r_lizarraga 1340 Oct 18 09:39 fir_test.cpp
-rw-rw-r-- 1 wes_2025_r_lizarraga wes_2025_r_lizarraga 2716 Oct 18 09:39 input.dat
-rw-rw-r-- 1 wes_2025_r_lizarraga wes_2025_r_lizarraga 3813 Oct 18 09:39 out.gold.dat
-rw-rw-r-- 1 wes_2025_r_lizarraga wes_2025_r_lizarraga 497 Oct 18 09:39 script.tcl

wes_2025_r_lizarraga@waiter:~/project1/FIR128$ vitis_hls -f script.tcl

CSIM for FIR128 base code

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
Compiling ../../../../fir_test.cpp in debug mode
Compiling ../../../../fir.cpp in debug mode
Generating csim.exe
INFO: [SIM 1] CSim file generation done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************

Starting C simulation ...
/tools/Xilinx/Vitis_HLS/2022.2/bin/vitis_hls
/home/wes_2025_r_lizarraga/project1/FIR128/FIR128/FIR128/csim.tcl
INFO: [HLS 200-10] Running '/tools/Xilinx/Vitis_HLS/2022.2/bin/unwrapped/lnx64.o/vitis_hls'
INFO: [HLS 200-10] For user 'wes_2025_r_lizarraga' on host 'waiter' (Linux_x86_64 version
5.15.0-122-generic) on Fri Oct 18 17:26:14 PDT 2024
INFO: [HLS 200-10] On os Ubuntu 22.04.5 LTS
INFO: [HLS 200-10] In directory '/home/wes_2025_r_lizarraga/project1/FIR128'
Sourcing Tcl script '/home/wes_2025_r_lizarraga/project1/FIR128/FIR128/FIR128/csim.tcl'
INFO: [HLS 200-1510] Running: source
/home/wes_2025_r_lizarraga/project1/FIR128/FIR128/FIR128/csim.tcl
INFO: [HLS 200-1510] Running: open_project FIR128
INFO: [HLS 200-10] Opening project '/home/wes_2025_r_lizarraga/project1/FIR128/FIR128'.
INFO: [HLS 200-1510] Running: set_top fir
INFO: [HLS 200-1510] Running: add_files fir.h
INFO: [HLS 200-10] Adding design file 'fir.h' to the project
INFO: [HLS 200-1510] Running: add_files fir.cpp
INFO: [HLS 200-10] Adding design file 'fir.cpp' to the project
INFO: [HLS 200-1510] Running: add_files -tb fir_test.cpp -cflags -Wno-unknown-pragmas -
csimflags -Wno-unknown-pragmas
INFO: [HLS 200-10] Adding test bench file 'fir_test.cpp' to the project
INFO: [HLS 200-1510] Running: open_solution FIR128 -flow_target vivado
INFO: [HLS 200-10] Opening solution
'/home/wes_2025_r_lizarraga/project1/FIR128/FIR128/FIR128'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1611] Setting target device to 'xc7z020-clg400-1'
INFO: [HLS 200-1505] Using flow_target 'vivado'
```

```
Resolution: For help on HLS 200-1505 see www.xilinx.com/cgi-
bin/docs/rdoc?v=2022.2;t=hls+guidance;d=200-1505.html
INFO: [HLS 200-1510] Running: set_part xc7z020clg400-1
INFO: [HLS 200-1510] Running: create_clock -period 10 -name default
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-1510] Running: source ./FIR128/FIR128/directives.tcl
INFO: [HLS 200-1510] Running: set_directive_top -name fir fir
INFO: [HLS 200-1510] Running: csim_design -clean -setup -quiet
Running Dispatch Server on port: 44237
INFO: [SIM 211-2] *************** CSIM start ***************
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
Compiling ../../../../fir_test.cpp in debug mode
Compiling ../../../../fir.cpp in debug mode
Generating csim.exe
INFO: [SIM 211-1] CSim file generation done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
INFO: [HLS 200-111] Finished Command csim_design CPU user time: 0.07 seconds. CPU system time:
0.12 seconds. Elapsed time: 0.19 seconds; current allocated memory: 0.000 MB.
INFO: [HLS 200-112] Total CPU user time: 0.69 seconds. Total CPU system time: 0.3 seconds.
Total elapsed time: 10.92 seconds; peak allocated memory: 214.363 MB.
Finished C simulation.
```

- fir.cpp:

fir_test.cpp: (SAMPLES=600)

```
for (i=0;i<SAMPLES;i++) {
  fscanf(fin,"%d",&signal);
  //Call the HLS block
  fir(&output,signal);
  // Save the results.
  fprintf(fp,"%d\n",output);
  printf("%i %d %d\n",i,signal,output);
}
```

Project: fir

Timing Estimate

Target	Estimated	Uncertainty
4.00 ns	2.893 ns	1.08 ns

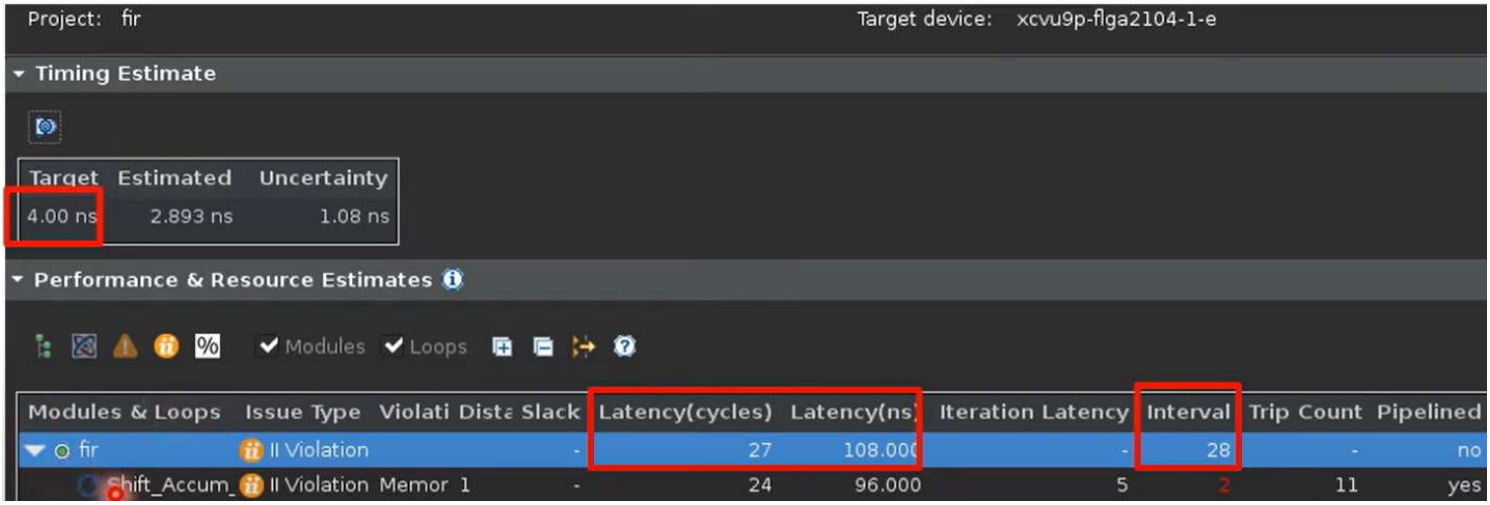
Performance & Resource Estimates

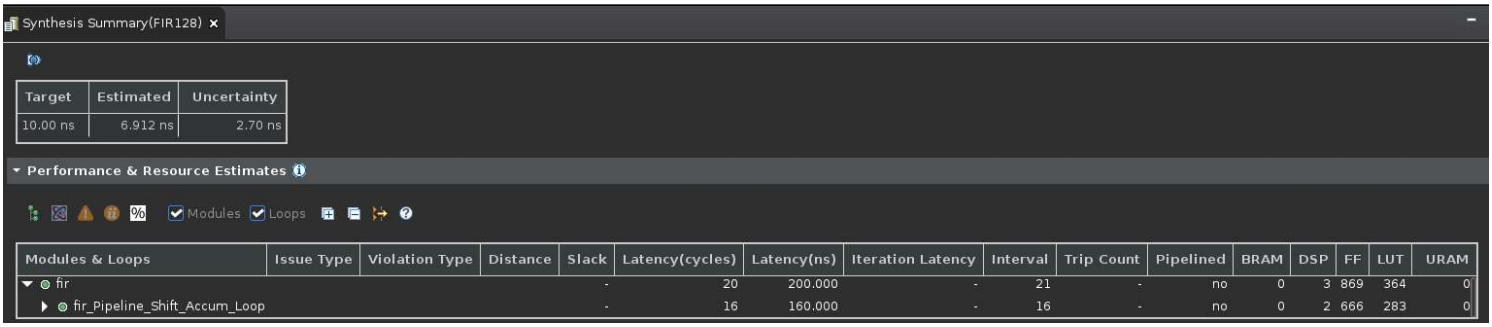
Modules ✔ Loops

Modules & Loops	Issue Type	Violati	Dista	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined
fir	II Violation			-	27	108.00(	-	28	-	no
Shift_Accum_	II Violation	Memor	1	-	24	96.000	5	2	11	yes

Baseline performance:

600 samples * 28 cycles = 16,800 cycles





2. Next, generate one or multiple designs to help you answer your report's questions. You should reference the design you generated for your experiment in your answers. You can reference the same design from multiple answers. Your resulting code must always be functionally correct (i.e., match the golden output). In your report, you must explain the effect of the following optimizations on your design. You can test other optimizations as you'd like, but you can leave these out of your report. For every design you include in your report, you can report the corresponding throughput instead of the estimated clock period and latency.
3. Finally, for Q6, generate your best architecture by combining any number of optimizations that you wish. You can use what you learned from your designs for Q1-Q5.
4. Your report should only include the answers to the following questions.

You must reference a design or multiple designs for the following questions. The source code in your design should have all the necessary pragmas. Please refer to Chapter 2 in the pp4fpga textbook before starting this assignment.

Questions:

- **Question 1 - Variable Bitwidths:** You can specify a precise data type for each variable in your design. There many different data types including floating point, integer, fixed point, all with varying bitwidths and options. The data type provides a tradeoff between accuracy, resource usage, and performance. Change the bitwidth of the variables inside the function body (do not change the bitwidth of the parameters). How does the bitwidth affect the performance?

Variable Bitwidth defininitely affects performance inside of a function, the actual number of bits for these C data types may vary, but the C standard dictates minimum bit widths (e.g. an int is at least 16 bits).

The more bits, it would take longer time to resolve searches, transfer data and calculations

- How does it affect the resource usage?

One benefit of using smaller bitwidth is the amount of storage that the data type requires, in addition to that in an example of a large array, setting the bitwidth to the minimum necessary width is key to

have the performance we need and the resolution of the calculation results, for example some cases we don't need signed values (An 8 bit signed value allows numbers in the range [-128,127]), maybe just unsigned integers). Like if we need decimal precision, then we can use as float or double.

- What minimum data size can you use without losing accuracy (i.e., your results still match the golden output)?

Changing the variable array "shift_reg"
changing the data type using the notation:
intN_t (where N is 8, 16, 32, and 64, as defined in stdint.h, ap_uint<7>)
Found out the test keep passsing as long as the bitwidth to be greater than 2^7
Less than that will be failing many of the test.
There are two separate classes for unsigned and signed data types: Unsigned: ap uint<width> & Signed: ap int<width>

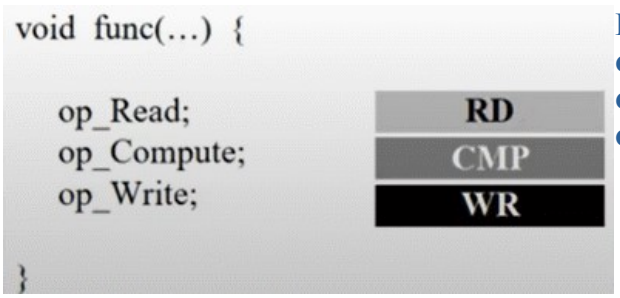
fir128_optimized1: Variable Bitwidths

```
ap_int<16> acc;
```

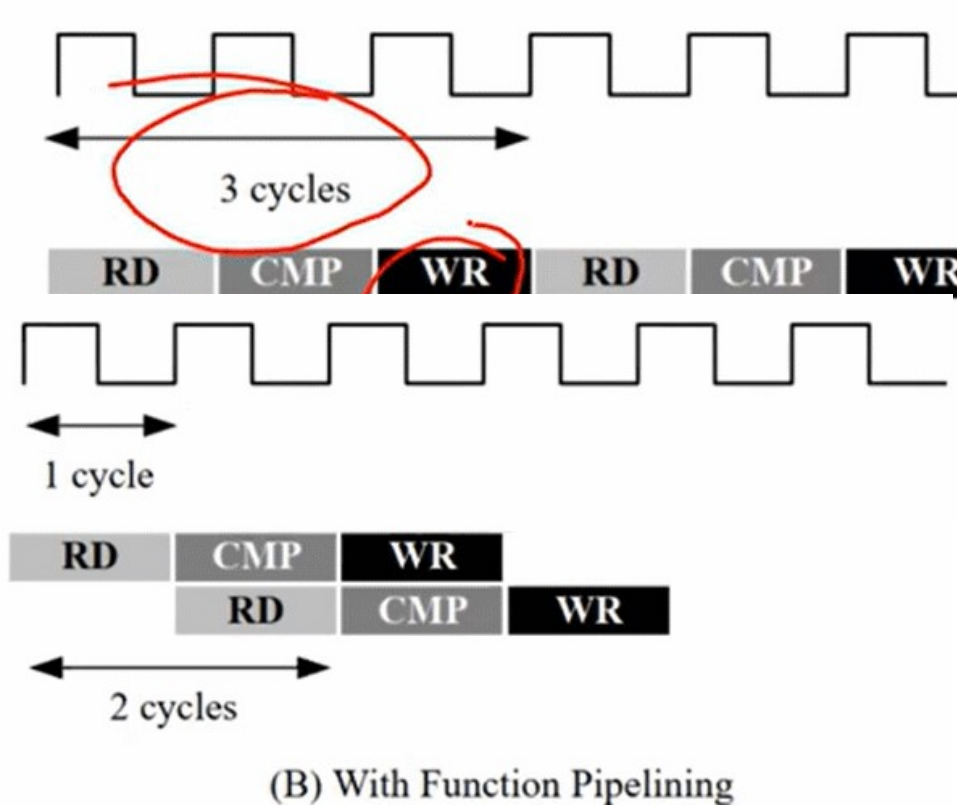
```
ap_uint<7> i;
```

```
#include "fir.h"

void fir (
  data_t *y,
  data_t x
  )
{

    coef_t c[N] = {10, 11, 11, 8, 3, -3, -8, -11, -11, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10,

    // Write your code here

    static ap_int<8> shift_reg[N];
    ap_int<16> acc;
    ap_uint<7> i;

    acc = 0;
    Shift_Accum_Loop:
    for (i = N-1; i >= 0; i--){
        if (i == 0){
            acc += x * c[0];
            shift_reg[0] = x;
        }else{
            shift_reg[i] = shift_reg[i-1];
            acc += shift_reg[i] * c[i];
        }
    }
    *y = acc;

}
```

- **Question 2 - Pipelining:** Pipelining increases throughput, typically at the cost of additional resources. The initiation interval (II) is an important design parameter that affects performance and resource usage. Explicitly set the loop initiation interval (II) starting at 1 and increasing in increments of 1 cycle. How does increasing the II affect the loop latency? What are the trends? At some point, setting the II to a larger value does not make sense. What is that value in this example? How would you calculate that value for a general for loop?



H LS Pipeline: Allows operations to happen concurrently: each execution step does not have to complete all operations before it begins the next operation



(B) With Function Pipelining

```
#pragma HLS pipeline II=1
```

Using HLS pipeline II=1, improved performance, but is using more area/ resources

fir128_optimized2: **Loop pipeline**

```
#pragma HLS pipeline
shift_reg[i] = shift_reg[i-1];
```

- **Question 3 - Removing Conditional Statements:** If/else statements and other conditionals can limit the possible parallelism and often require additional resources. Rewriting the code to remove them can make the resulting design more efficient. This is known as code hoisting.
  Rewrite the code to remove any conditional statements. Compare the designs with and without if/else condition. Is there a difference in performance and/or resource utilization? Does the presence of the conditional branch have any effect when the design is pipelined? If so, how and why?
  If statements have significant impact on processing time (increasing the II), because processor has to stop to compare and each individual IF statements gets multiplied by the number of iterations of the loop.
  Loop unrolling can be used to reduce the number of iterations of the loop, see e3xplanation **next**

fir128_optimized3: Removing Conditional Statements

```
coef_t c[N] = {10, 11, 11, 8, 3, -3, -8, -11, -11, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10

// Write your code here

static data_t shift_reg[N];
acc_t acc;
int i;

acc = 0;
Shift_Accum_Loop:
for (i = N-1; i >= 1; i--){
    shift_reg[i] = shift_reg[i-1];
    acc += shift_reg[i] * c[i];
}
acc += x * c[0];
shift_reg[0] = x;

*y = acc;
```

- **Question 4 - Loop Partitioning:** Dividing the loop into two or more separate loops may allow for each of those loops to be executed in parallel (via unrolling), enable loop-level pipelining, or provide other benefits. This may increase performance and resource usage.
  Is there an opportunity for loop partitioning in FIR filters? Compare your hardware designs before and after loop partitioning. What is the difference in performance? How does the number of resources change? Why?
  Implementing Loop unrolling by a factor of 4, it improved around 15% performance, unrolling to a greater factor (but being carefully of not going out of bounds), this would improve performance too.
  Optimization can be partitioned, in one section by #pragma pipeline and the other loop using loop unrolling

fir128_optimized4: Loop partitioning

```
for (i = N-1; i >= 1; i--){
        #pragma HLS pipeline
        shift_reg[i] = shift_reg[i-1];
}

shift_reg[0] = x;
for (i = N-1; i >= 1; i--){
    #pragma HLS unroll factor=4
    acc += shift_reg[i] * c[i];
}

acc += x * c[0];
*y = acc;
```

- **Question 5 - Memory Partitioning:** The storage of the arrays in memory plays an important role in area and performance. On one hand, you could put an array entirely in one memory (e.g., BRAM). But this limits the number of read and writes accesses per cycle. Or you can divide the array into two or more memories to increase the number of ports. Or you could instantiate each variable as a register allowing simultaneous access to all the variables at every clock cycle.
  Compare the memory partitioning parameters: block, cyclic, and complete. What is the difference in performance and resource usage (particularly with respect to BRAMs and FFs)? Which one gives the best performance? Why?
  n Vitis HLS, BRAMs and FFs are types of FPGA resources that can be used to implement array arguments and ports
  BRAMs
  Block RAMs can store more data than FFs. They are useful for storing large amounts of data that don't need fast throughput, such as buffer objects or ROM lookup tables. The ports of BRAMs can operate with separate clocks, which allows data to be transferred across clock domains.
  FFs
  FIFOs can be used instead of ping-pong buffers when data accesses are fully sequential. The size of FIFOs can be directly controlled, which is not possible with ping-pong buffers.
  Vitis HLS is a tool that allows users to synthesize C/C++ functions into RTL code to create FPGA algorithms.
  The Vitis HLS Synthesis Summary report includes information such as timing slack, latency, and initiation interval. The Vitis compiler also generates HLS Reports for each kernel, which contain details about the performance and logic usage of the hardware.
  For this homework I didn't have enough time to go through optimizing the memory partitioning, but I hope I can apply this in the coming projects

- **Question 6 - Best Design:** Combine any number of optimizations to get your best architecture. A design with high throughput will likely take a lot of resources. A design with small resource usage likely will have lower performance, but that could still be the best depending the application goals.
  In what way is it the best? What optimizations did you use to obtain this result?
  It is possible to create a design that outputs a result every cycle, i.e., get one sample per cycle, so a throughput of 100 MHz (assuming a 10 ns clock).

Here I described the optimization I was able to do, based on the shown techniques:

A. Reduce variable Bit widths
B. Removing Conditional Statements
C. Loop partitioning
D. Loop pipeline
E. Loop unrolling

Here's my optimized code:

```
coef_t c[N] = {10, 11, 11, 8, 3, -3, -8, -11, -11, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10, -10,

// Write your code here

static ap_int<8> shift_reg[N];
ap_int<16> acc;
ap_uint<7> i;

acc = 0;

for (i = N-1; i >= 1; i--){
        #pragma HLS pipeline ii=3
        shift_reg[i] = shift_reg[i-1];
}

shift_reg[0] = x;
for (i = N-1; i >= 1; i--){
    #pragma HLS unroll factor=4
    acc += shift_reg[i] * c[i];
}

acc += x * c[0];
*y = acc;
```

It is possible to improve the optimization by additional changes in the code, and combining the different techniques until we reached the desired target performance and resources usage (trade-offs)

It is possible that some optimizations have little (or no effect). Some optimizations may only work when used in combination with others. This is what makes the design space exploration process difficult.

- **Note**: You should use ap_int types if necessary for required bit width. You can read about ap_int from here or from section 2.10 of the textbook.

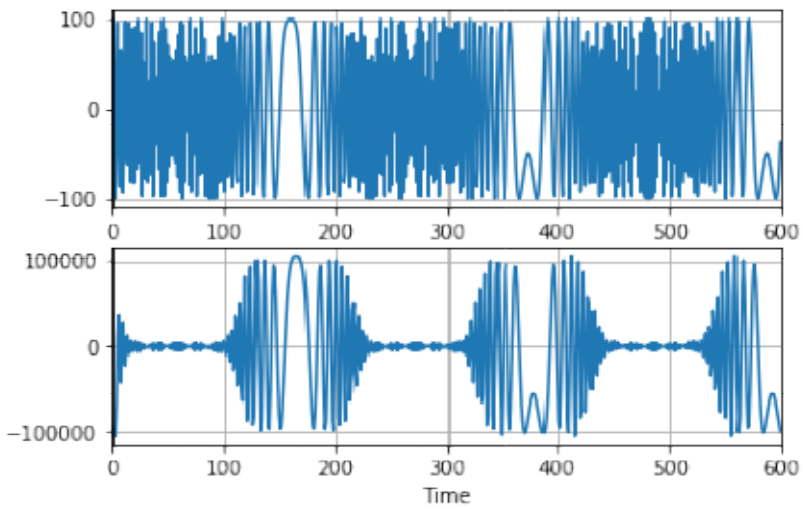
7) PYNQ Demo

The following are steps to implement your FIR11 HLS design on the PYNQ board. You will provide the input data (chirp signal) from the Notebook and get the output from the PL on PYNQ. To do that, you must write a *host_fir.ipynb* program.

The specific things you must do in this section are:

- Download an appropriate image for your board from here and write it to your SDCard (PYNQ-Z2 instructions).
- Go through Lab: Pynq Memory Mapped IO example and learn how to write an IP for PYNQ and interact with it.
- Implement your **11-tap** FIR design on PYNQ board.
- Write a host program *host_fir.ipynb*.

The expected output is as shown below:



```python
from pynq import Overlay
from pynq import MMIO
from pynq import allocate
import numpy as np
import matplotlib.pyplot as plt
import array


fs = 50000
t = 3 # time in seconds
data = np.random.uniform(-1,1,fs*t) # t-seconds random samples between -1 and 1
print(data)
# scale to 16 bit for data transfer
data_scaled = np.int16(data/np.max(np.abs(data)) * (2**15)-1)

# plot data
plt.figure(figsize=(8,3))
plt.plot(data)
plt.xlim(0,100)
plt.grid(True)
plt.tight_layout()
plt.show()


ol = Overlay("./design_1_wrapper.bit") # designate a bitstream to be flashed to the FPGA
ol.download() # flash the FPGA
fir_filter = MMIO(0x40000000, 0x10000) # (IP_BASE_ADDRESS, ADDRESS_RANGE), told to us in Vivado

in_buffer = allocate(shape=(len(data),),dtype=np.int16)
out_buffer = allocate(shape=(len(data),),dtype=np.int32)
my_array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

import numpy as np
my_numpy_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], dtype=np.int16)

i = 0
for x in range(11):
    fir_filter.write(0x20, my_array[i]) #
    i=i+1

import time
time.sleep(0.4)
i = 0
for x in range(11):
    print("output:", fir_filter.read(0x10)) # read corresponding output value from the output address of the fabric
    i=i+1


```

```
[ 0.37041574 -0.05092066 -0.23544514 ... -0.6665152  -0.08542
 -0.44788747]
```