The Abdus Salam International Centre for Theoretical Physics
www.ictp.it
Trieste, Italy

IAEA International Atomic Energy Agency

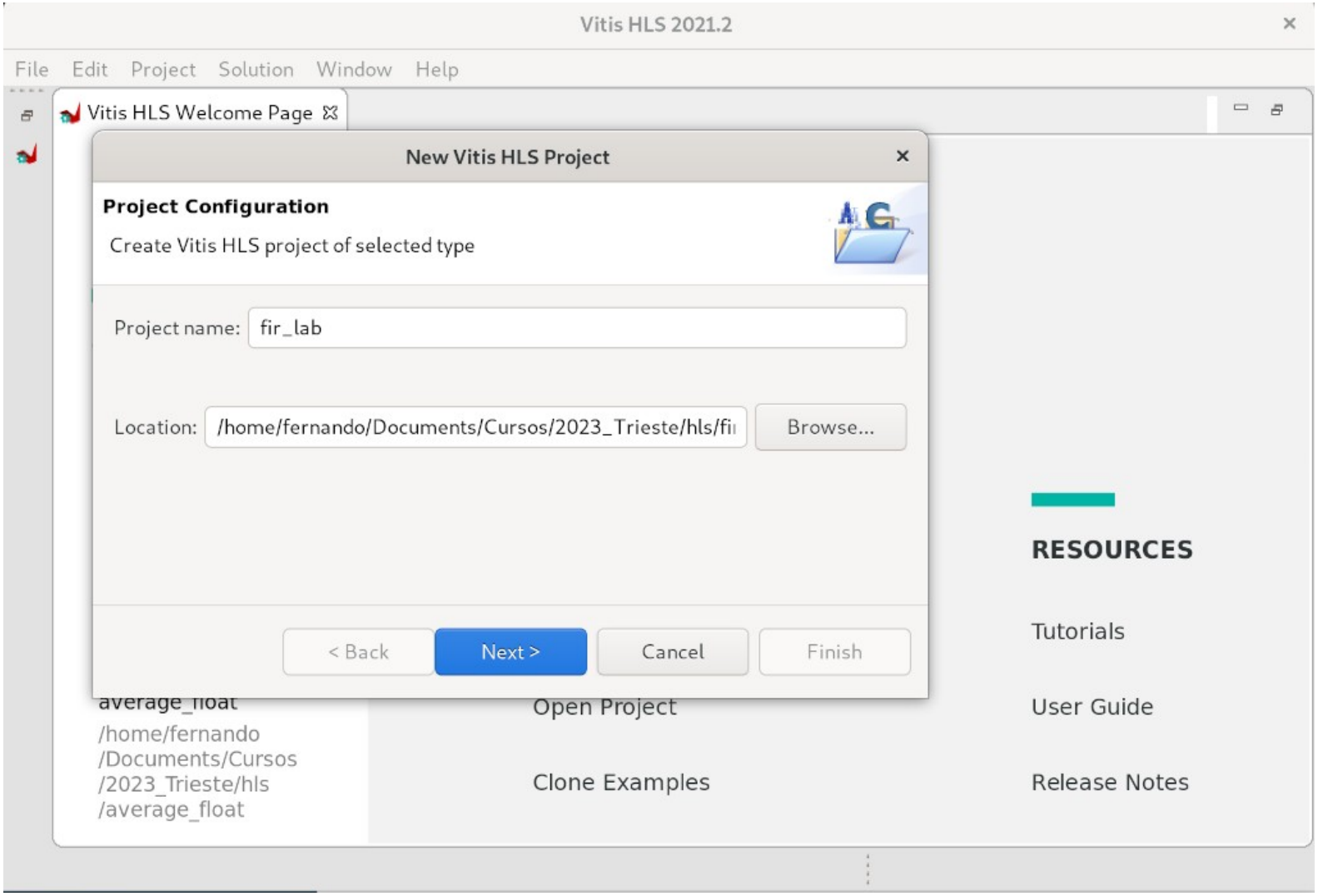
Joint ICTP-IAEA School on Systems-on-Chip Based on FPGA for Scientific Instrumentation and Reconfigurable Computing

HLS Demo

Fernando Rincón
University of Castilla-La Mancha
fernando.rincon@uclm.es

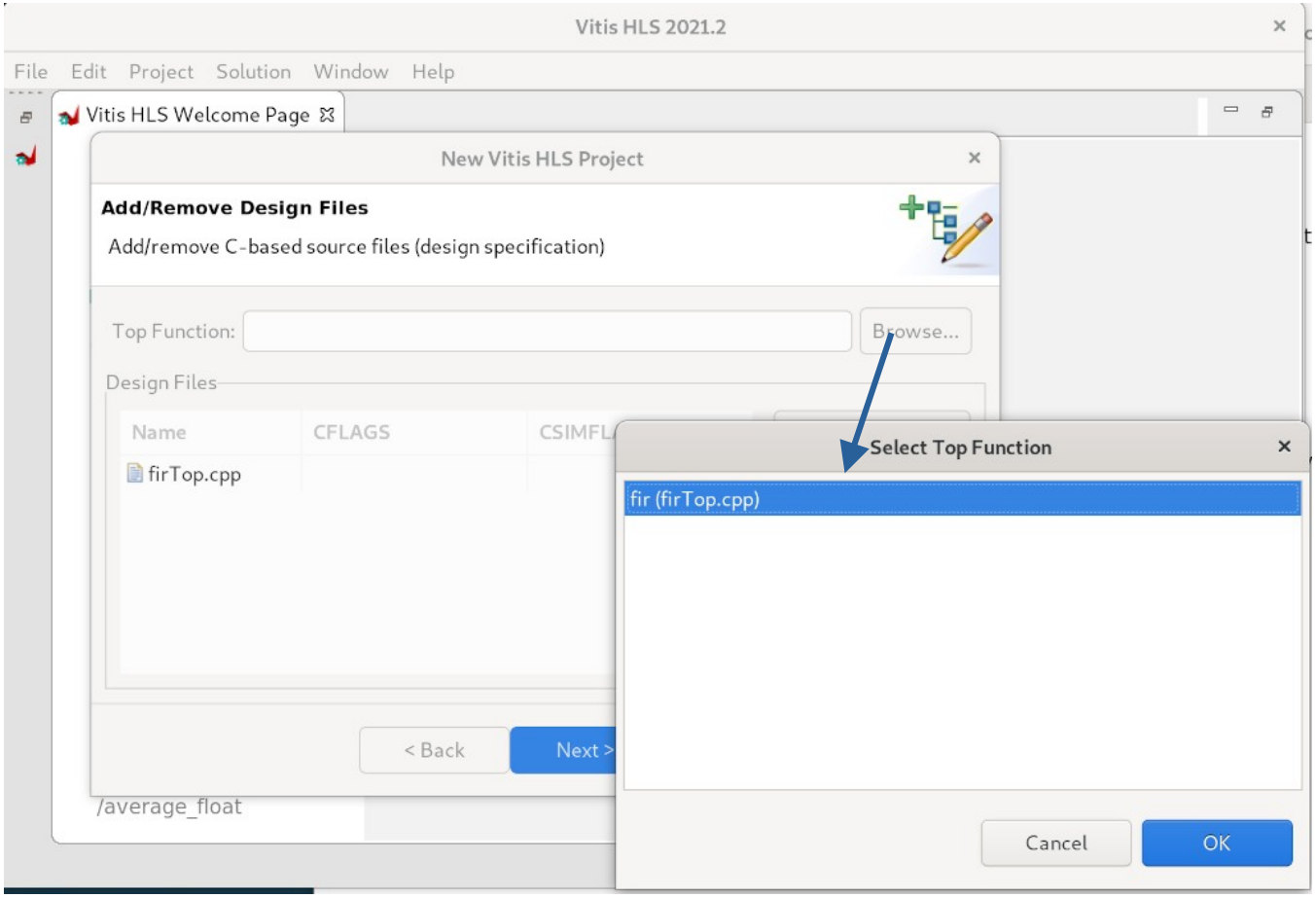
Project Creation

- File → New Project ...



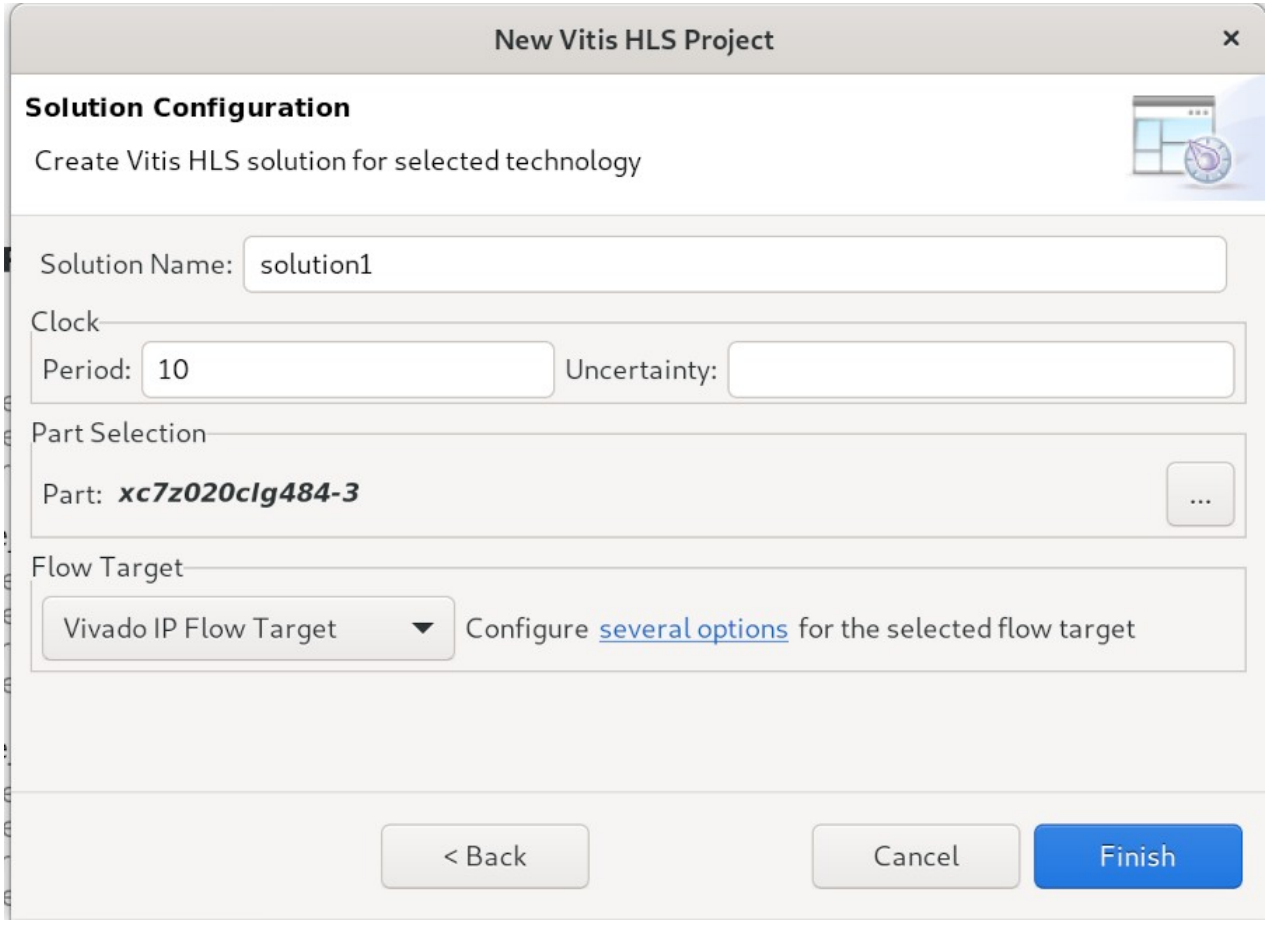
Project Creation

- Add firTop.cpp as the top file
  - Then select the top module



Project Creation

- Skip testbench
- And select Zedboard or xc7z020clg484-3 part

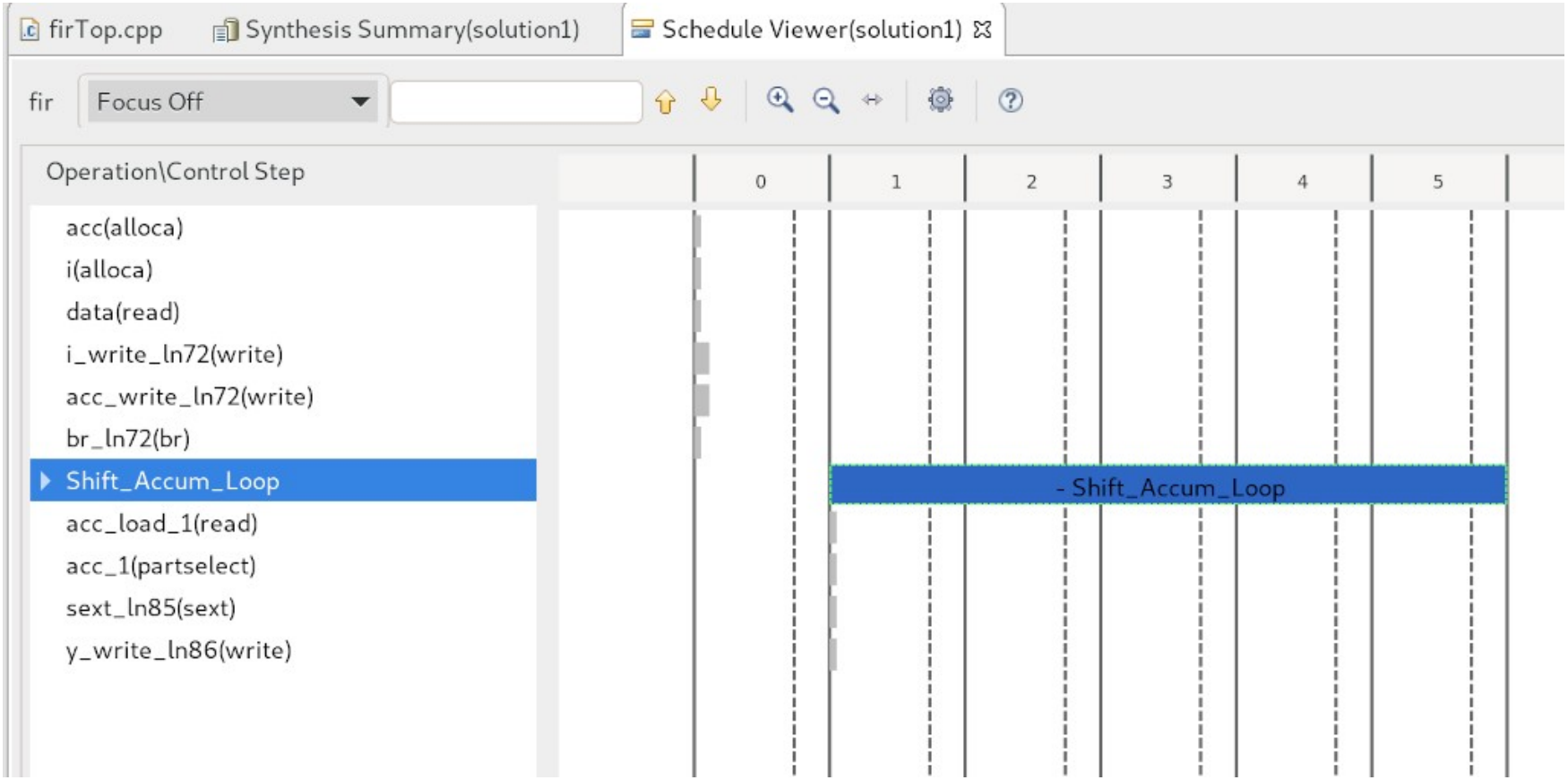


First synthesis

- Solution → Run C Synthesis → Active Solution
- Note down:
  - Estimated clock cycle
  - Fir latency
  - BRAM, DSPs, FFs and LUTs
- Note the issues at the module and loop
- Note the Hw ports

Schedule Analysis

- Loop latency = 5 cycles * 181 iterations = 905 cycles
- Expand the loop and find where time is spent



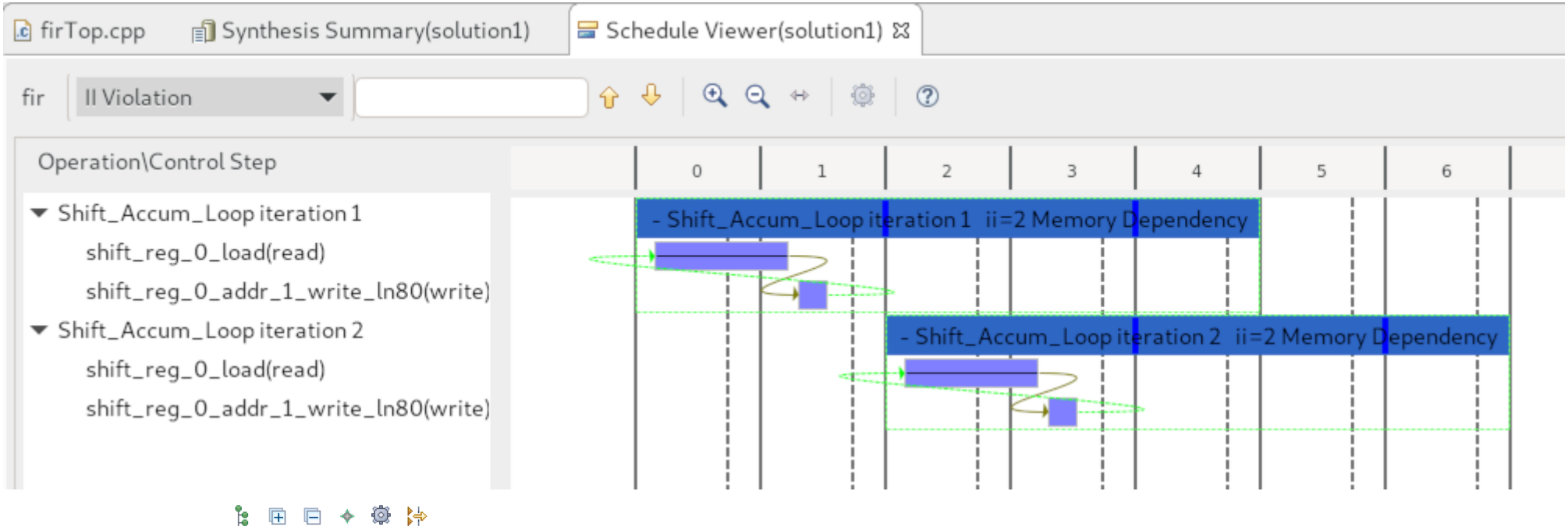
Let's pipeline the loop

- Replace the pragma disabling pipeline
  - #pragma HLS PIPELINE off
  - Pipeline is now the default for Vitis HLS synthesize
- Alternatively
  - #pragma HLS PIPELINE II=1
- Repeat the synthesis
- Note down the new latency and resource usage
- Did we achieve II=1?
  - Why not?

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fir	II Violation			-	366	3.660E3	-	367	-	no	3	2	206	299	0
Shift_Accum_Loop	II Violation	Memory Dependency	1	-	364	3.640E3	5	2	181	yes	-	-	-	-	-

Schedule Analysis

- Select Shift_Acum_Loop and with the context menu: Goto II Violation



No filter settings

Name	BRAM	URAM	Pragma	Variable	Storage	Impl	Latency
fir	3	-					
shift_reg_0_U	1	-		shift_reg_0	ram_1p	auto	1
shift_reg_1_U	1	-		shift_reg_1	ram_1p	auto	1
firCoeff_U	1	-		firCoeff	rom_1p	auto	1
Shift_Accum_Loop							

shift_reg partitioning

- Let's turn the shift_reg memory into a real shift register:
  - #pragma HLS ARRAY_PARTITION dim=1 type=complete variable=shift_reg
- This should avoid read/write contention
- But will increase resources

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fir				-	185	1.850E3	-	186	-	no	1	2	5981	1163	0
Shift_Accum_Loop				-	183	1.830E3	4	1	181	yes	-	-	-	-	-

Interfaces

- Observe the top level interface
  - start/stop protocol
- Observe input/output variables
  - Y bus with valid signal

▼ TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst	reset	ap_rst
ap_ctrl	ap_ctrl_hs	ap_done ap_idle ap_ready ap_start

▼ SW I/O Information

▼ Top Function Arguments

Argument	Direction	Datatype
y	out	int*
x	in	int*

▼ SW-to-HW Mapping

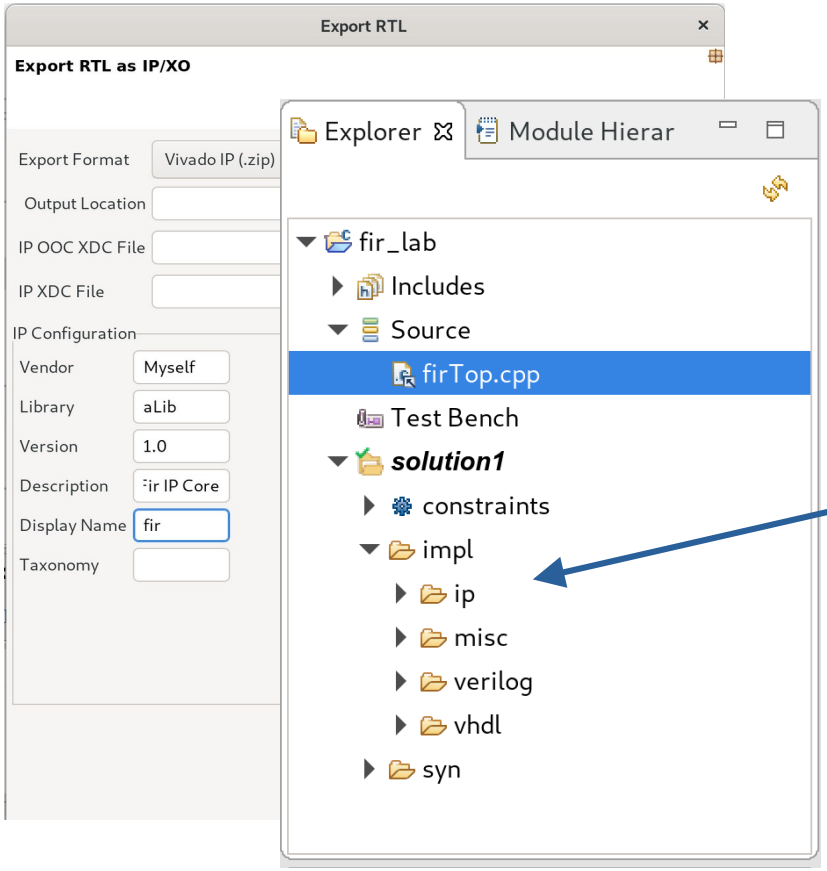
Argument	HW Interface	HW Type
y	y	port
y	y_ap_vld	port
x	x	port

Interfaces

- Apply pragmas to turn the module into an endless stream processor
  - Remove start/stop protocol:
    - #pragma HLS INTERFACE mode=ap_ctrl_hs port=return
  - Set x & y ports to stream interfaces
    - #pragma HLS INTERFACE mode=axis register_mode=both port=y register
    - #pragma HLS INTERFACE mode=axis register_mode=both port=x register
- Resynthesize and check the new interface

Export RTL

- To build the IP to be integrated in Vivado
- Solution → Export RTL



Add the path to this folder in Vivado as a new repository