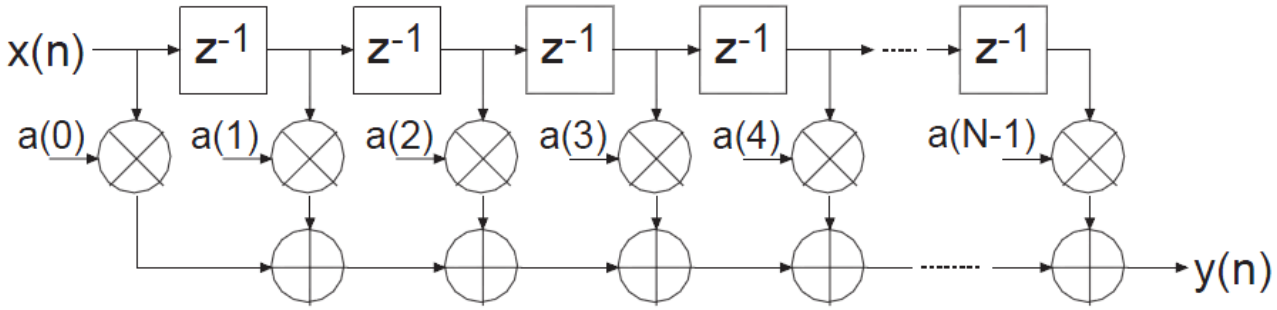
MUNI
FI

PA221: FIR Filter Verification

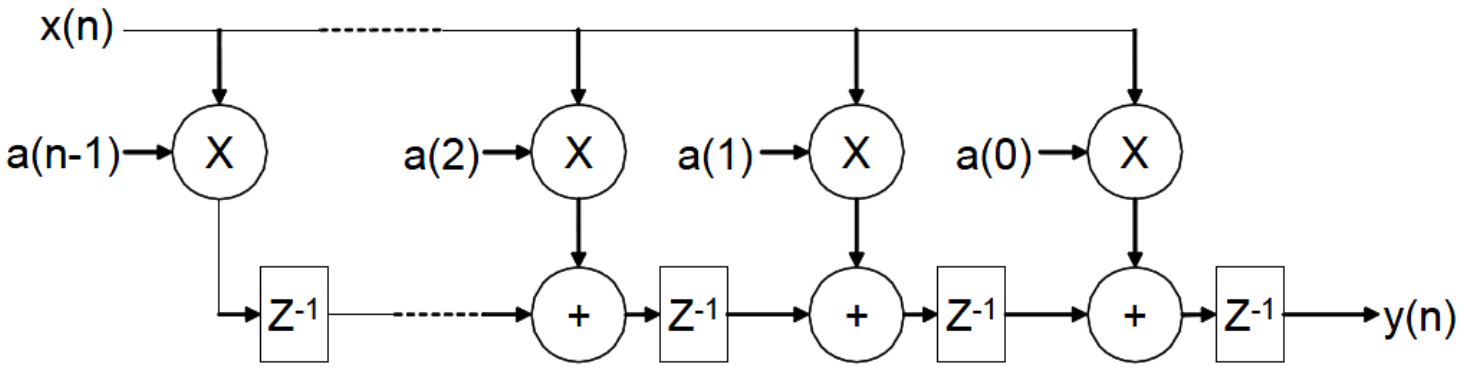
jan.kral@fi.muni.cz

FIR Filter

F_{MAX} = ?



Conventional Tapped Delay Line FIR Filter Representation



Transposed Direct-Form

$$y[n] = a[0]\,x[n] + a[1]\,x[n-1] + \cdots + a[N]\,x[n-N] = \sum_{i=0}^{N} a[i]\;x[n-i]$$

Assignment

FIR filter design and verification

- **Design** a FIR filter according to specification
  - use Matlab to generate filter coefficients
  - use Vivado IP core wizard to generate FIR filter module
- **Verify** the FIR functionality (automatic testbench)
  - read input data from a file and feed them to the filter
  - read reference output data from a file and compare it with actual filter output
  - report results of the verification to a **log** file, including PASS/FAIL message and total number of errors detected

FIR Filter Parameters

- Entity (IP core) name FIR_50k
- Low-pass FIR, Equiripple
- Fpass = 50 kHz, Fstop = 200 kHz
- Apass = 1dB, Astop = 40 dB
- Sampling frequency 6.25 MHz
- Clock frequency 50 MHz (=> 8 clock cycles per sample)
- Input: 9b, no fractional part (range -256 to +255)
- Output: 9b, no fractional part (range -256 to +255)

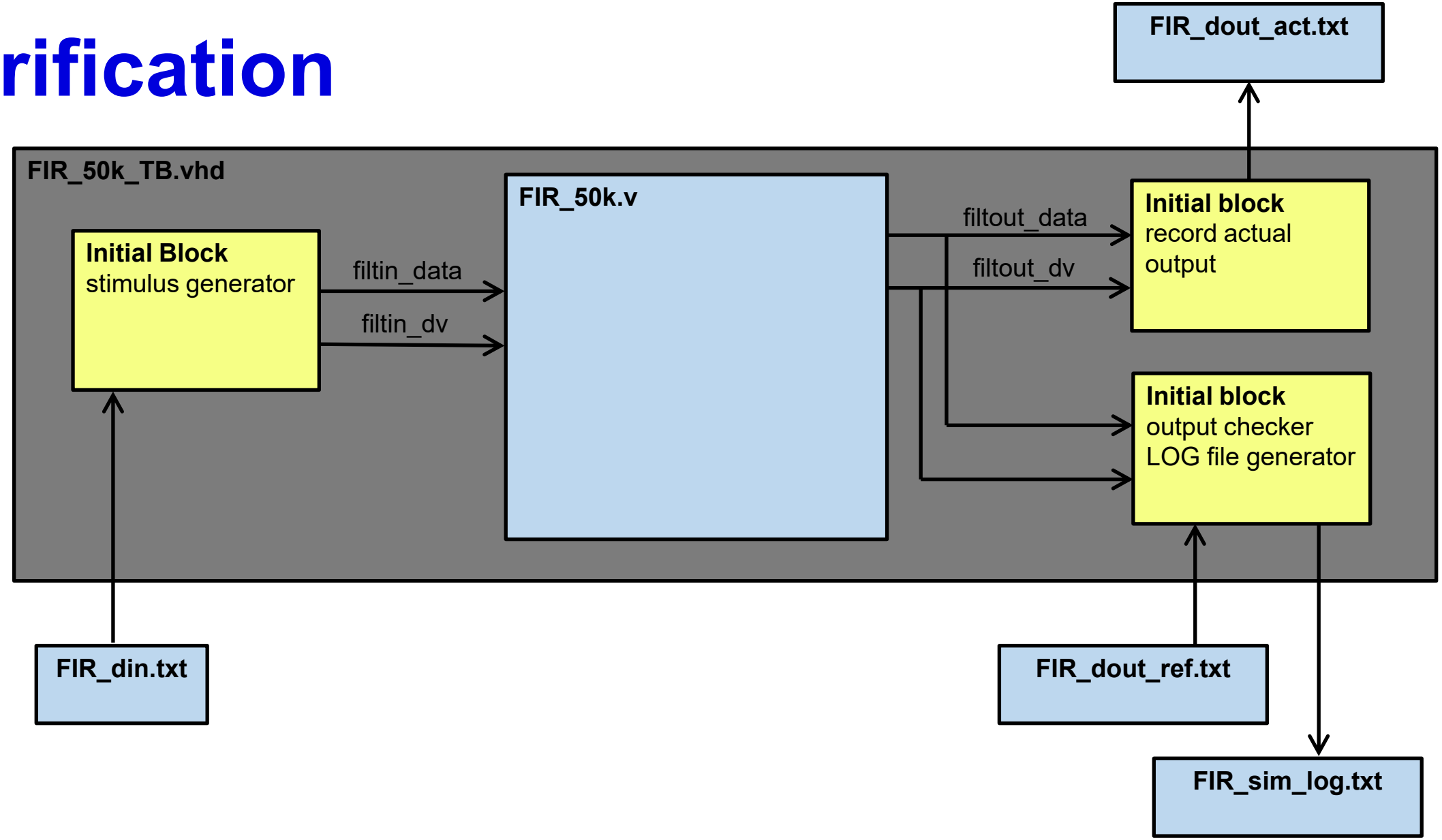
Verification: Self-testing testbench

- Automatically generates stimuli for the DUT (design under test)
- Automatically verifies correctness of the DUT outputs
- Generates a LOG file with simulation results

FIR Filter Parameters

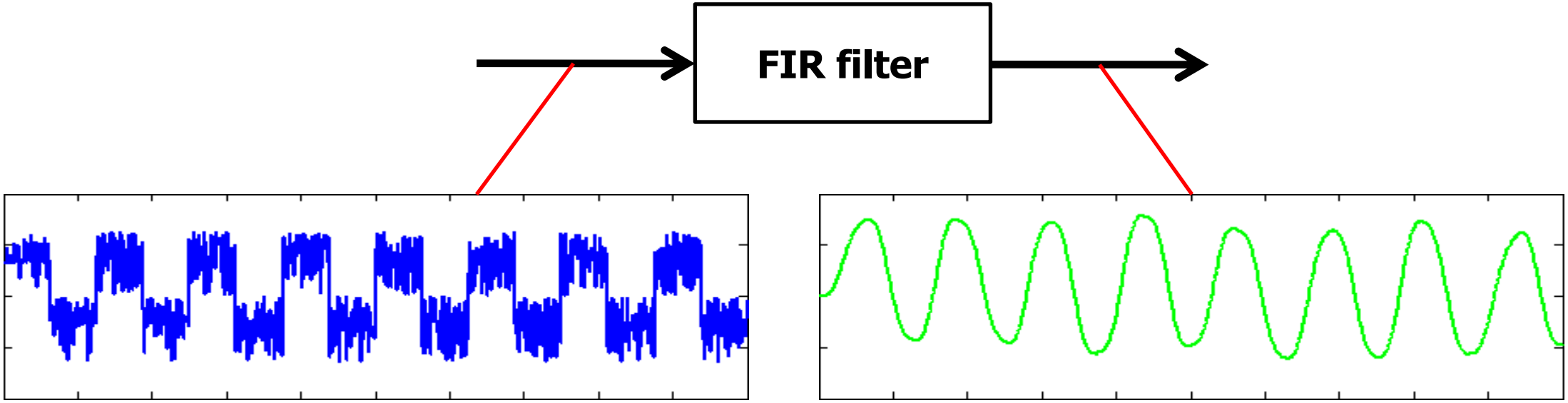
- Instantiate the design under test (DUT; FIR_50k)
- Create an initial block that reads data from **FIR_din.txt** file and feeds the data to DUT. Note that there should be one sample for each 8 clock cycles (50 MHz clock at 6.25 MHz sampling frequency).
- Create an initial block that writes data to **FIR_dout_act.txt** file whenever FIR output valid signal is asserted. Each sample on a line.
- Create an initial block that reads data from **FIR_dout_ref.txt** file and verifies data on the DUT output (whenever FIR output valid signal is asserted). Any discrepancy is reported to both a textual **log** file and simulator console as an error.
- Correctly finish the simulation (do not run forever).

Verification



Expected Waveforms

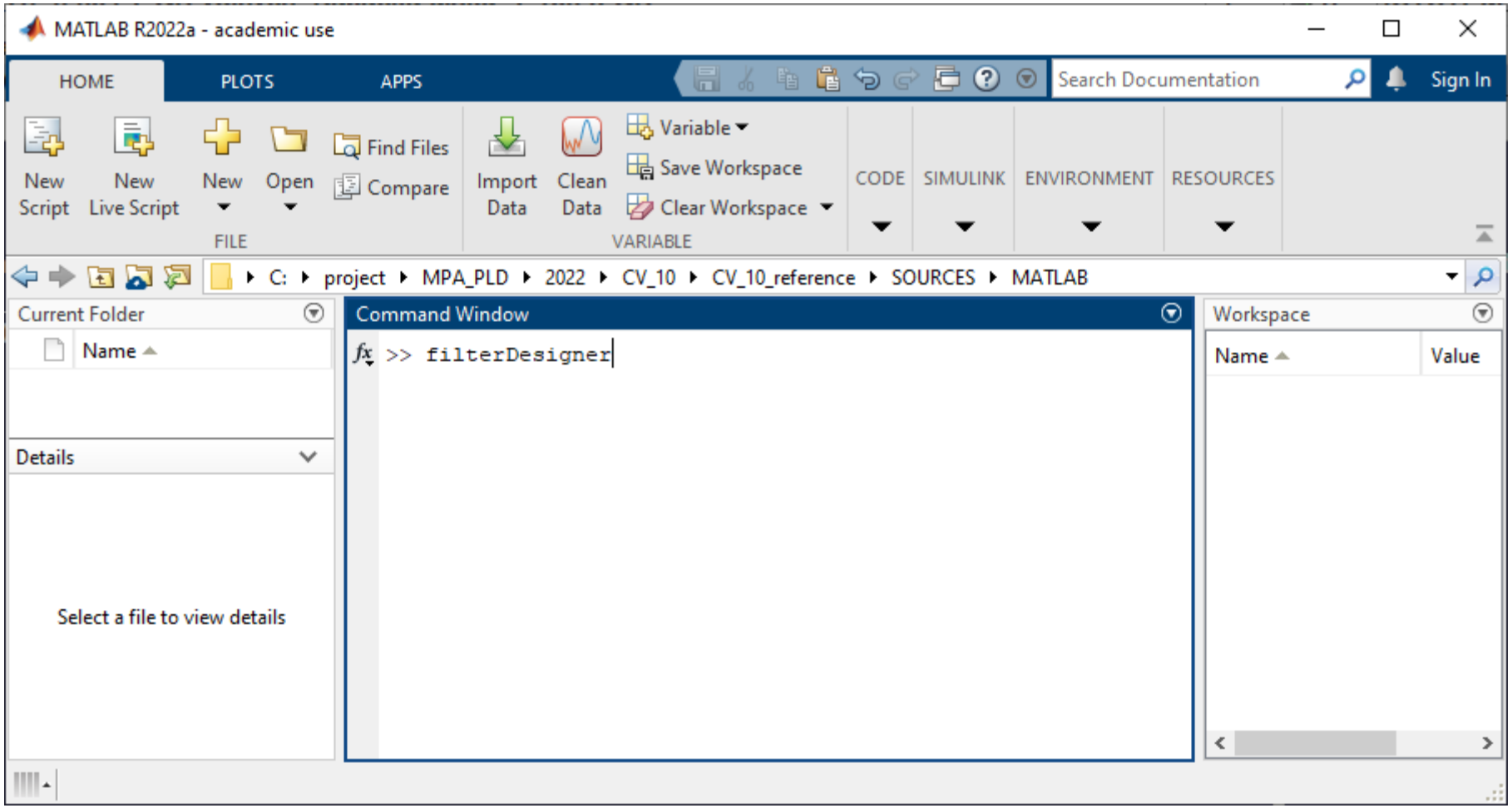
- See the analog waveforms in the ModelSim / Questa.



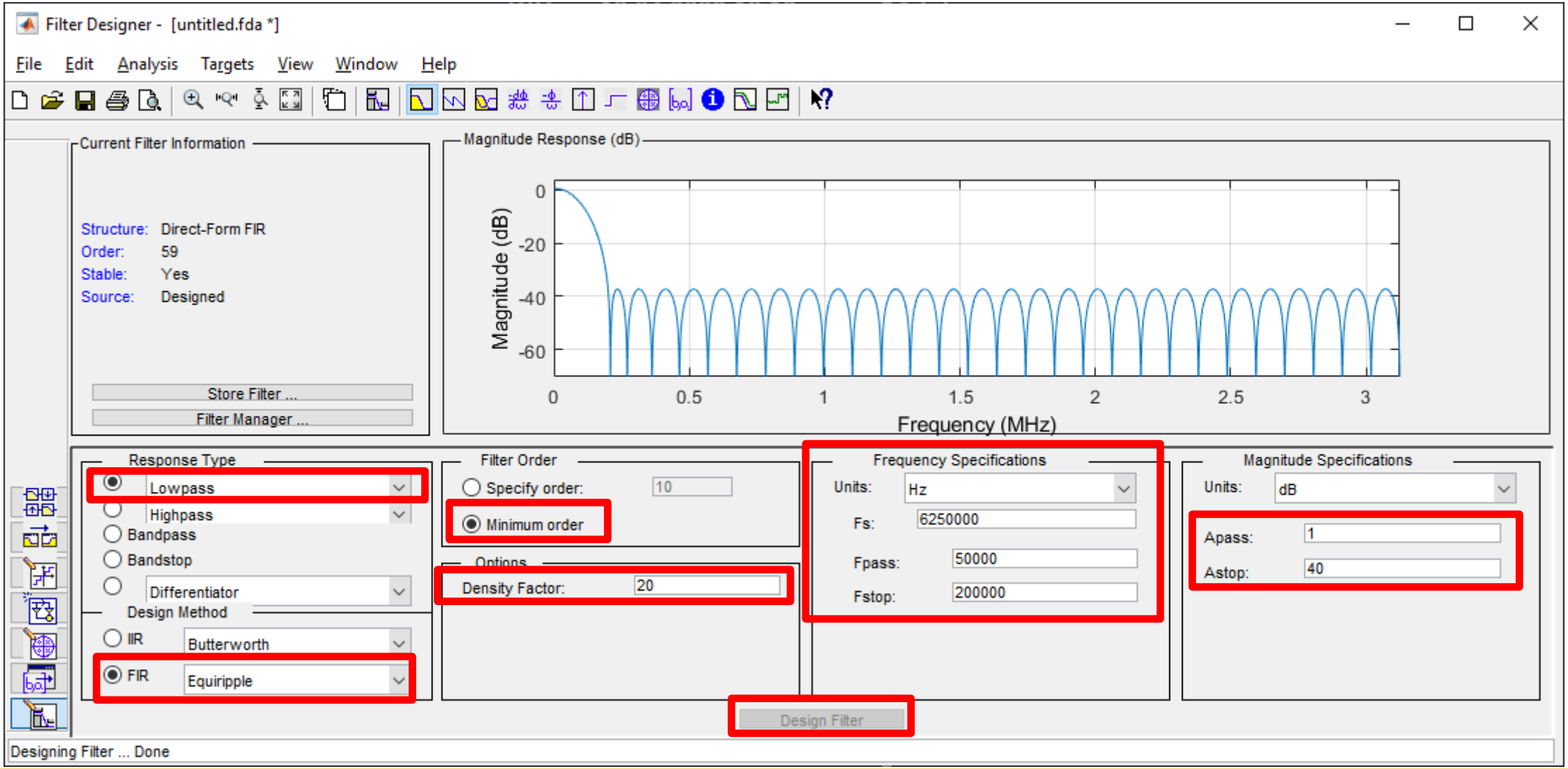
FIR filter design: filterDesigner

Matlab Filter Design & Analysis Tool

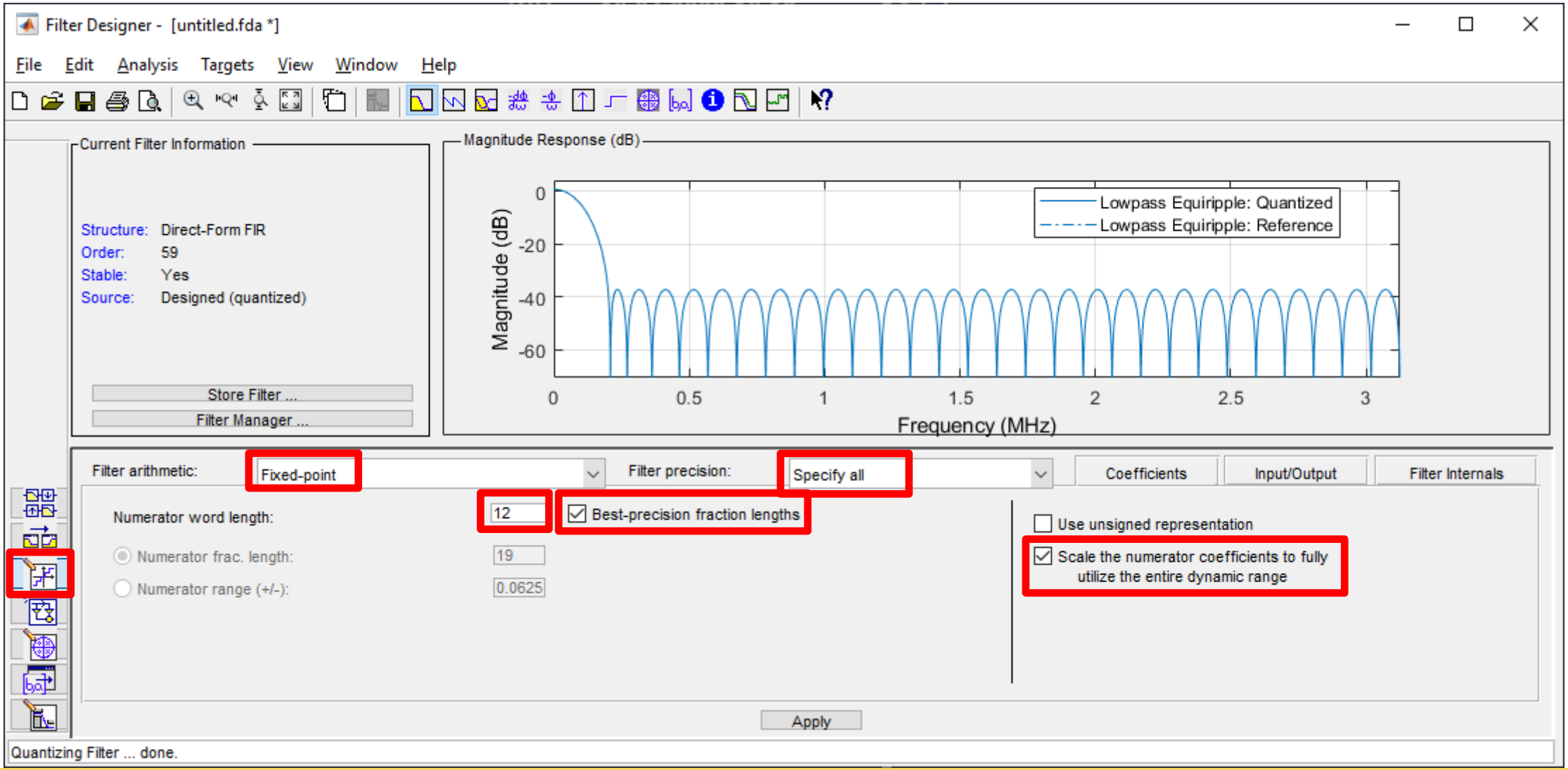
Matlab: Filter Design



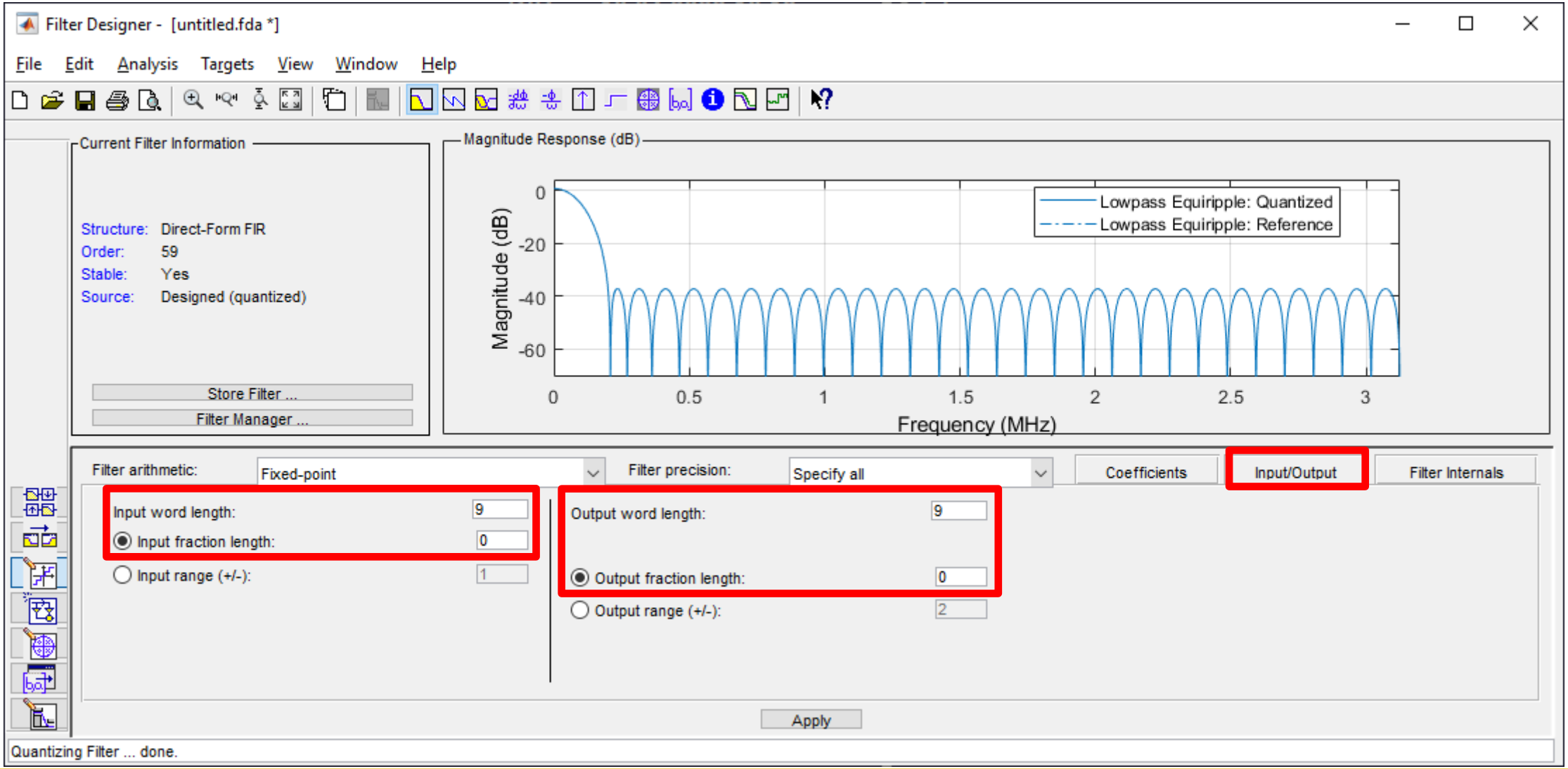
Matlab: Filter Design



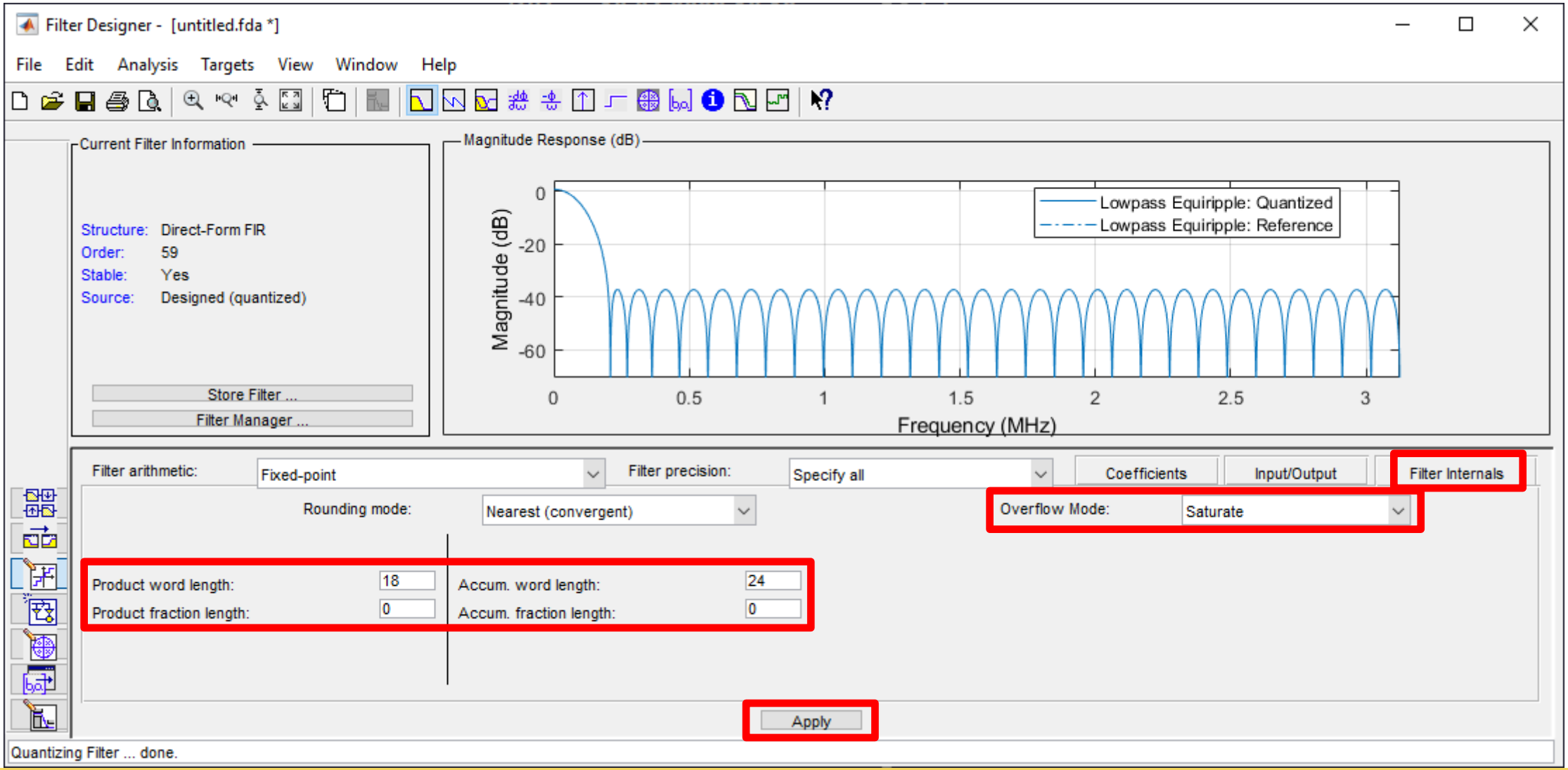
Matlab: Filter Design



Matlab: Filter Design



Matlab: Filter Design

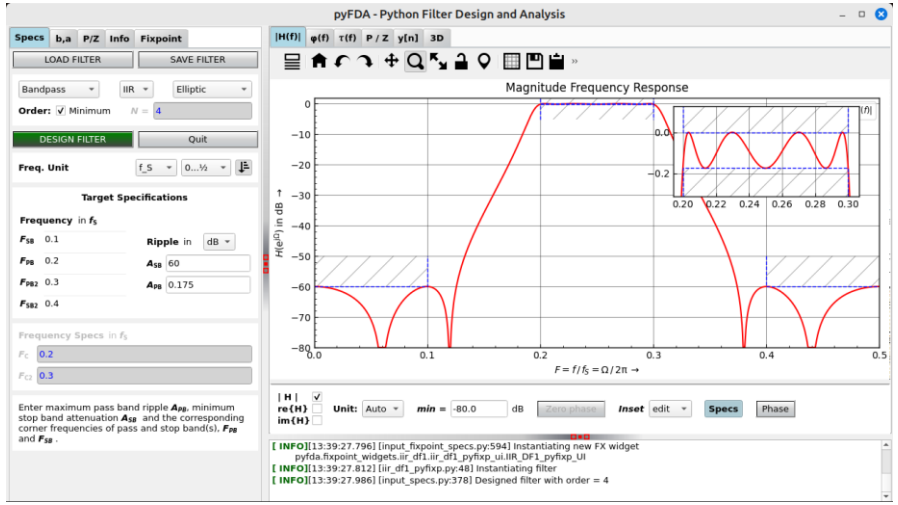


Matlab: Filter Design

- Export as TXT or CSV file.
- Modify the file into the following format:
  - c0, c1, c2, …
- If you have no Matlab, you can find the exported file in w03_filter_template/sources/matlab/FIR_50k.fcf

Alternative Tools: Filter Design

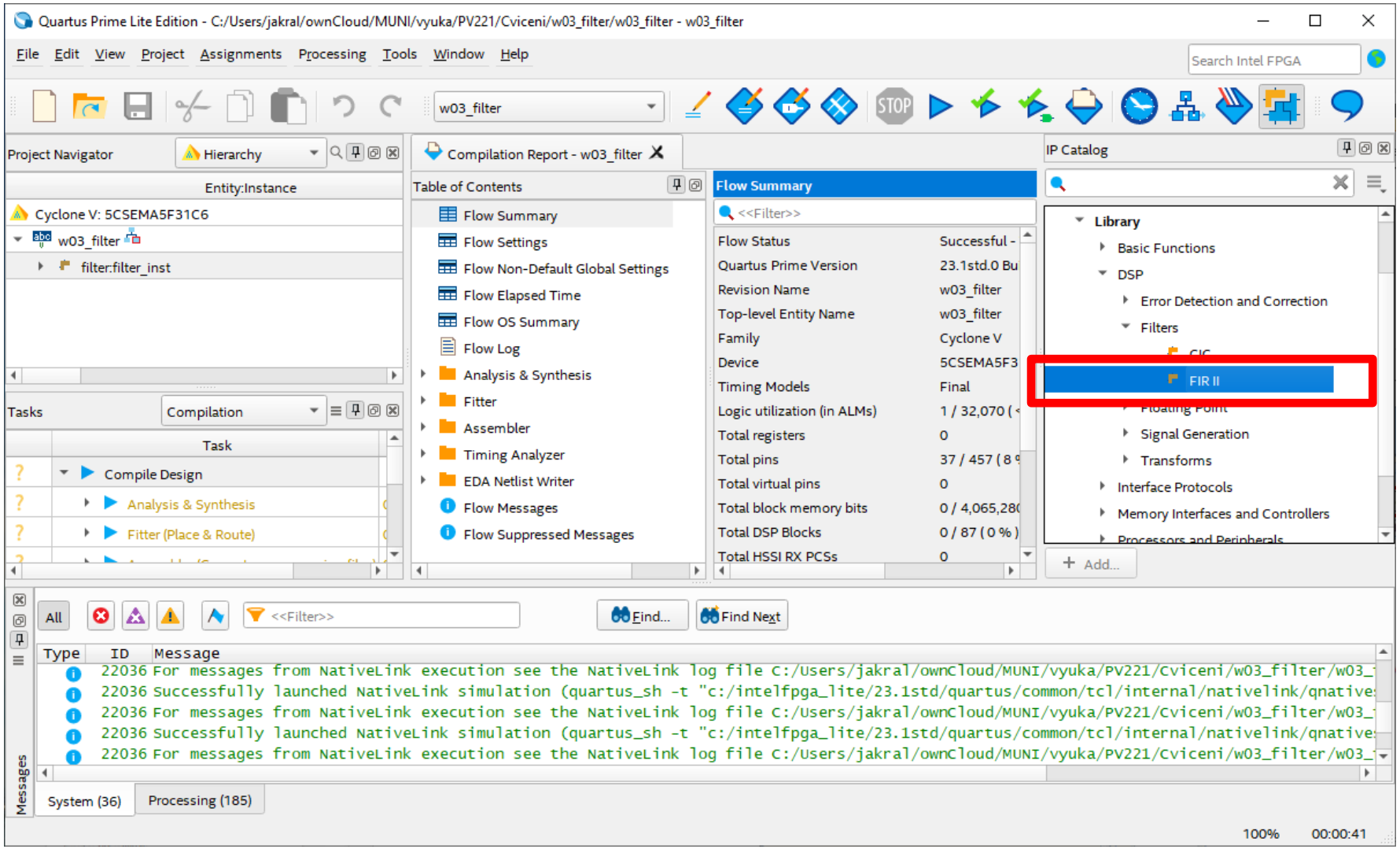
- pyfda (https://github.com/chipmuenk/pyfda)
- T-Filter – online
- (http://t-filter.engineerjs.com/)
- FIR Filter Designer - online (https://wirelesslibrary.labs.b-com.com/FIRfilterdesigner/#/)
- GNU Octave (https://www.allaboutcircuits.com/technical-articles/design-of-fir-filters-design-octave-matlab/)
- rePhase (https://www.minidsp.com/applications/advanced-tools/rephase-fir-tool)



FIR II IP Core

Quartus IP Core Library

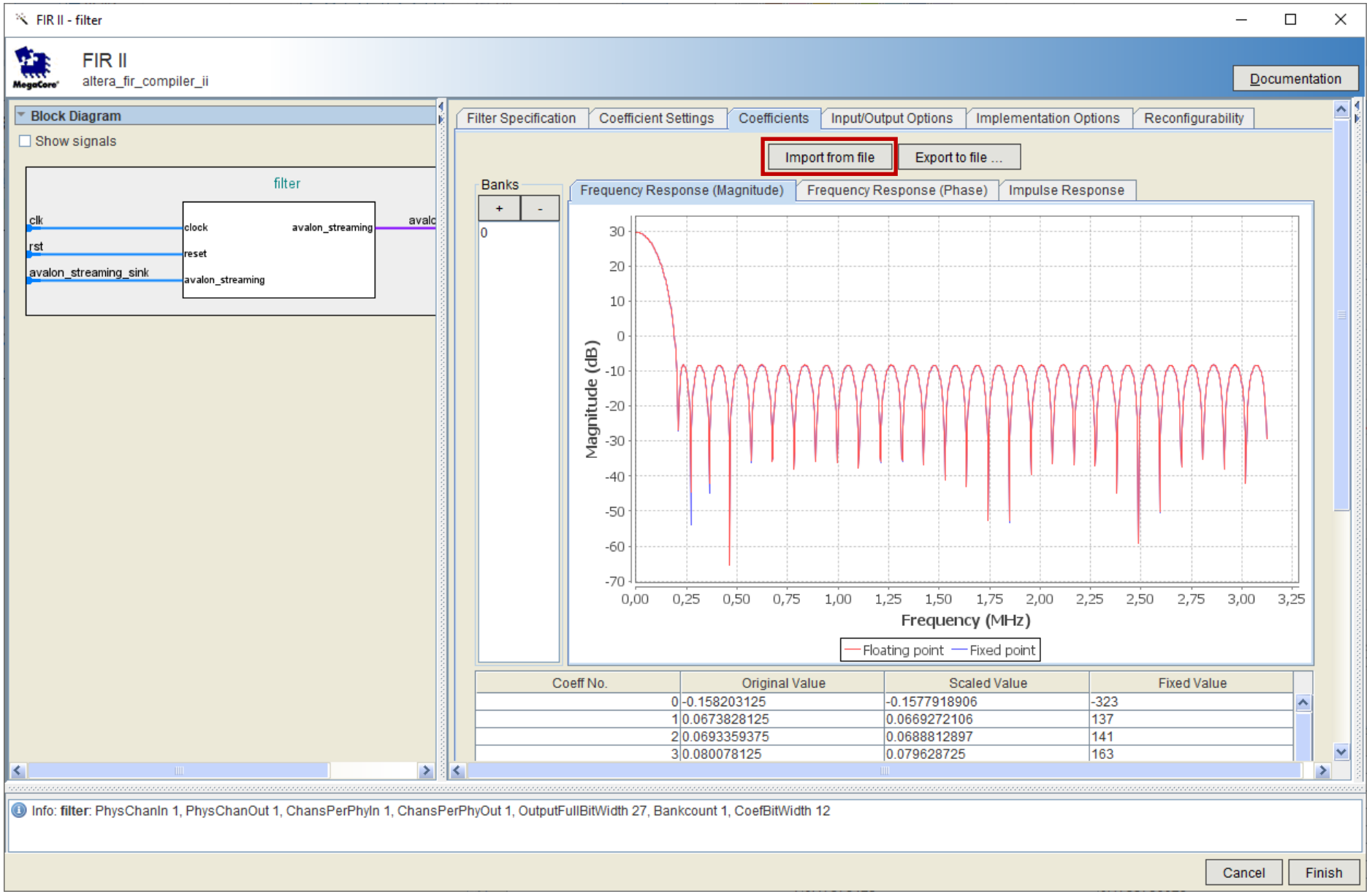
IP Core


Quartus Prime Lite Edition - C:/Users/jakral/ownCloud/MUNI/vyuka/PV221/Cviceni/w03_filter/w03_filter - w03_filter
File Edit View Project Assignments Processing Tools Window Help
Search Intel FPGA
w03_filter
Project Navigator
Hierarchy
Entity:Instance
Cyclone V: 5CSEMA5F31C6
w03_filter
filter:filter_inst
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Compilation Report - w03_filter
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Fitter
Assembler
Timing Analyzer
EDA Netlist Writer
Flow Messages
Flow Suppressed Messages
Flow Summary
<<Filter>>
Flow Status Successful -
Quartus Prime Version 23.1std.0 Bu
Revision Name w03_filter
Top-level Entity Name w03_filter
Family Cyclone V
Device 5CSEMA5F3
Timing Models Final
Logic utilization (in ALMs) 1 / 32,070 (
Total registers 0
Total pins 37 / 457 (8
Total virtual pins 0
Total block memory bits 0 / 4,065,28
Total DSP Blocks 0 / 87 (0 %)
Total HSSI RX PCSs 0
IP Catalog
Library
Basic Functions
DSP
Error Detection and Correction
Filters
FIR II
Signal Generation
Transforms
Interface Protocols
Memory Interfaces and Controllers
Add...
All
<<Filter>>
Find...
Find Next
Type ID Message
22036 For messages from NativeLink execution see the NativeLink log file C:/Users/jakral/ownCloud/MUNI/vyuka/PV221/Cviceni/w03_filter/w03_
22036 Successfully launched NativeLink simulation (quartus_sh -t "c:/intelfpga_lite/23.1std/quartus/common/tcl/internal/nativelink/qnative
22036 For messages from NativeLink execution see the NativeLink log file C:/Users/jakral/ownCloud/MUNI/vyuka/PV221/Cviceni/w03_filter/w03_
22036 Successfully launched NativeLink simulation (quartus_sh -t "c:/intelfpga_lite/23.1std/quartus/common/tcl/internal/nativelink/qnative
22036 For messages from NativeLink execution see the NativeLink log file C:/Users/jakral/ownCloud/MUNI/vyuka/PV221/Cviceni/w03_filter/w03_
Messages
System (36)
Processing (185)
100%
00:00:41

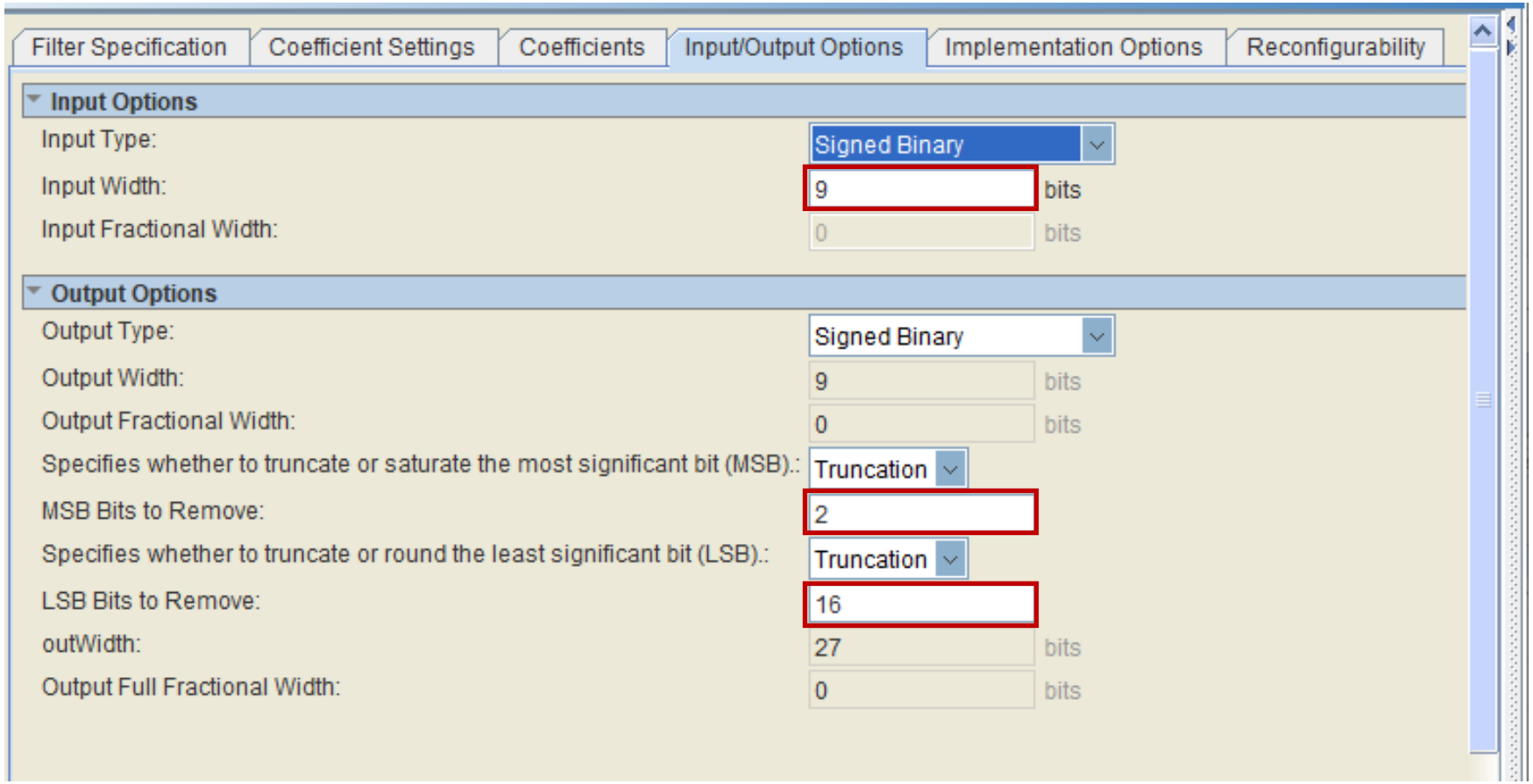
IP Core

Expected file format:

c0, c1, c2, …



IP Core


Filter Specification
Coefficient Settings
Coefficients
Input/Output Options
Implementation Options
Reconfigurability
Input Options
Input Type:
Signed Binary
Input Width:
9
bits
Input Fractional Width:
0
bits
Output Options
Output Type:
Signed Binary
Output Width:
9
bits
Output Fractional Width:
0
bits
Specifies whether to truncate or saturate the most significant bit (MSB).:
Truncation
MSB Bits to Remove:
2
Specifies whether to truncate or round the least significant bit (LSB).:
Truncation
LSB Bits to Remove:
16
outWidth:
27
bits
Output Full Fractional Width:
0
bits

IP Core

- Calculate number of expected DSP blocks:
  1) for non-symmetrical coefficients
  2) for symmetrical coefficients
- Verify the calculation on Implementation Options tab.

Verilog: Reading Data from a File

```
integer fid_din;
reg [256:0] line; // buffer for line read from file

initial begin
  // open the file
  fid_din = $fopen("../../sources/data/FIR_din.txt", "r");
  if (fid_din == 0) begin
    $display("ERROR: File not found.");
    $finish;
  end

  // check FEOF and try to read a line from the file
  while (!$feof(fid_din) && $fgets(line, fid_din) != 0) begin
    if ($sscanf(line, "%d", data) != 1) begin
      $display("ERROR: Failed to scan number from line.");
      $finish;
    end
  end
  $fclose(fid_din);
end
```

Verilog: Writing Data to a File

```
integer fid_dout_act;

initial begin
  // open the file for writing
  fid_dout_act = $fopen("../../sources/data/FIR_dout_act.txt", "w");
  if (fid_dout_act == 0) begin
    $display("ERROR: File could not be opened.");
    $finish;
  end

  while (!sim_finished) begin
    // check if the data is valid
    if (dv) begin
      // write the data to file
      $fdisplay(fid_dout_act, "%d", data);    // internally adds new line
    end
    #(CLK_PER); // wait for one clock period
  end

  $fclose(fid_dout_act); // close the file
end
```