Overview of Matlab Design Process

Mitch D'Ewart

July 28, 2016

U.S. DEPARTMENT OF ENERGY
Office of Science

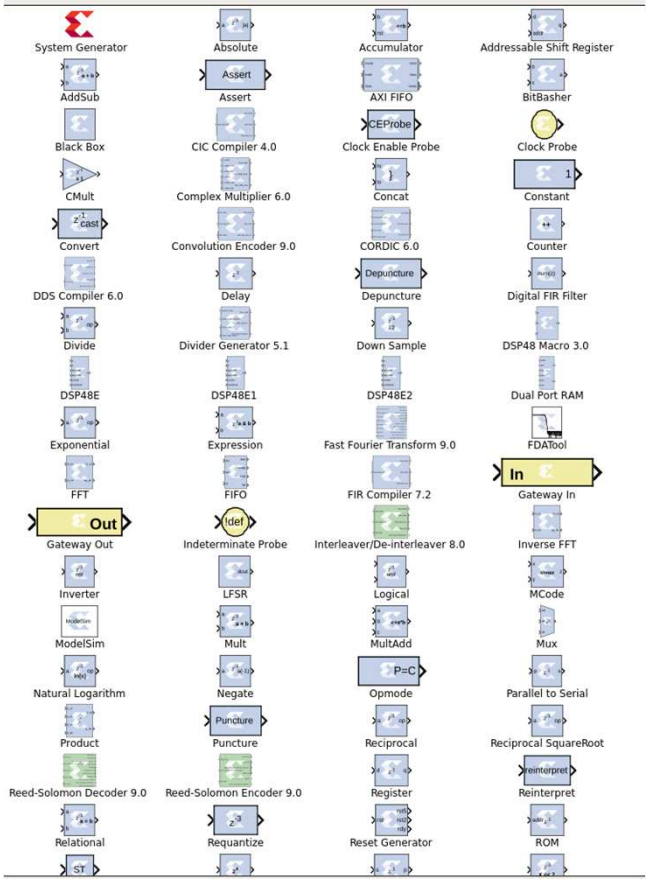
SLAC NATIONAL ACCELERATOR LABORATORY

Outline

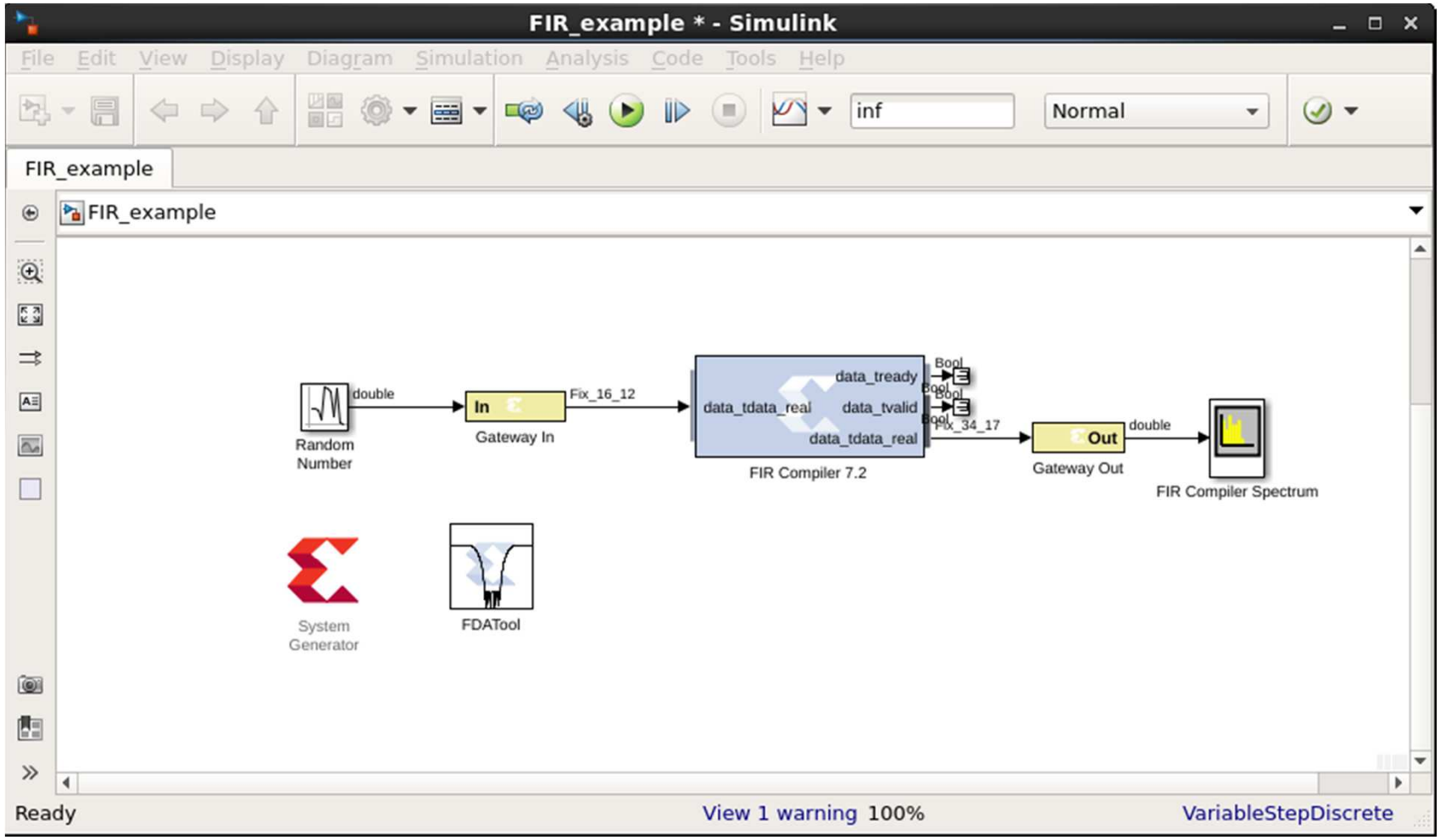
- System Generator Introduction
  - What is System Generator
  - Blocks for rapid development
  - Example
- Interface to Common Platform
- Model Based Design Flow
  - Overview
  - Examples

System Generator

- DSP design tool from Xilinx
- Uses MathWorks model based Simulink design environment
  - Xilinx specific blockset
- 90+ DSP building blocks
  - Adders
  - Multiplies
  - Registers
- No previous experience with RTL/Xilinx design required

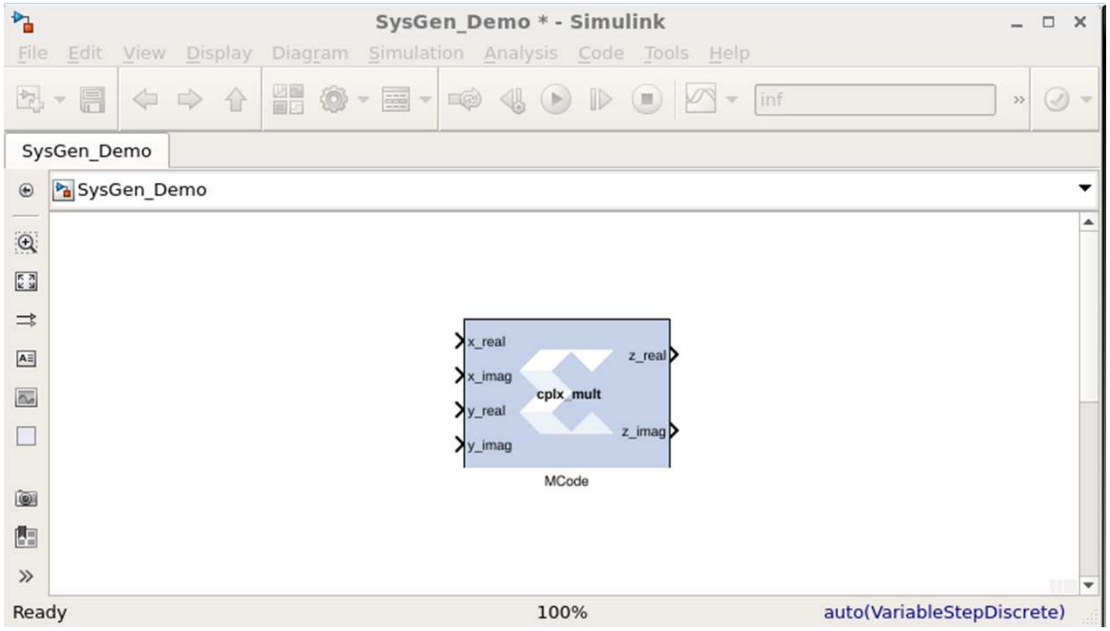


FIR Filter Generation



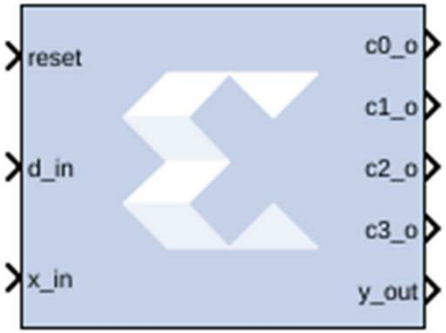
Matlab MCode Integration

```
function [z_real, z_imag] = cplx_mult(x_real, x_imag, y_real, y_imag)
    z_real = x_real * y_real - x_imag * y_imag;
    z_imag = x_real * y_imag + x_imag * y_real;
end
```



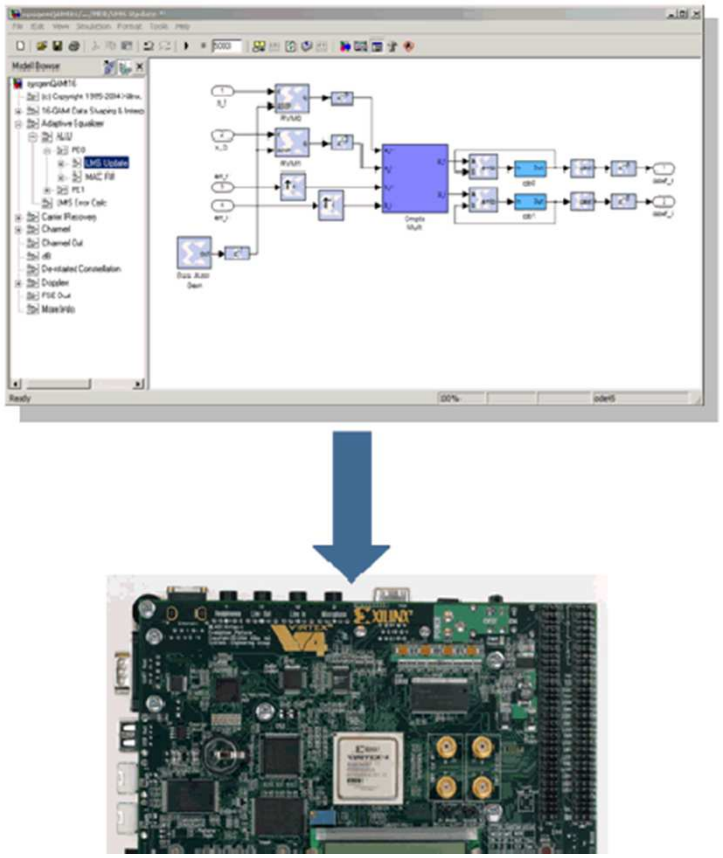
Blackbox HDL Integration

```
-- ------------------------------------------------------
ENTITY farrow IS  ------> Interface
GENERIC (IL : INTEGER := 3); -- Input buffer length -1
PORT ( clk                    : IN STD_LOGIC; -- System clock
       ce                     : IN STD_LOGIC;
       reset                  : IN STD_LOGIC; -- Asynchronous reset
       x_in                   : IN S8;        -- System input
       d_in                   : IN U8;        -- Delay input
       c0_o, c1_o, c2_o, c3_o : OUT S9;       -- Phase delays
       y_out                  : OUT S9);      -- System output
END farrow;
-- ------------------------------------------------------
```



Hardware Cosimulation

- Mixes hardware and software simulation
  - Automated generation of FPGA bitstream from Simulink
  - Generates simulink block representing FPGA simulation
  - Speed up simulation
  - Built in support for evaluation boards
  - Also possible to specify custom compilation target

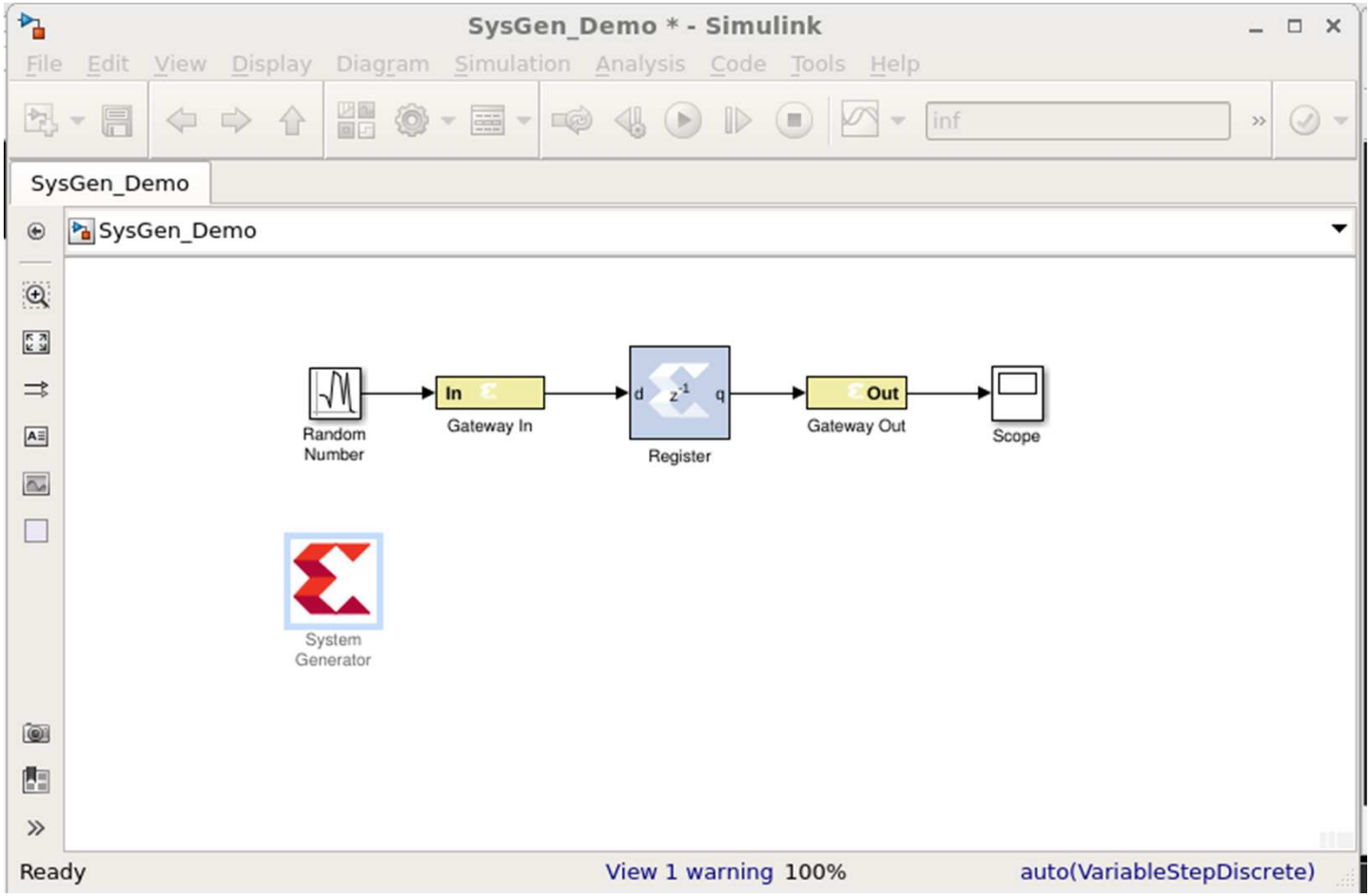


Output Products

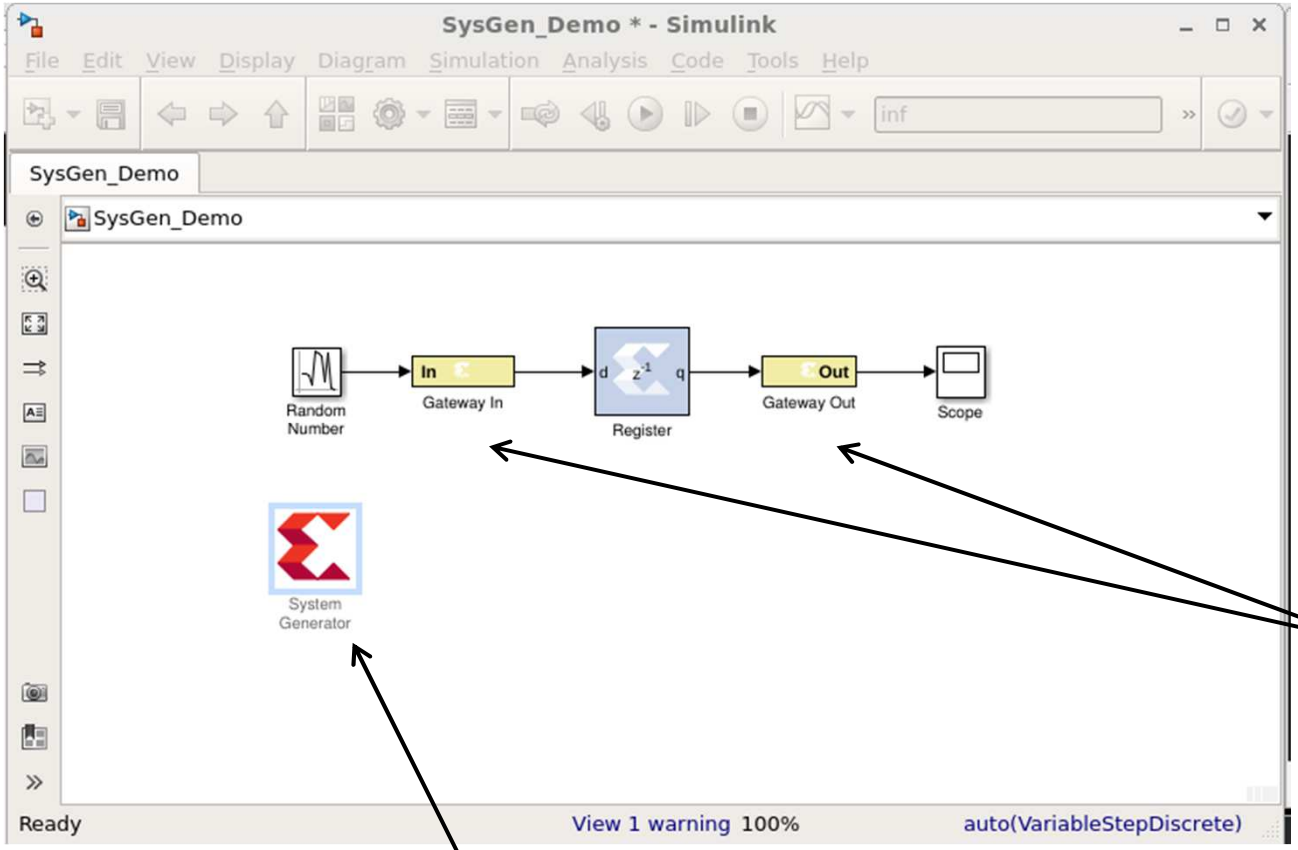
- HDL Netlist
- Various hardware co-simulation
- IP Catalog
  - IP core that can be added to vivado
- Synthesized Checkpoint*
  - Checkpoint file that can be used in any Vivado IDE project

*Synthesized checkpoint is used to integrate with the common platform firmware

Example


SysGen_Demo * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
SysGen_Demo
SysGen_Demo
Random Number
In
Gateway In
d
z-1
q
Register
Out
Gateway Out
Scope
System Generator
Ready
View 1 warning 100%
auto(VariableStepDiscrete)

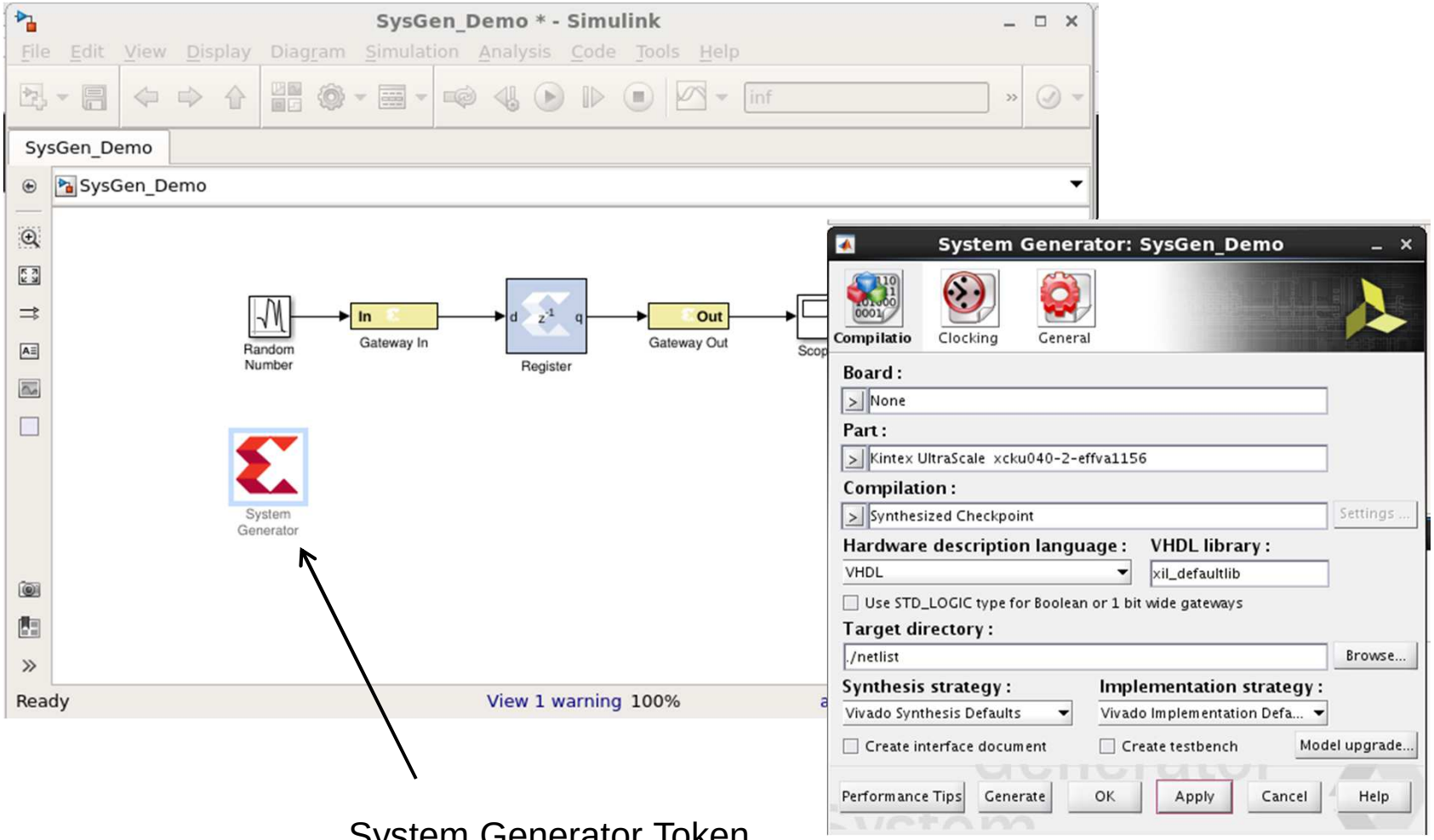
Elements of System Generator Model



Gateway In/Out

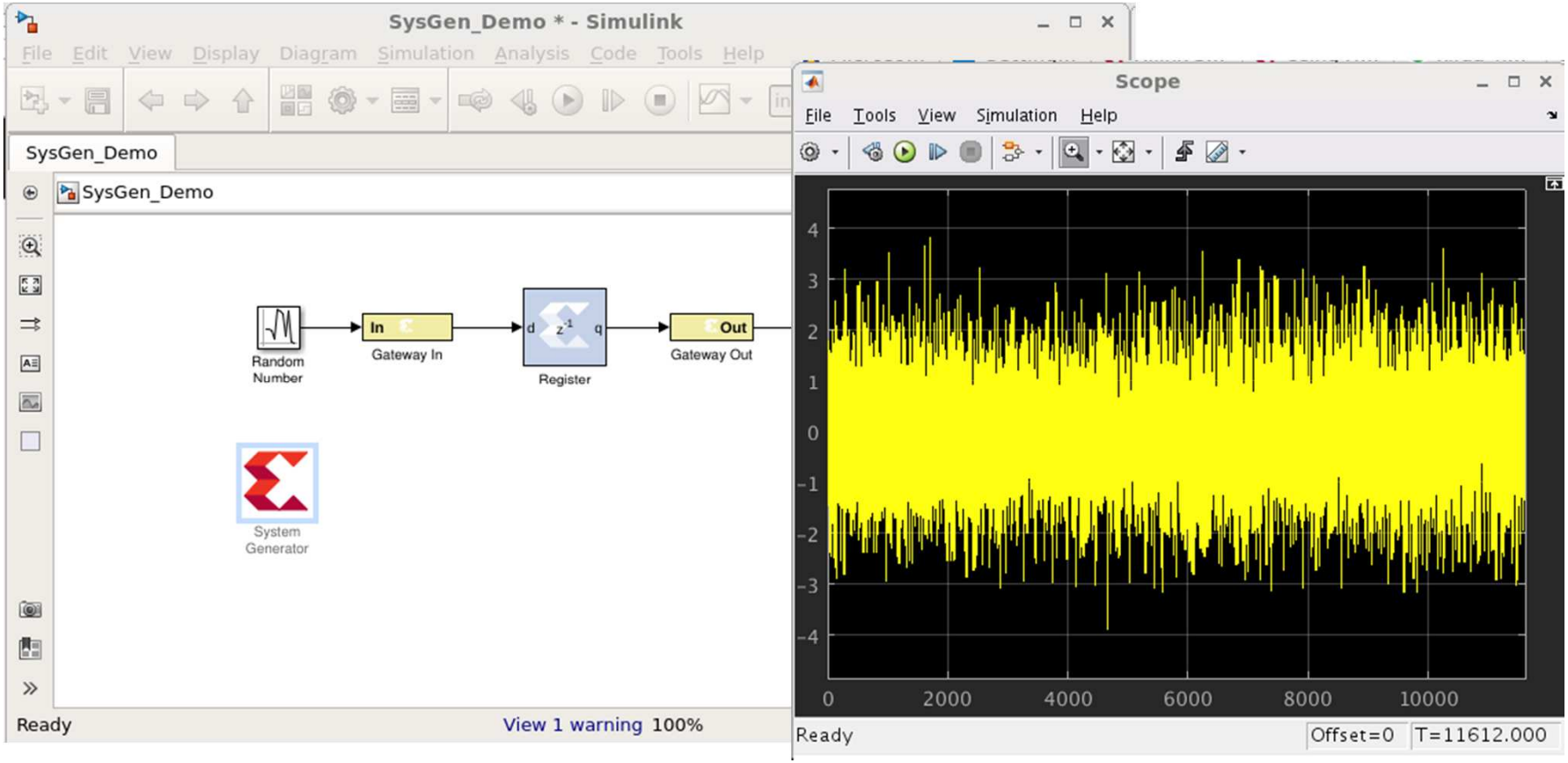
System Generator Token

System Generator Token

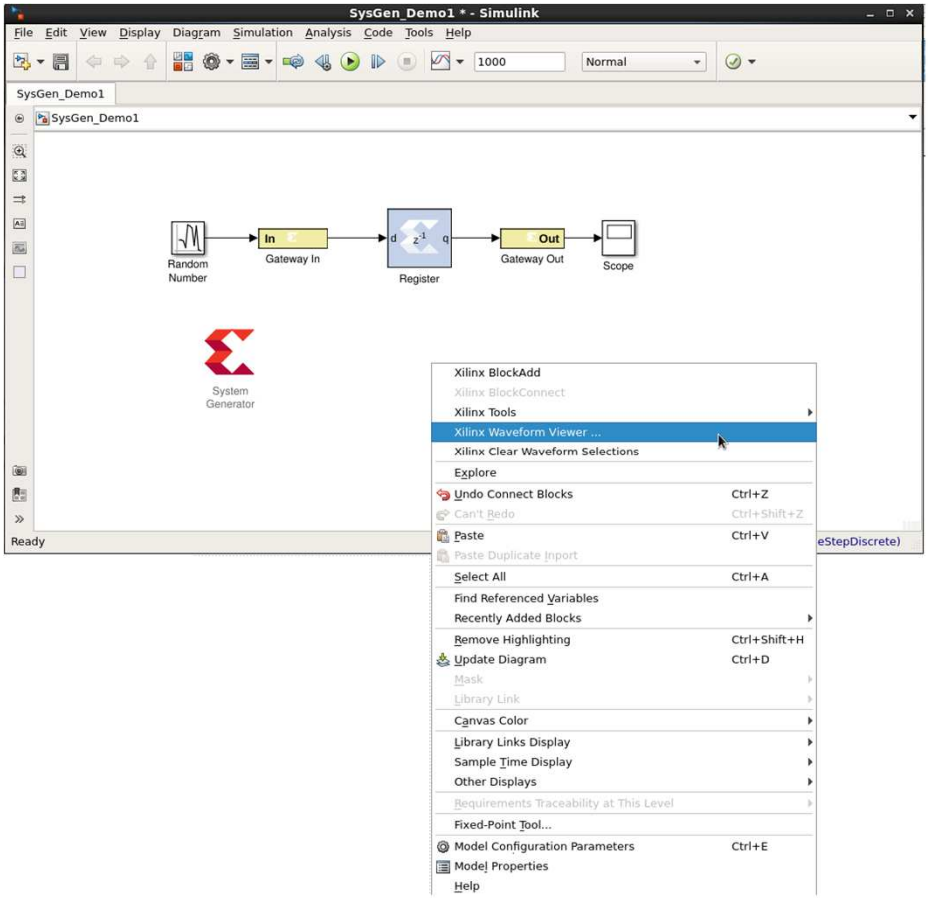


System Generator Token

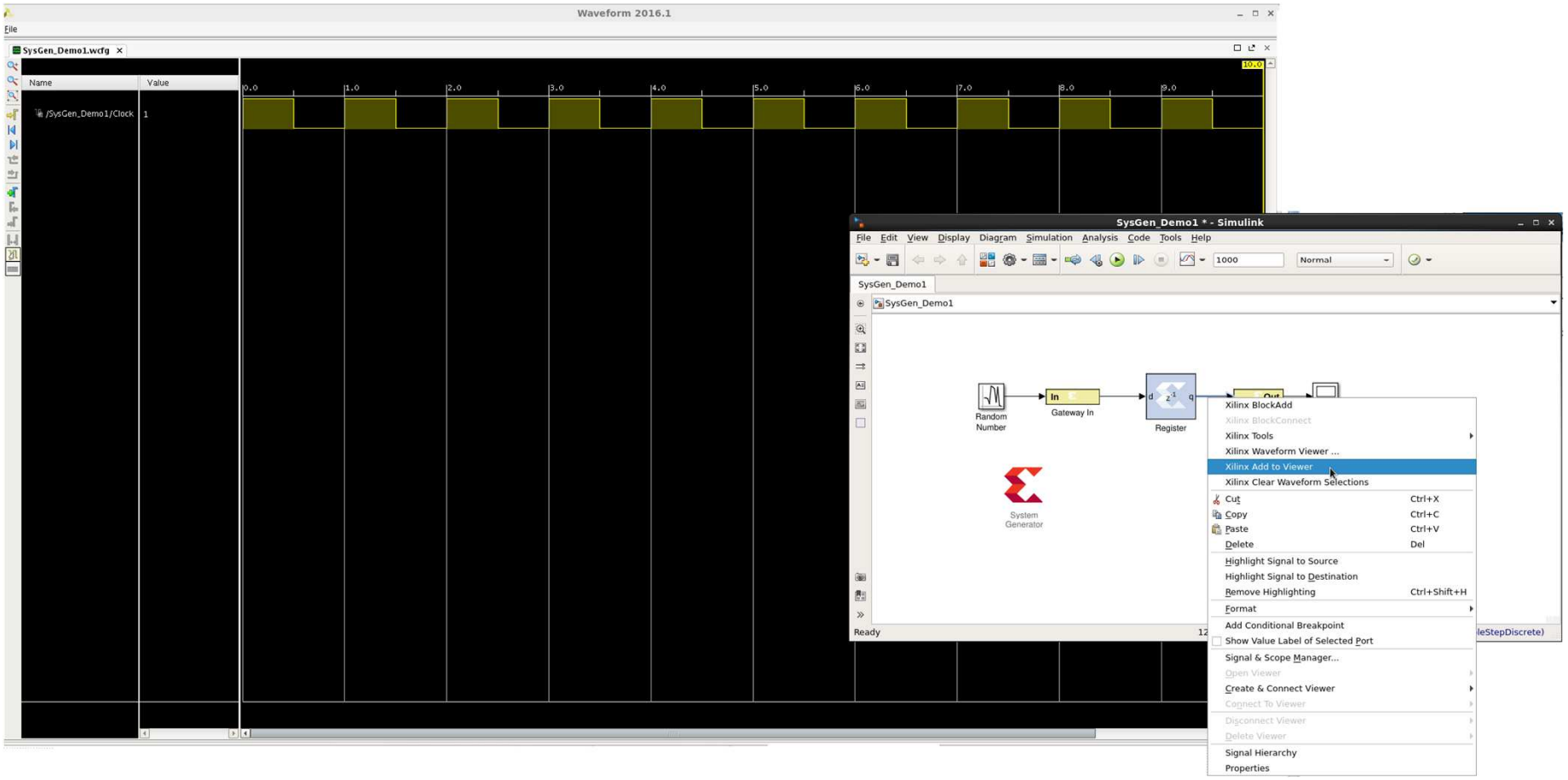
First Example



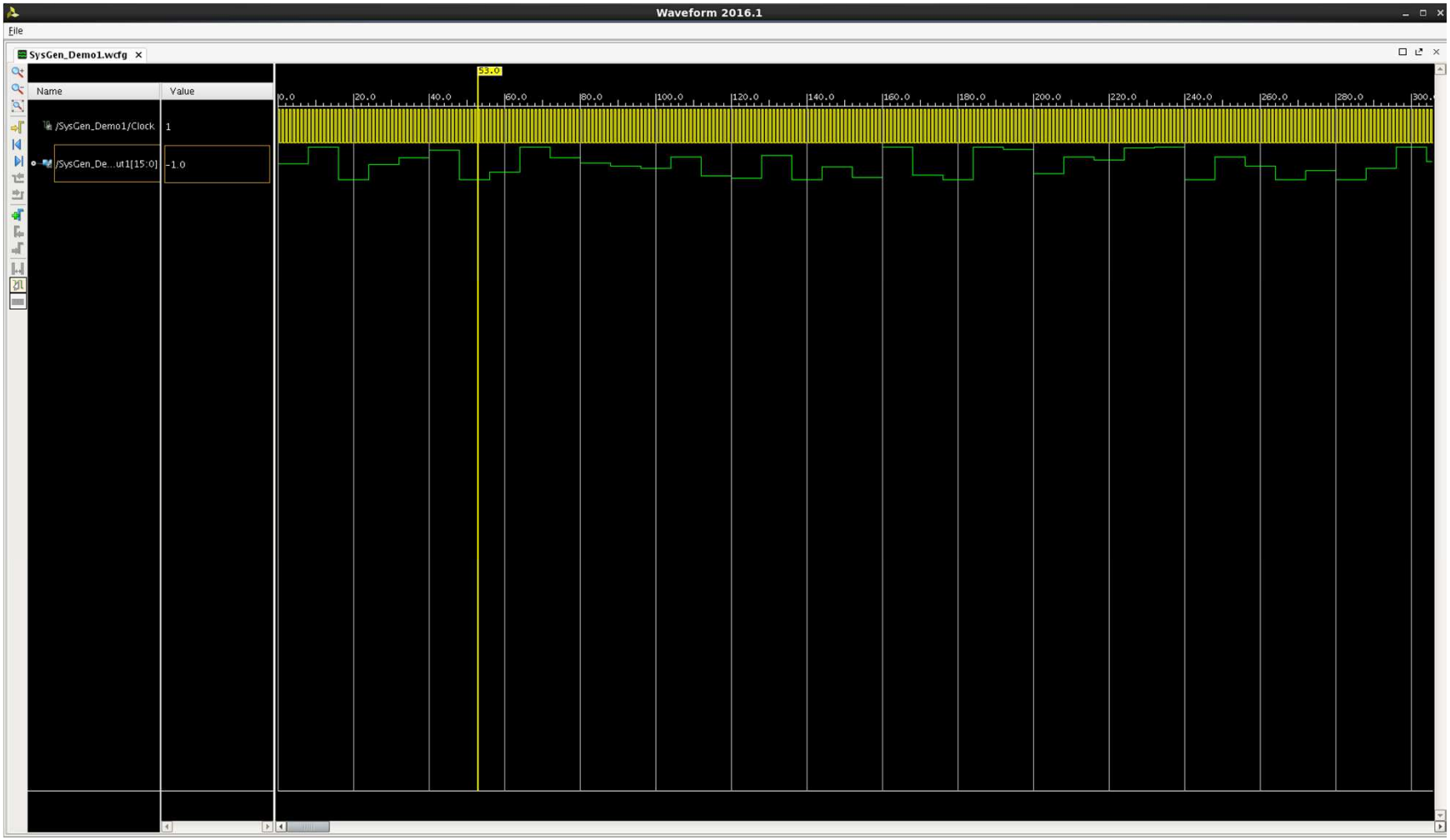
Useful Tools – Xilinx Waveform Viewer



Useful Tools – Xilinx Waveform Viewer

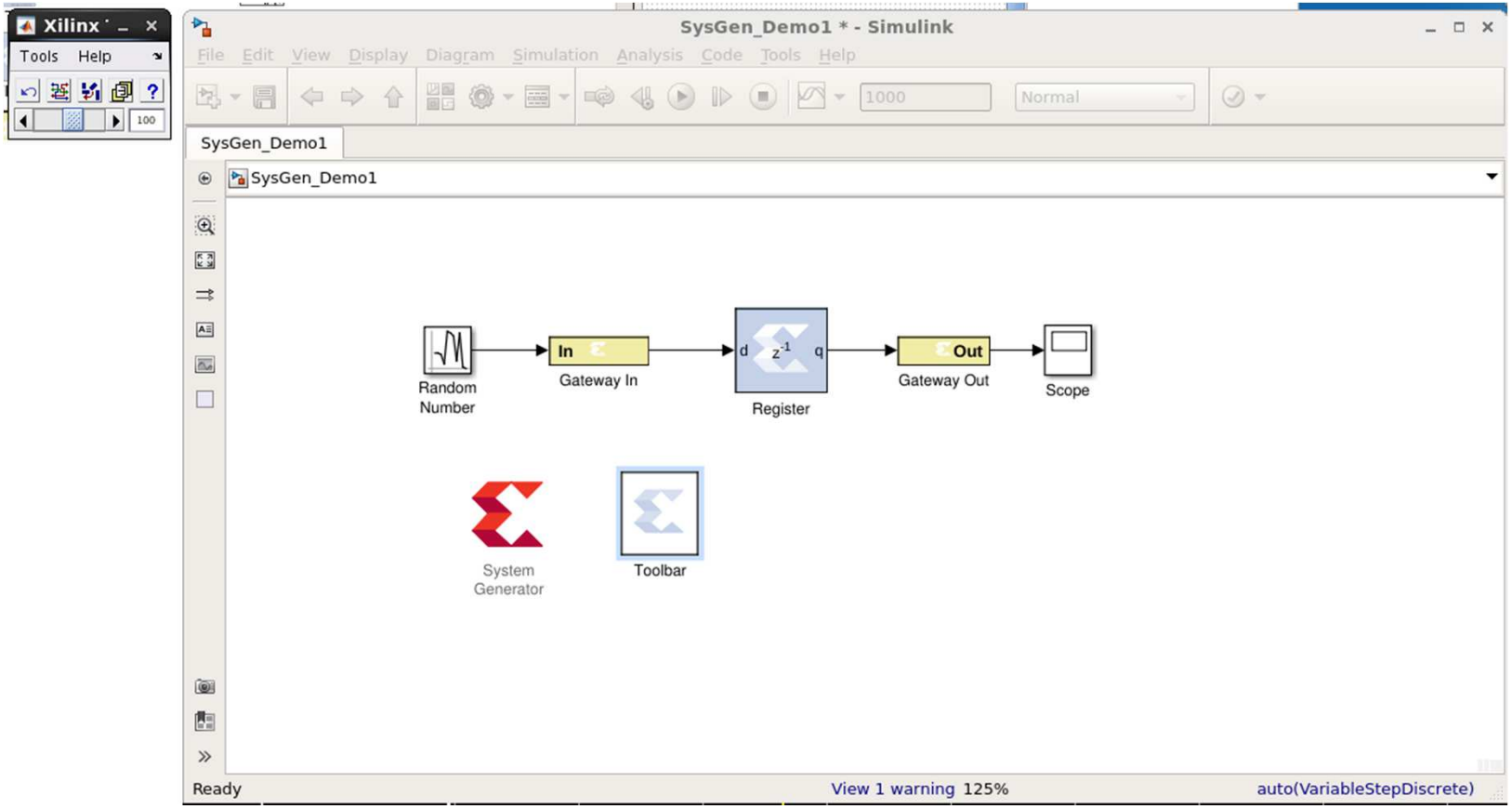


Useful Tools – Xilinx Waveform Viewer



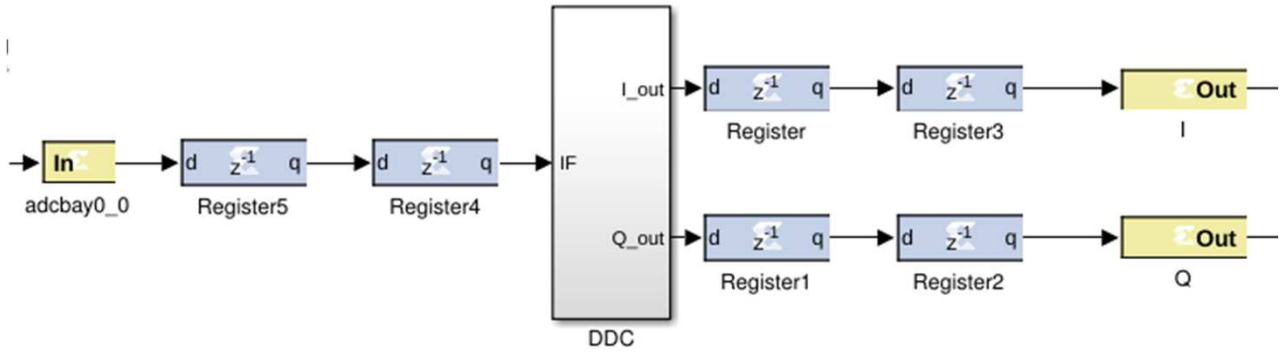
Useful Tools – Xilinx Toolbar

- Undo, reroute, auto layout, terminate, zoom, help

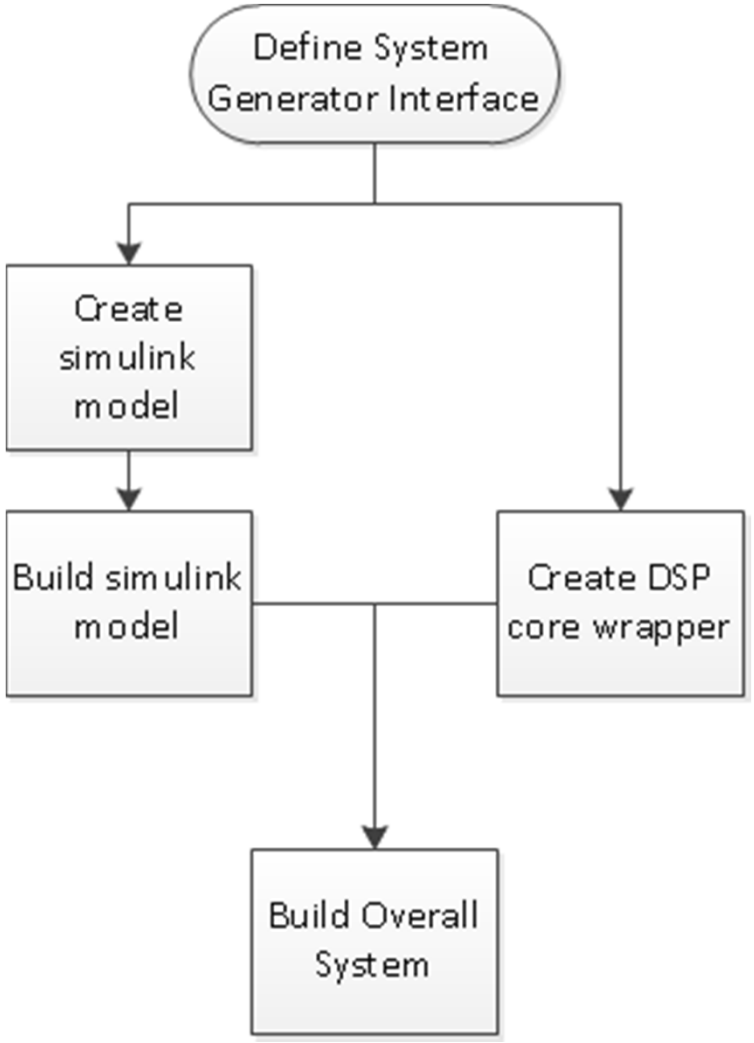


Notes for Higher Performance Design

- Register inputs and outputs of design
- Insert pipeline registers
- Use saturation arithmetic and rounding only when necessary
- Set the data rate option on all gateway blocks
- Know about the target FPGA



Interface to Common Platform



Interface to Common Platform

Create new DSP core module

1. Define interface to system generator
   - ADC/DAC, debug stream, AXI4-Lite interface, etc.
2. System generator interface
   - Define input/output gateways with compatible interface
     - Gateway names, bit widths, types
     - Define scratch pad register
       - Manually assign address to max AXI4-Lite address size
       - Set scratch pad address to 0xFFC for a 12-bit address space
3. Application core interface
   - Create system generator wrapper/interface
   - Maps desired inputs/outputs to defined system generator interface
   - Synchronize clocks if needed

Example - Create New DSP Core Module

- New module DspCoreLib/AdcToDacLoopBackCore

```
[mdewart@tid-pc92646 DspCoreLib]$ ls
AdcToDacLoopBackCore/  LICENSE.txt

[mdewart@tid-pc92646 AdcToDacLoopBackCore]$ ls
cores.txt  rtl/  sim.txt  simulink/  sources.txt
```

- rtl
  - system generator wrapper to interface to overall system
- Simulink
  - simulink (system generator) model
  - ouput products
    - netlist/dspcore_stub.vhd
    - netlist/dspcore.dcp

Interface to Common Platform

Create new DSP core module

1. Define interface to system generator
   - ADC/DAC, debug stream, AXI4-Lite interface, etc.
2. System generator interface
   - Define input/output gateways with compatible interface
     - Gateway names, bit widths, types
     - Define scratch pad register
       - Manually assign address to max AXI4-Lite address size
       - Set scratch pad address to 0xFFC for a 12-bit address space
3. Application core interface
   - Create system generator wrapper/interface
   - Maps desired inputs/outputs to defined system generator interface
   - Synchronize clocks if needed

Example - Define Interface to System Generator

- ADC to DAC loopback test interface:
  - ADC/DAC raw stream
    - 16-bit signed data
      - toDsp_adcData<0-5>
      - fromDsp_dacData<0-1>
      - fromDsp_dacbay1_1
  - Debug raw stream
    - fromDsp_debug<0-1>
  - 12-bit AXI4-Lite register space

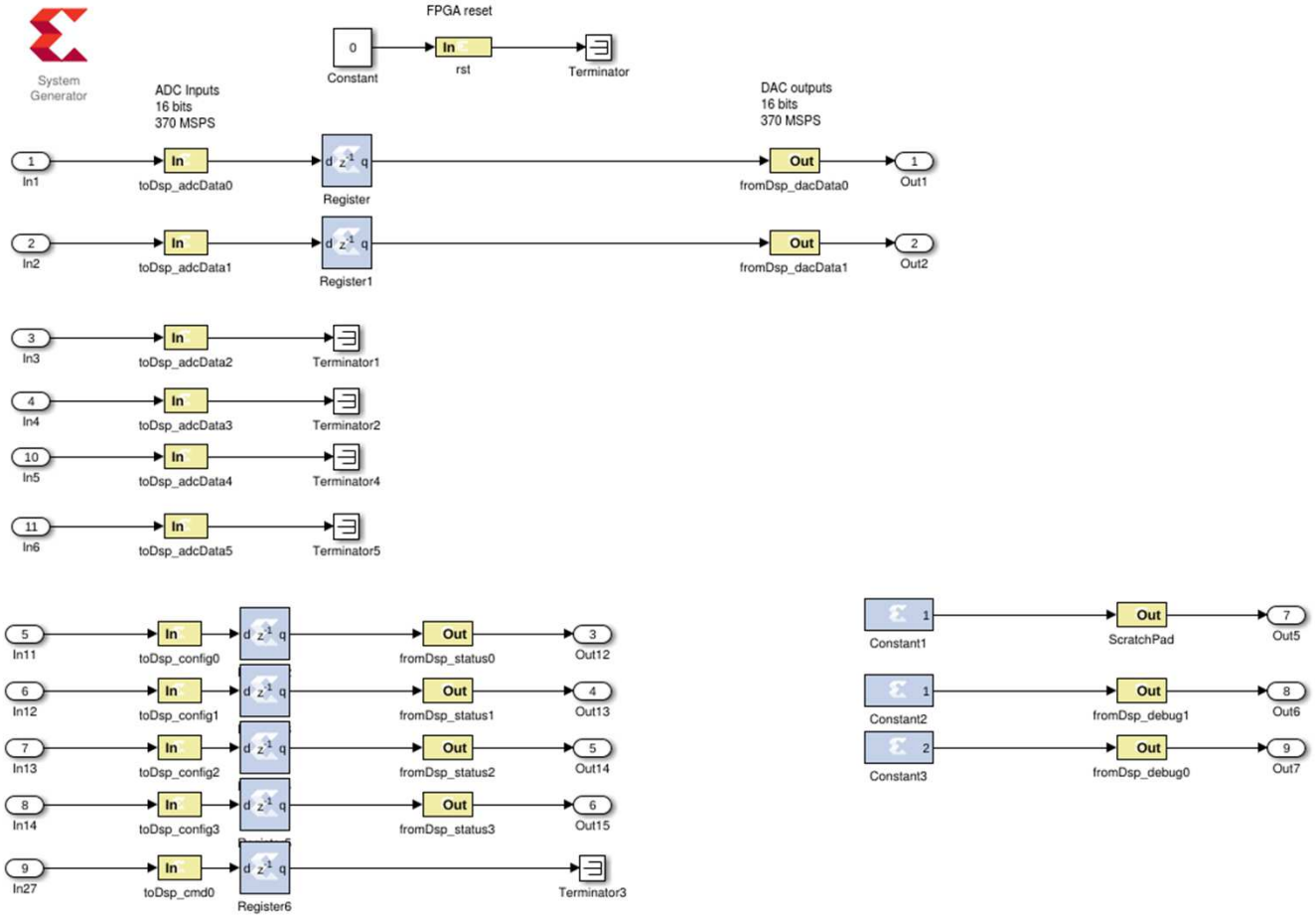
Interface to Common Platform

Create new DSP core module

1. Define interface to system generator
   - ADC/DAC, debug stream, AXI4-Lite interface, etc.
2. System generator interface
   - Define input/output gateways with compatible interface
     - Gateway names, bit widths, types
     - Define scratch pad register
       - Manually assign address to max AXI4-Lite address size
       - Set scratch pad address to 0xFFC for a 12-bit address space
3. Application core interface
   - Create system generator wrapper/interface
   - Maps desired inputs/outputs to defined system generator interface
   - Synchronize clocks if needed

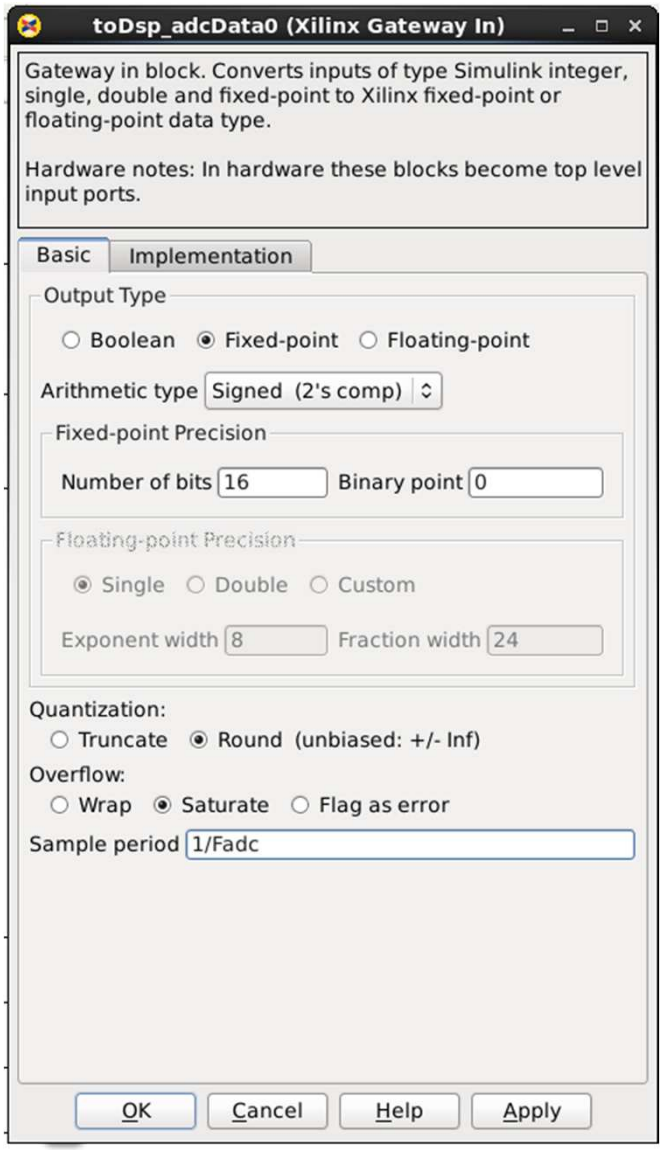
Example - System Generator Interface

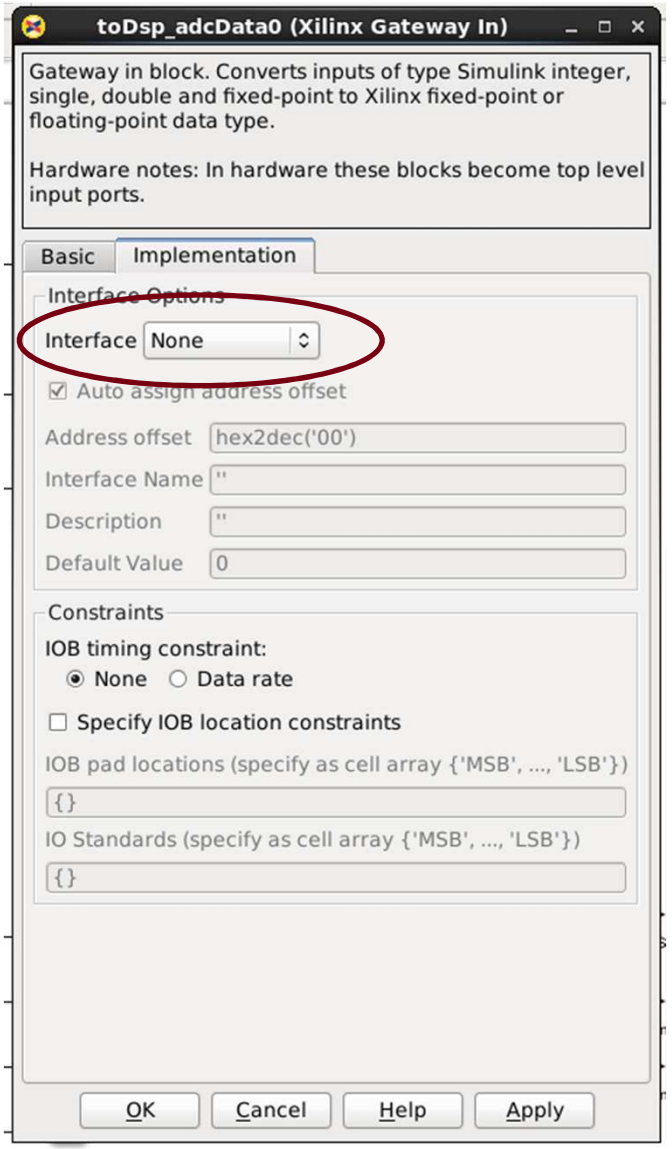
```
[mdewart@tid-pc92646 simulink]$ ls
ADCtoDACloopback.slx  netlist/
```



Example - System Generator Interface

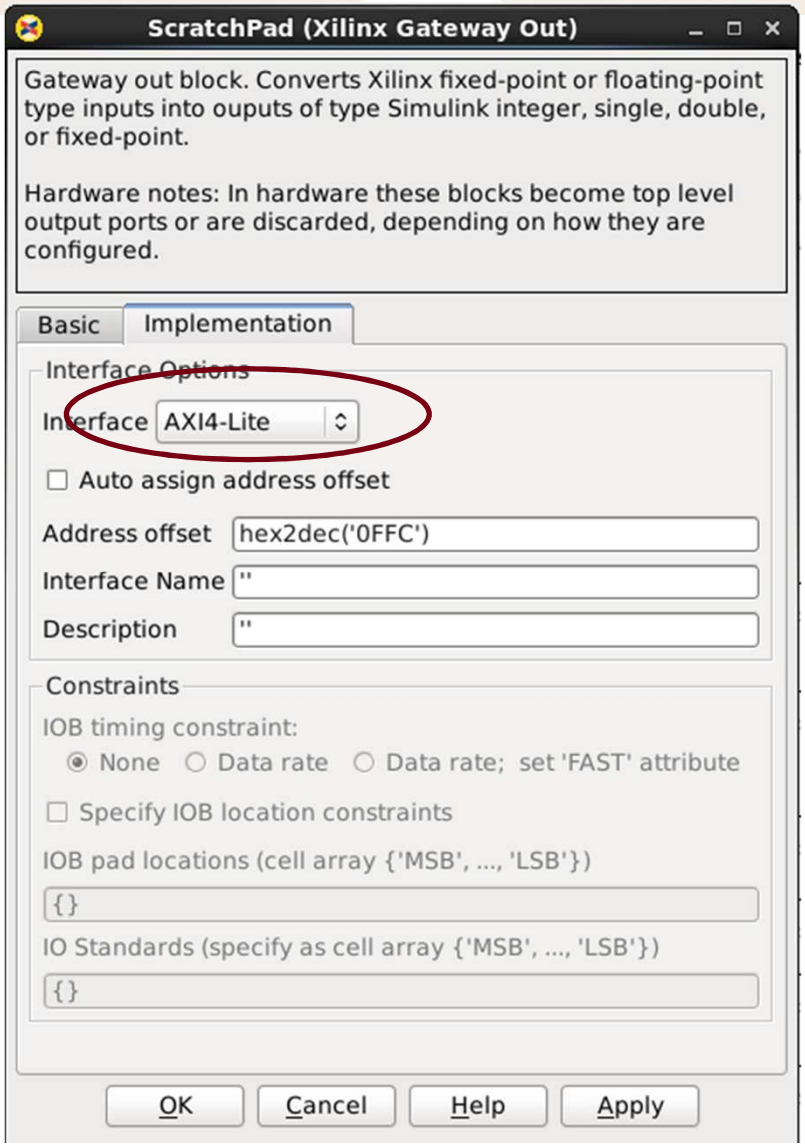
- Raw stream
  - ADC, DAC data
  - Diagnostic





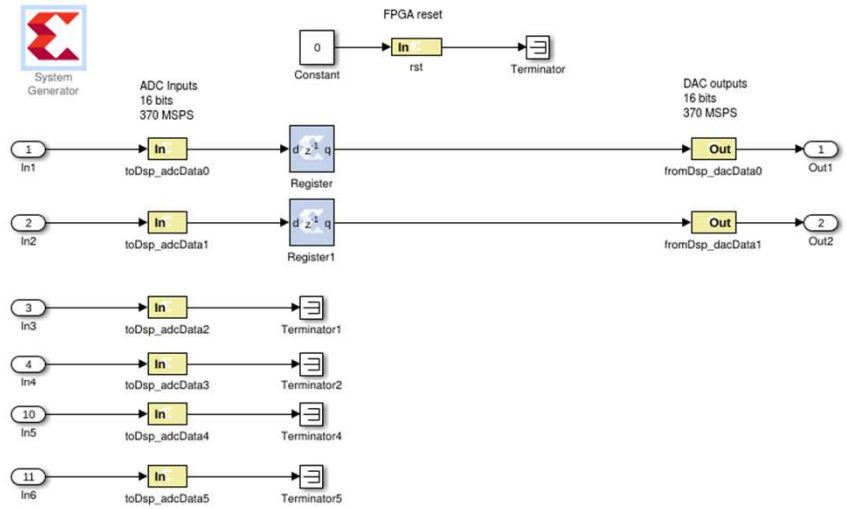
Example - System Generator Interface

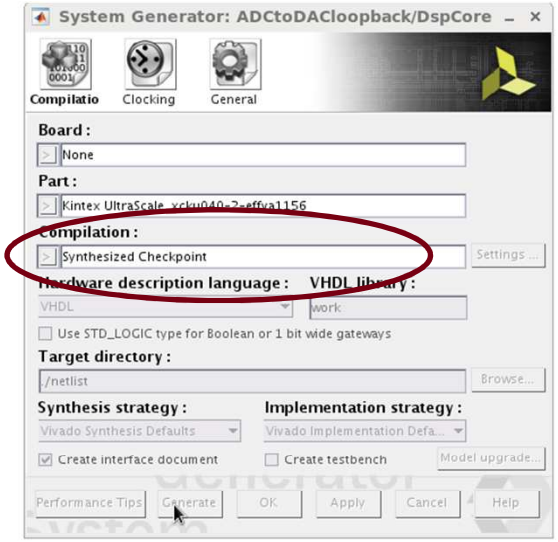
- AXI4-Lite slave interface
  - Status, configuration registers
    - Up to 32-bit width
    - User defined data type
  - Scratch pad
    - 0xFFC will ensure 12-bit address space

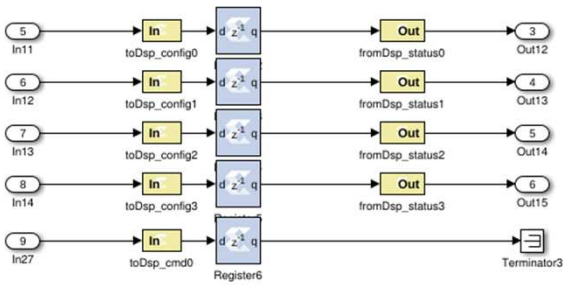


Example - System Generator Interface

- Generate core







Example - Output dspcore_stub.vhd

```
-- Generated from Simulink block
library IEEE;
use IEEE.std_logic_1164.all;
library work;
entity dspcore_stub is
  port (
    rst : in std_logic_vector( 1-1 downto 0 );
    todsp_adcdata0 : in std_logic_vector( 16-1 downto 0 );
    todsp_adcdata1 : in std_logic_vector( 16-1 downto 0 );
    todsp_adcdata2 : in std_logic_vector( 16-1 downto 0 );
    todsp_adcdata3 : in std_logic_vector( 16-1 downto 0 );
    todsp_adcdata4 : in std_logic_vector( 16-1 downto 0 );
    todsp_adcdata5 : in std_logic_vector( 16-1 downto 0 );
    clk : in std_logic;
    dspcore_aresetn : in std_logic;
    dspcore_s_axi_awaddr : in std_logic_vector( 12-1 downto 0 );
    dspcore_s_axi_awvalid : in std_logic;
    dspcore_s_axi_wdata : in std_logic_vector( 32-1 downto 0 );
    dspcore_s_axi_wstrb : in std_logic_vector( 4-1 downto 0 );
    dspcore_s_axi_wvalid : in std_logic;
    dspcore_s_axi_bready : in std_logic;
    dspcore_s_axi_araddr : in std_logic_vector( 12-1 downto 0 );
    dspcore_s_axi_arvalid : in std_logic;
```

ADC, raw stream

AXI4-Lite interface

Interface to Common Platform

Create new DSP core module

1. Define interface to system generator
   - ADC/DAC, debug stream, AXI4-Lite interface, etc.
2. System generator interface
   - Define input/output gateways with compatible interface
     - Gateway names, bit widths, types
     - Define scratch pad register
       - Manually assign address to max AXI4-Lite address size
       - Set scratch pad address to 0xFFC for a 12-bit address space
3. Application core interface
   - Create system generator wrapper/interface
   - Maps desired inputs/outputs to defined system generator interface
   - Synchronize clocks if needed

Example - Application Core Interface

SLAC

```
[mdewart@tid-pc92646 rtl]$ ls
AdcToDacLoopbackWrapper.vhd
```

```
   -----------
   -- DSP Core
   -----------
   U_DspCore : entity work.DspCore
      port map (
         -- Clock and Resets
         clk                   => dspClk,
         rst(0)                => dspRst,
         dspcore_aresetn       => dspRstL,
         -- ADC Channels
         todsp_adcdata0        => adc(0),
         todsp_adcdata1        => adc(1),
         todsp_adcdata2        => adc(2),
         todsp_adcdata3        => adc(3),
         -- DAC Channels
         fromdsp_dacdata0      => dac(0),
         fromdsp_dacdata1      => dac(1),
         fromdsp_dacdata2      => dac(2),
         fromdsp_dacdata3      => dac(3),
         -- AXI-Lite Interface
         dspcore_s_axi_awaddr  => writeMaster.awaddr(ADDR_WIDTH_C-1 downto 0),
         dspcore_s_axi_awvalid => writeMaster.awvalid,
         dspcore_s_axi_wdata   => writeMaster.wdata,
         dspcore_s_axi_wstrb   => writeMaster.wstrb,
         dspcore_s_axi_wvalid  => writeMaster.wvalid,
         dspcore_s_axi_bready  => writeMaster.bready,
```

Example – Integrate into Overall System

- Include in overall build

```
[mdewart@tid-pc92646 AdcToDacLoopBackCore]$ cat sources.txt
vhdl work rtl/AdcToDacLoopbackWrapper.vhd
vhdl work simulink/netlist/dspcore.dcp
```

- Version control should include:

```
[mdewart@tid-pc92646 DspCoreLib]$ ls
AdcToDacLoopBackCore/  LICENSE.txt

[mdewart@tid-pc92646 AdcToDacLoopBackCore]$ ls
cores.txt  rtl/  sim.txt  simulink/  sources.txt

[mdewart@tid-pc92646 rtl]$ ls
AdcToDacLoopBackCore.vhd

[mdewart@tid-pc92646 simulink]$ ls
ADCtoDACloopback.slx  netlist/
```

Note: netlist included in version control so simulink model doesn't need to be rebuilt on checkout

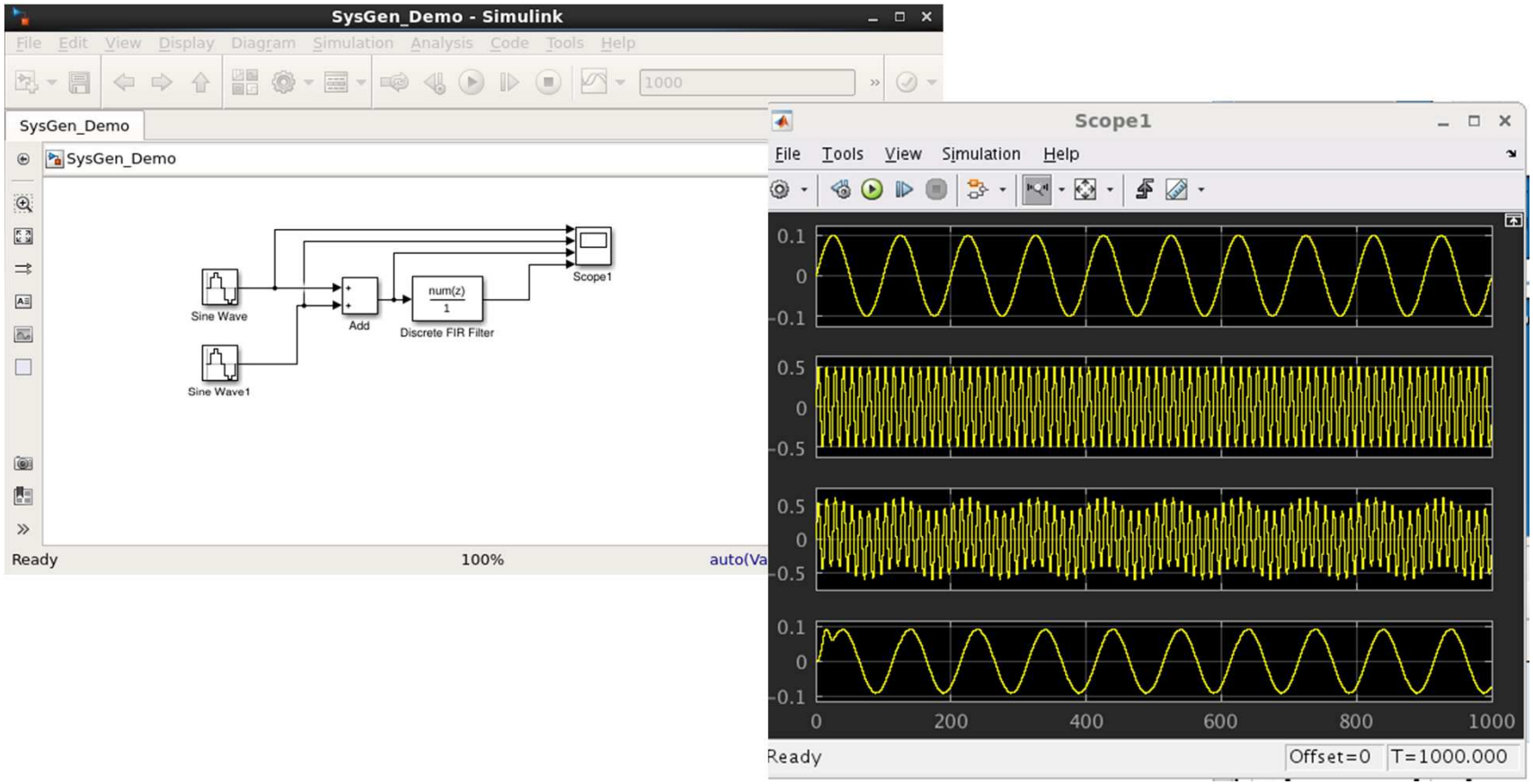
Model Based Design with System Generator

SLAC

- Design process
  - Generate high level design using Matlab/Simulink
  - Create FPGA specific design using System Generator
  - Simulate and compare FPGA vs Matlab/Simulink models
- Examples
  - Discrete filter
  - FDATool integration
  - Digital down converter (DDC)

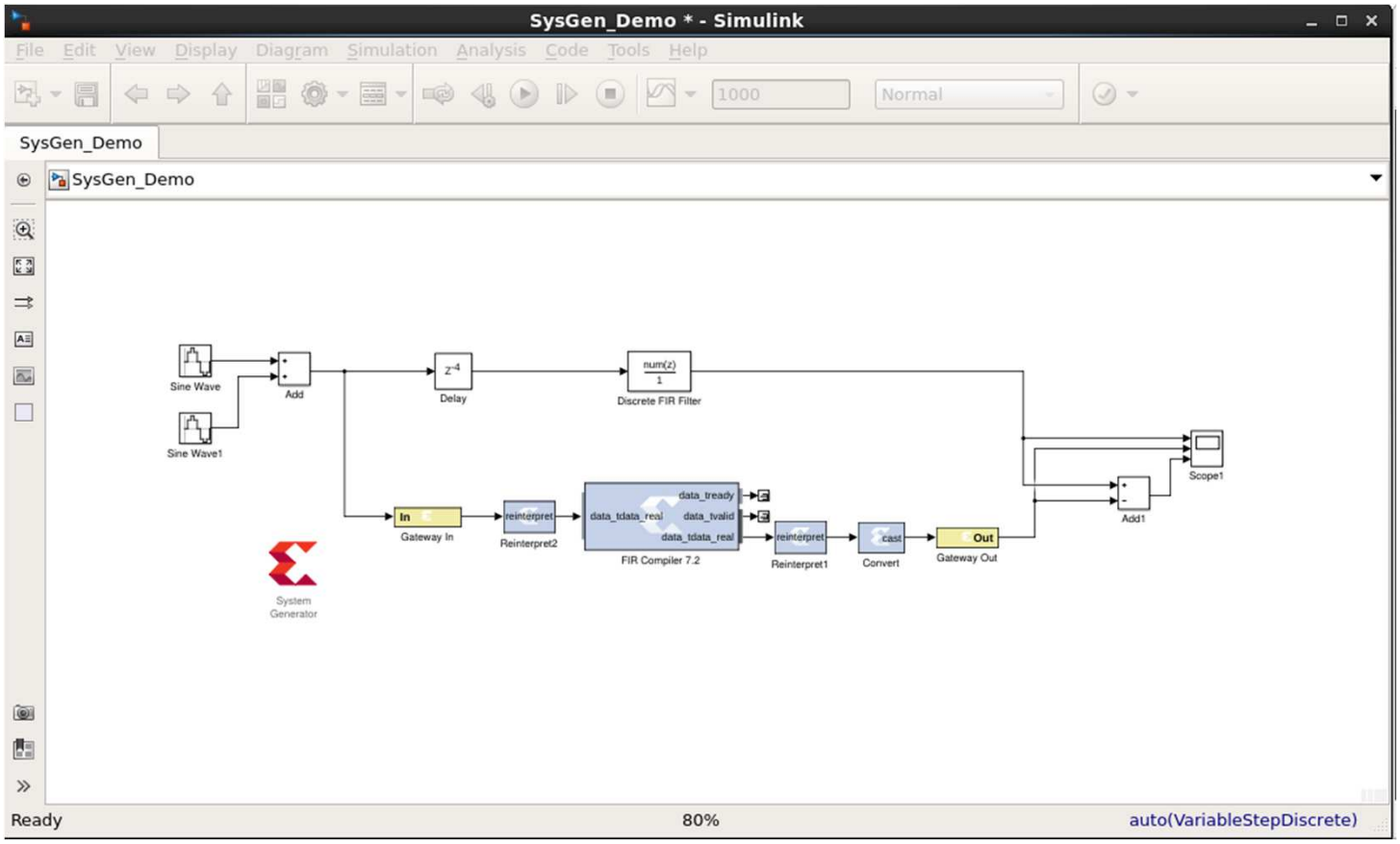
Model Based Design Example FIR Filter

- Design FIR filter using Matlab/Simulink



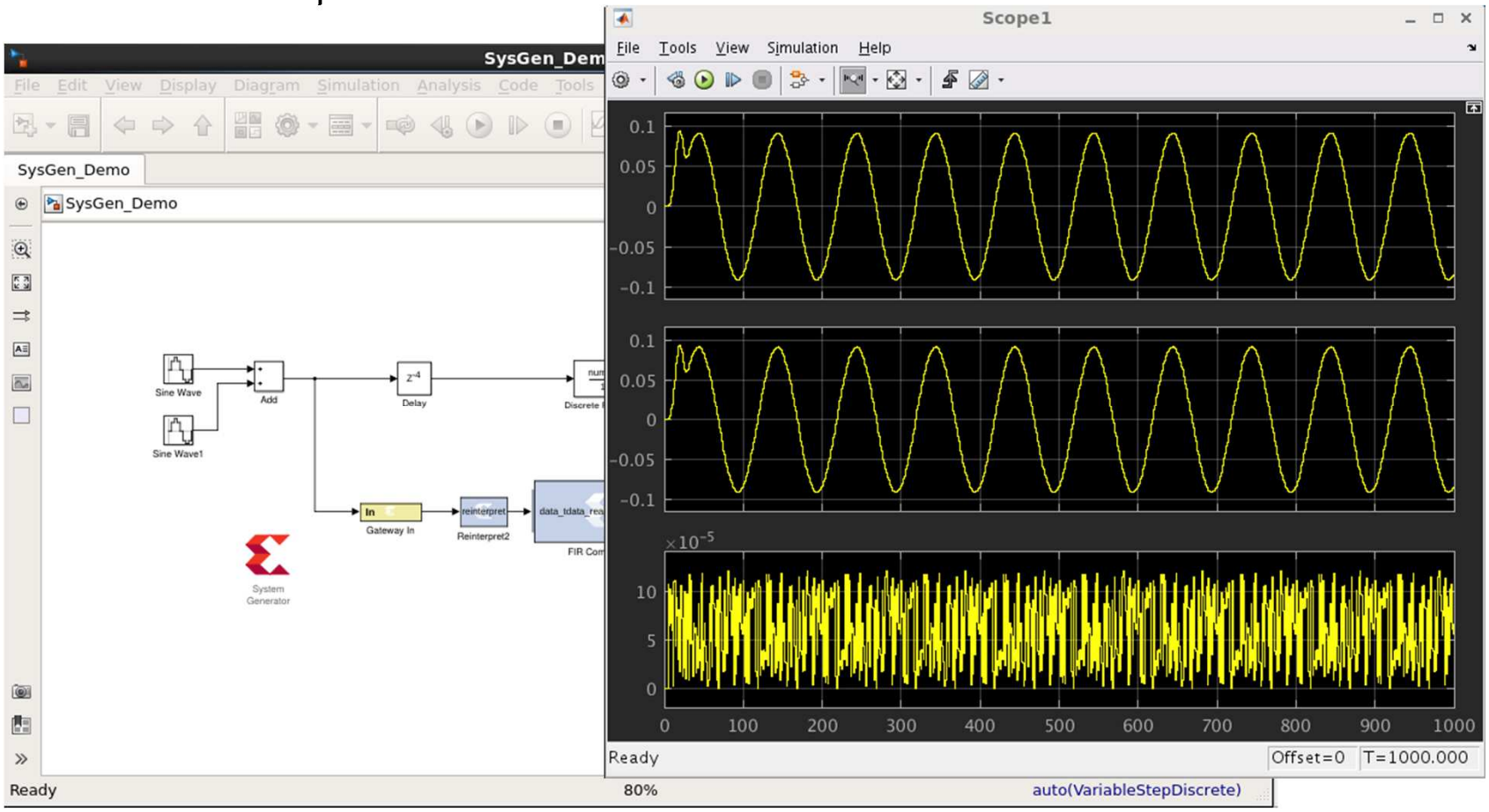
Model Based Design Example FIR Filter

- Generate FPGA specific design with System Generator

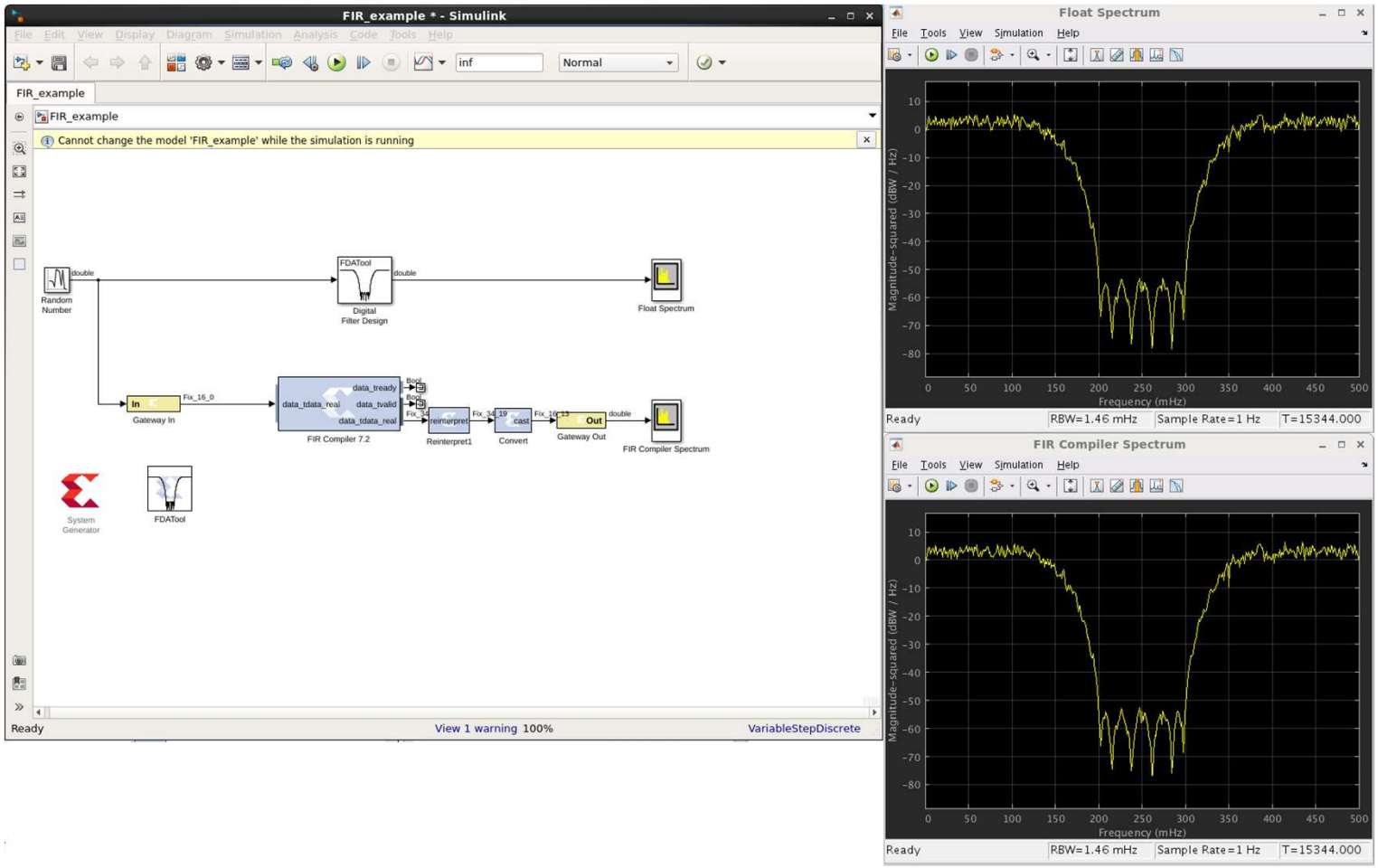


Model Based Design Example FIR Filter

- Simulate and compare



FDATool Example



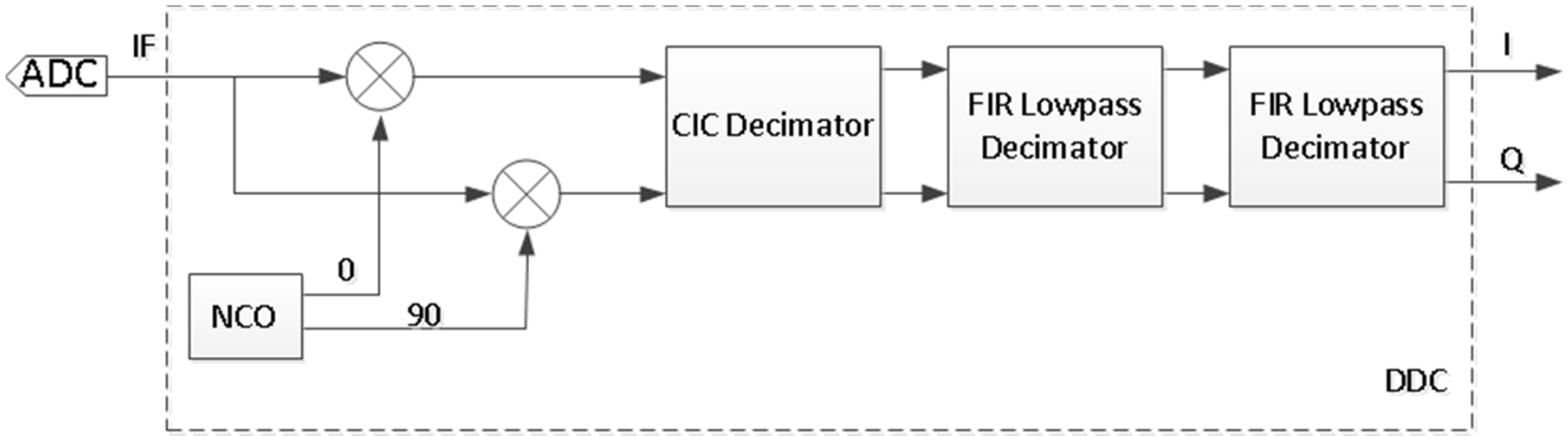
Example Digital Down Converter (DDC) Design

SLAC

- Generate high level design using Matlab/Simulink
  - 357 MS/s ADC
  - 85 MHz IF
  - Decimation factor 64
  - Filter bandwidth 2.33 MHz
  - Stopband attenuation 80 dB
  - Passband ripple 0.4 dB

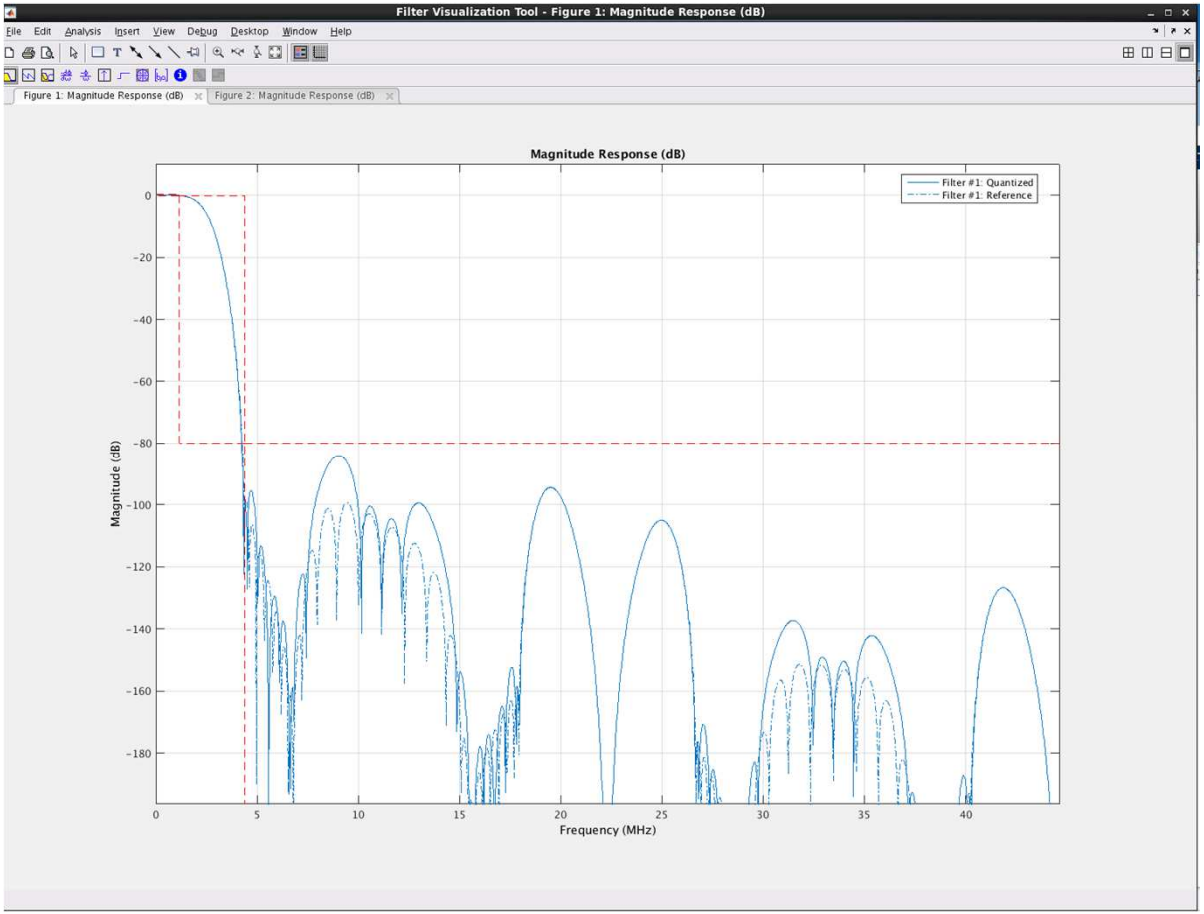
Example Digital Down Converter (DDC) Design

- Generate high level design using Matlab/Simulink



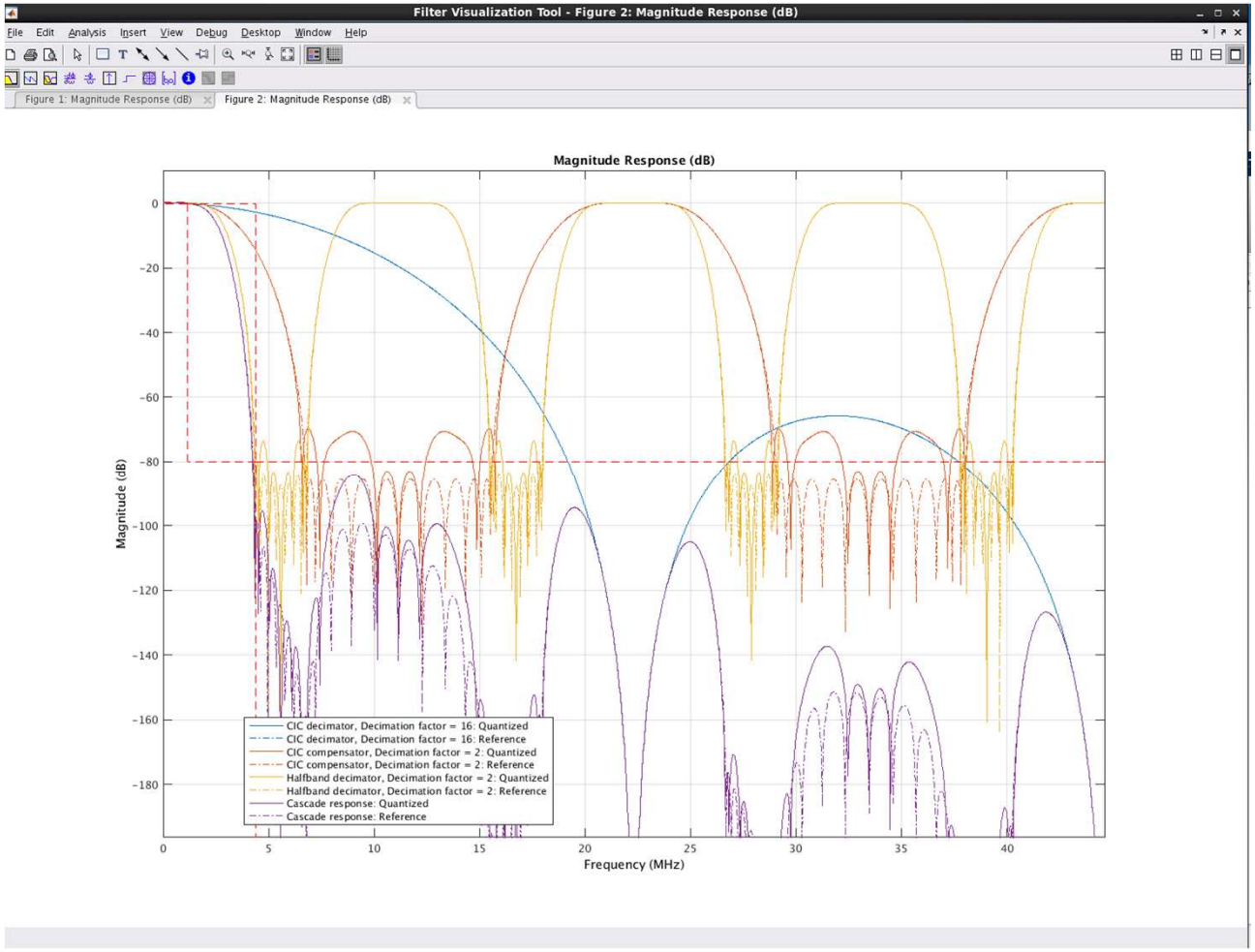
Example Digital Down Converter (DDC) Design

- Matlab design overall filter response



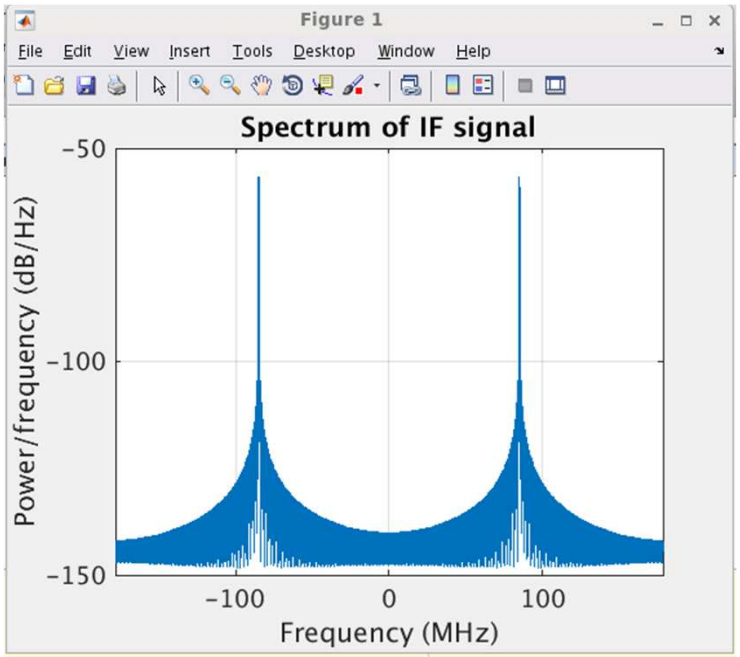
Example Digital Down Converter (DDC) Design

- Matlab design sub filter response

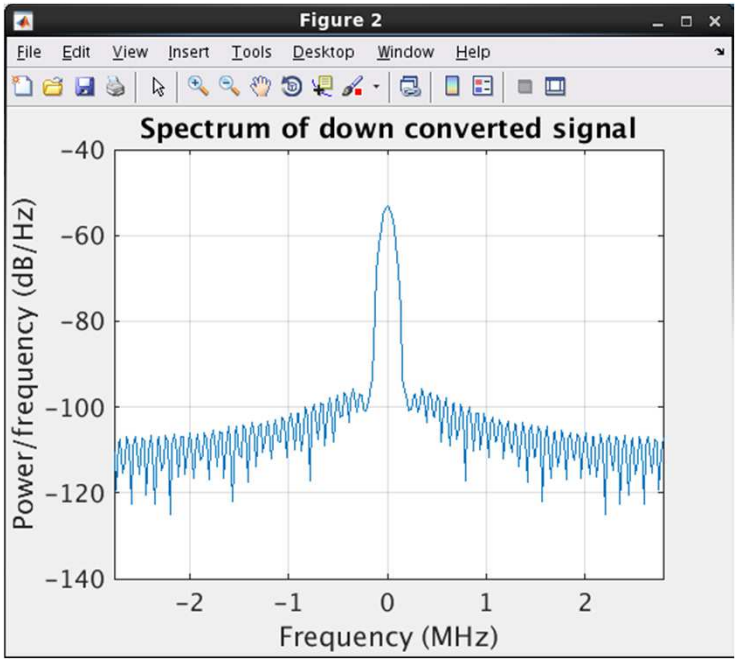


Example Digital Down Converter (DDC) Design

- Matlab simulation with simulated IF

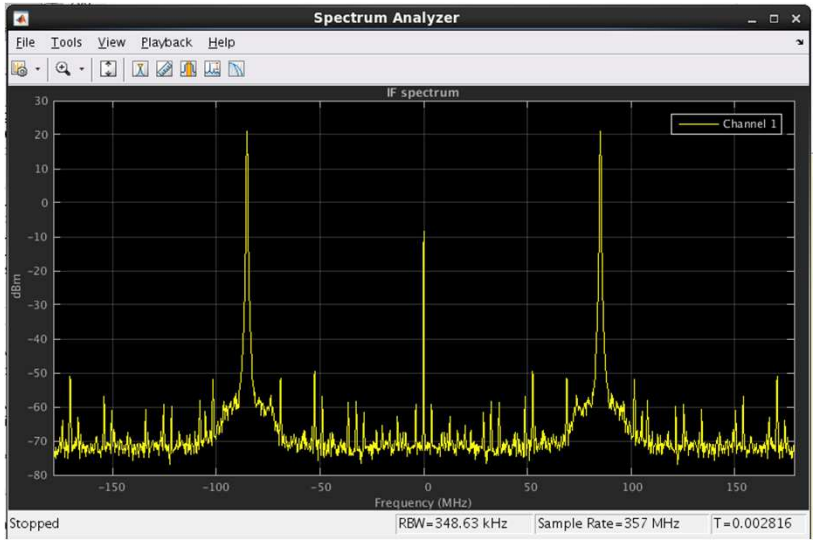


DDC

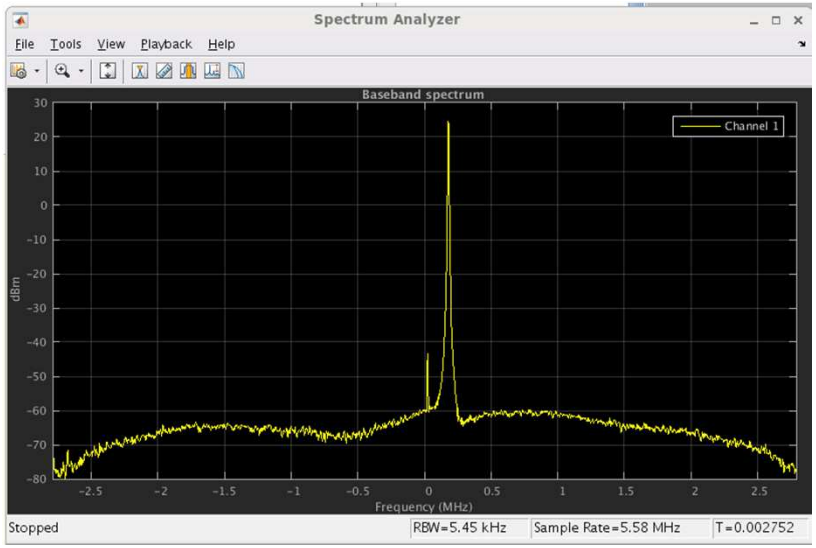


Example Digital Down Converter (DDC) Design

- Matlab simulation with real data

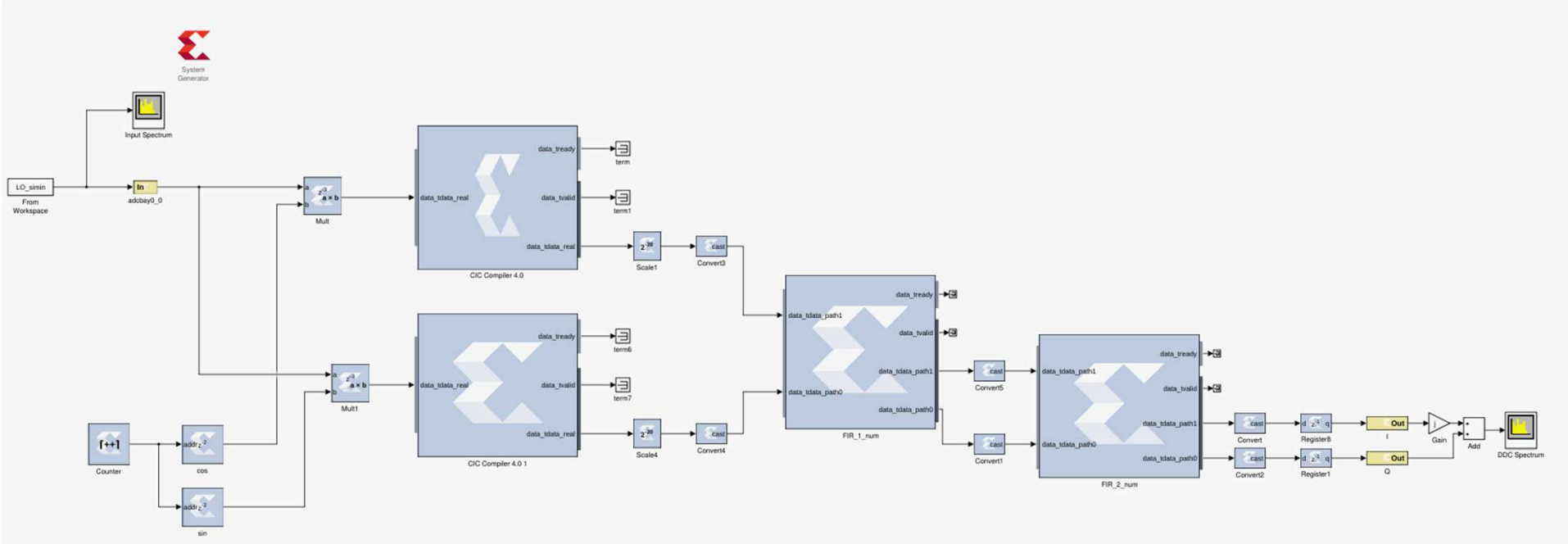


DDC

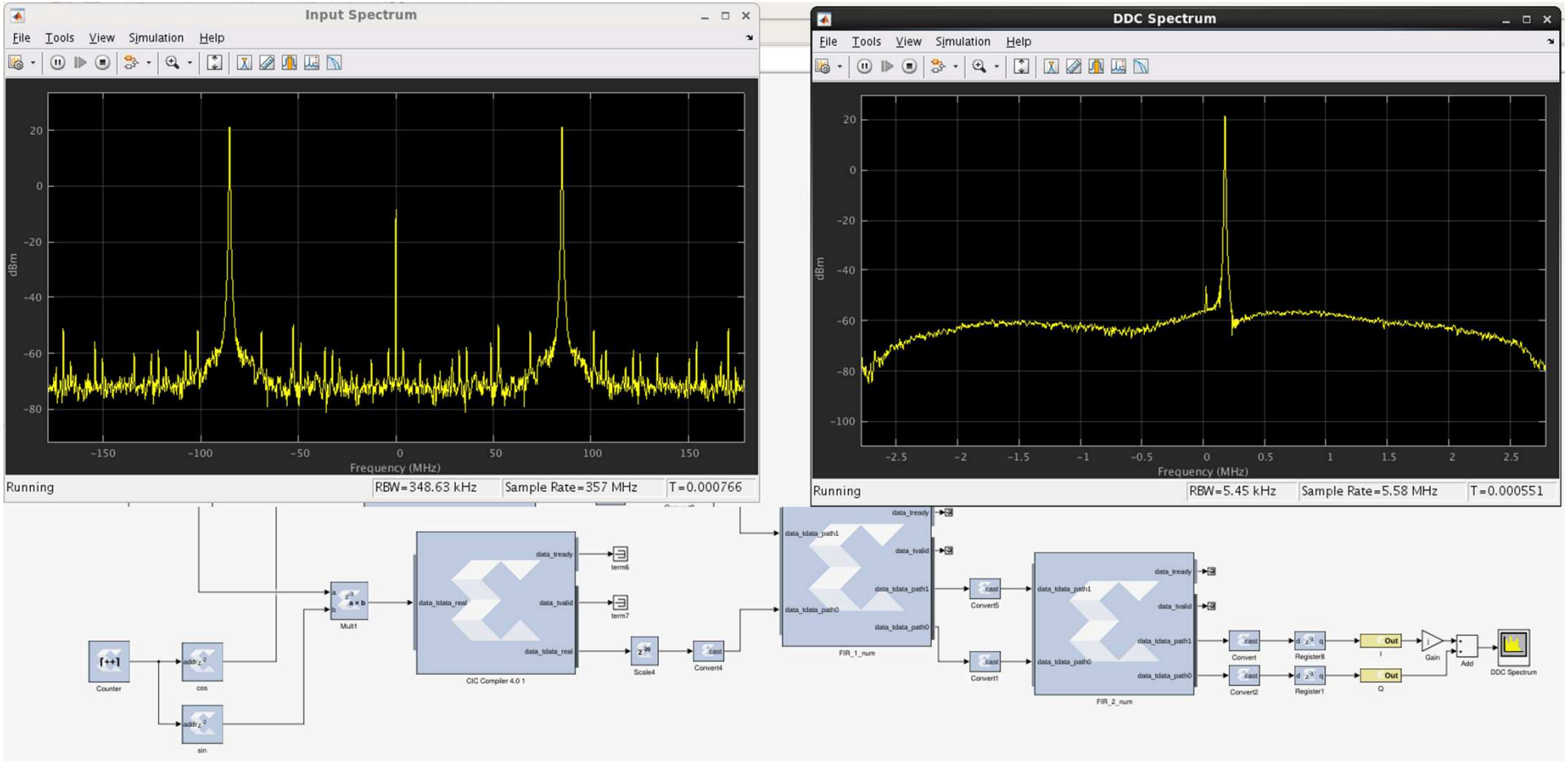


Example Digital Down Converter (DDC) Design

- System generator implementation

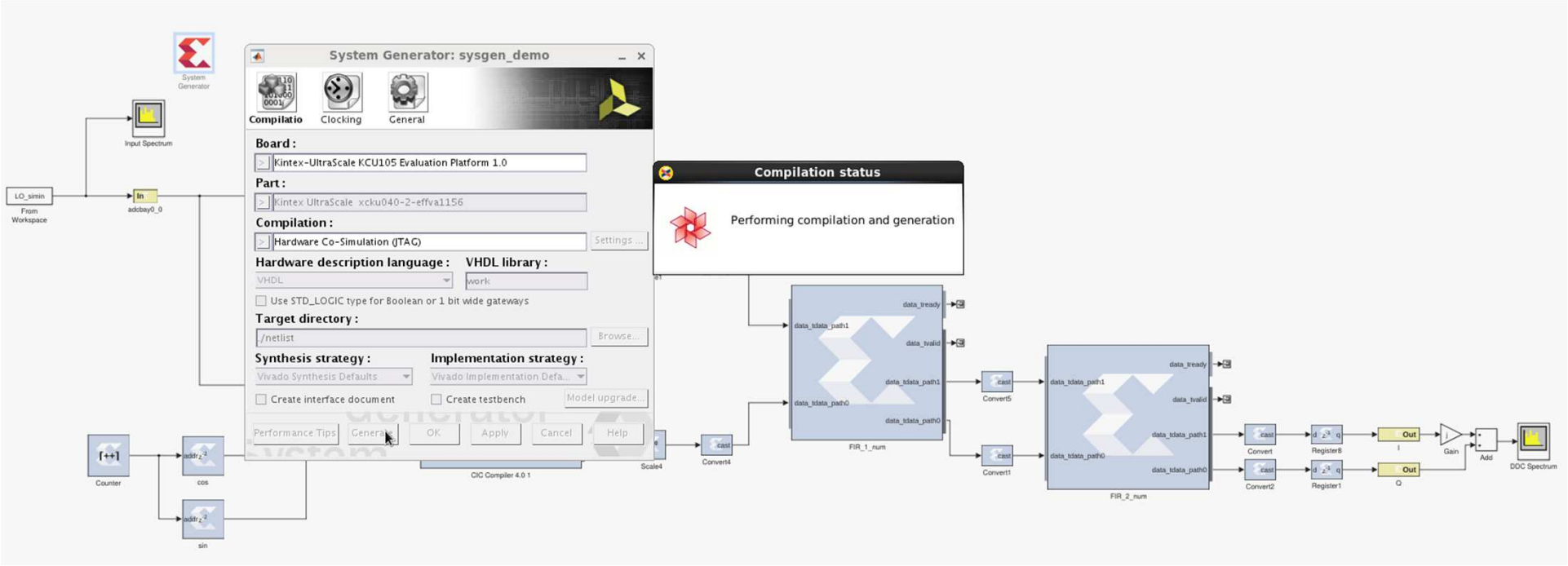


Example Digital Down Converter (DDC) Design



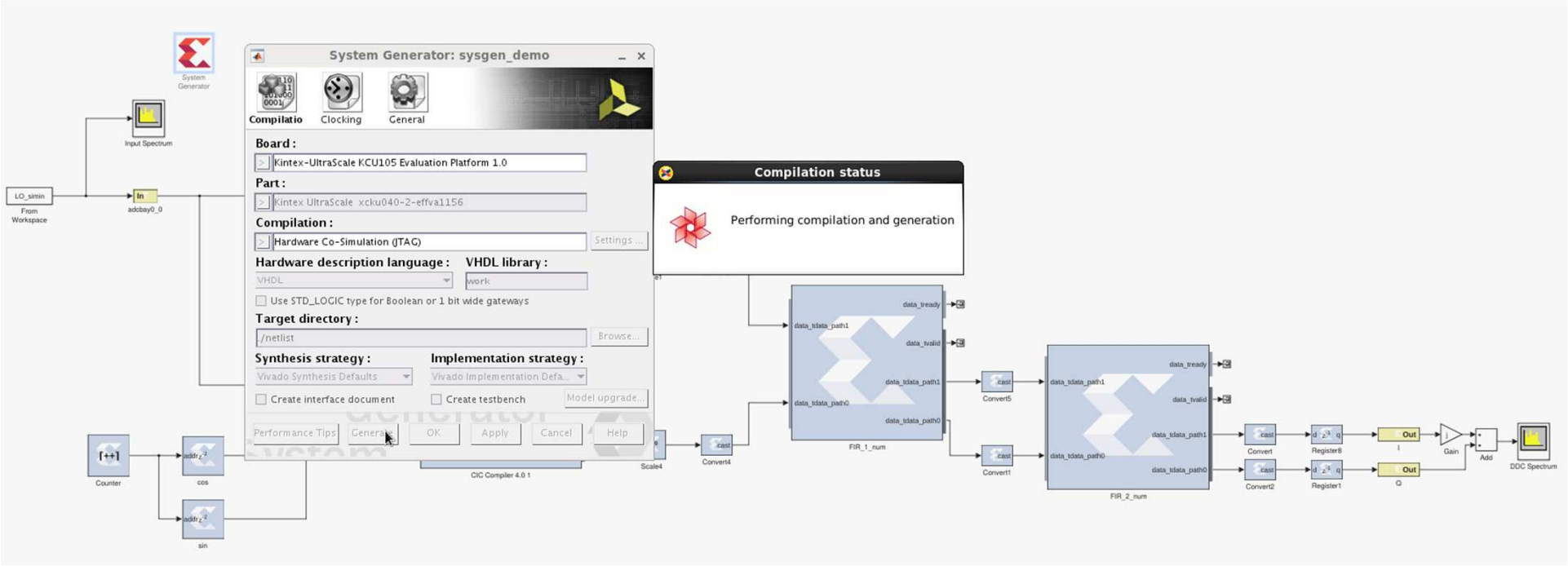
Example Digital Down Converter (DDC) Design

- System generator co-simulation



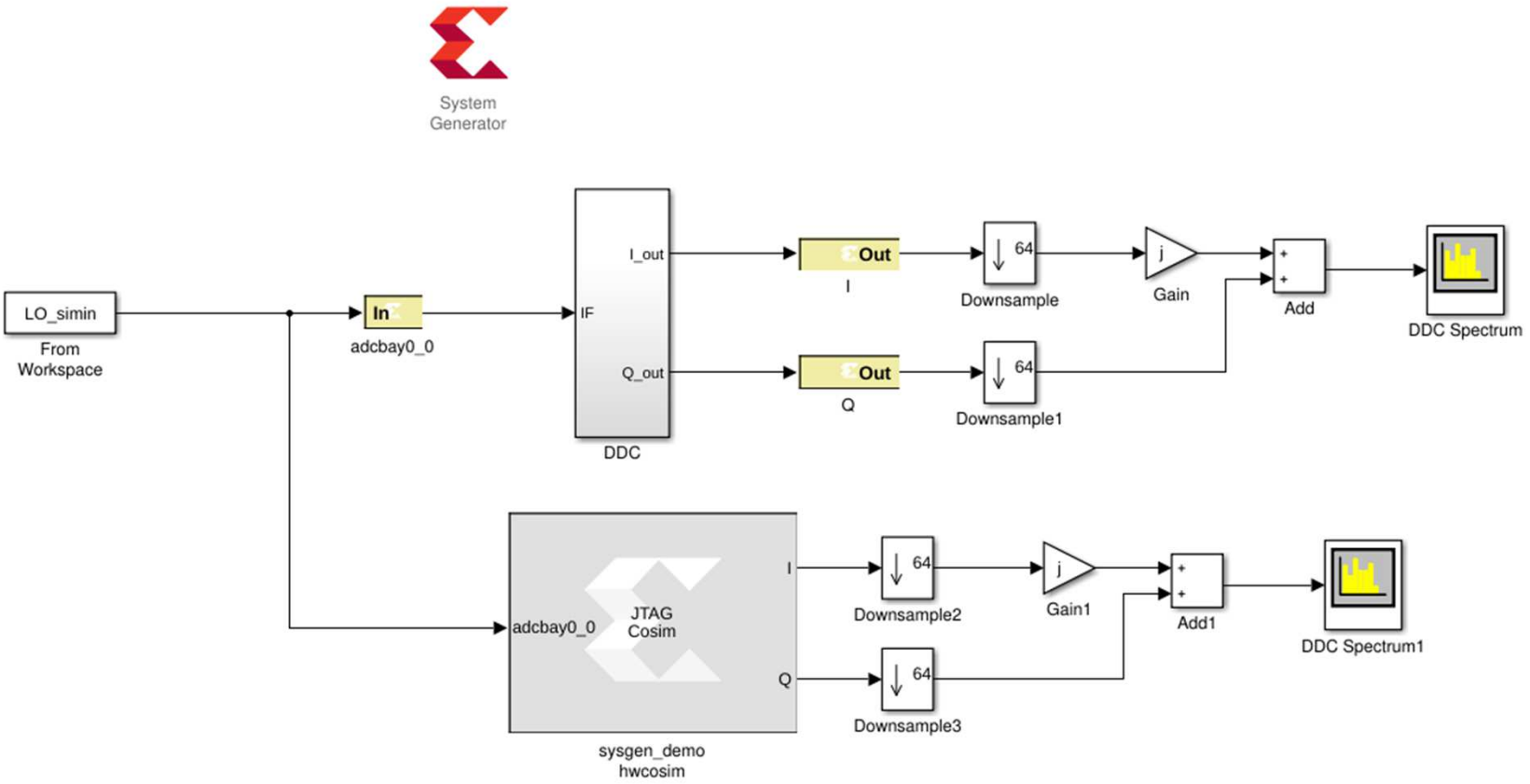
Example Digital Down Converter (DDC) Design

- System generator co-simulation

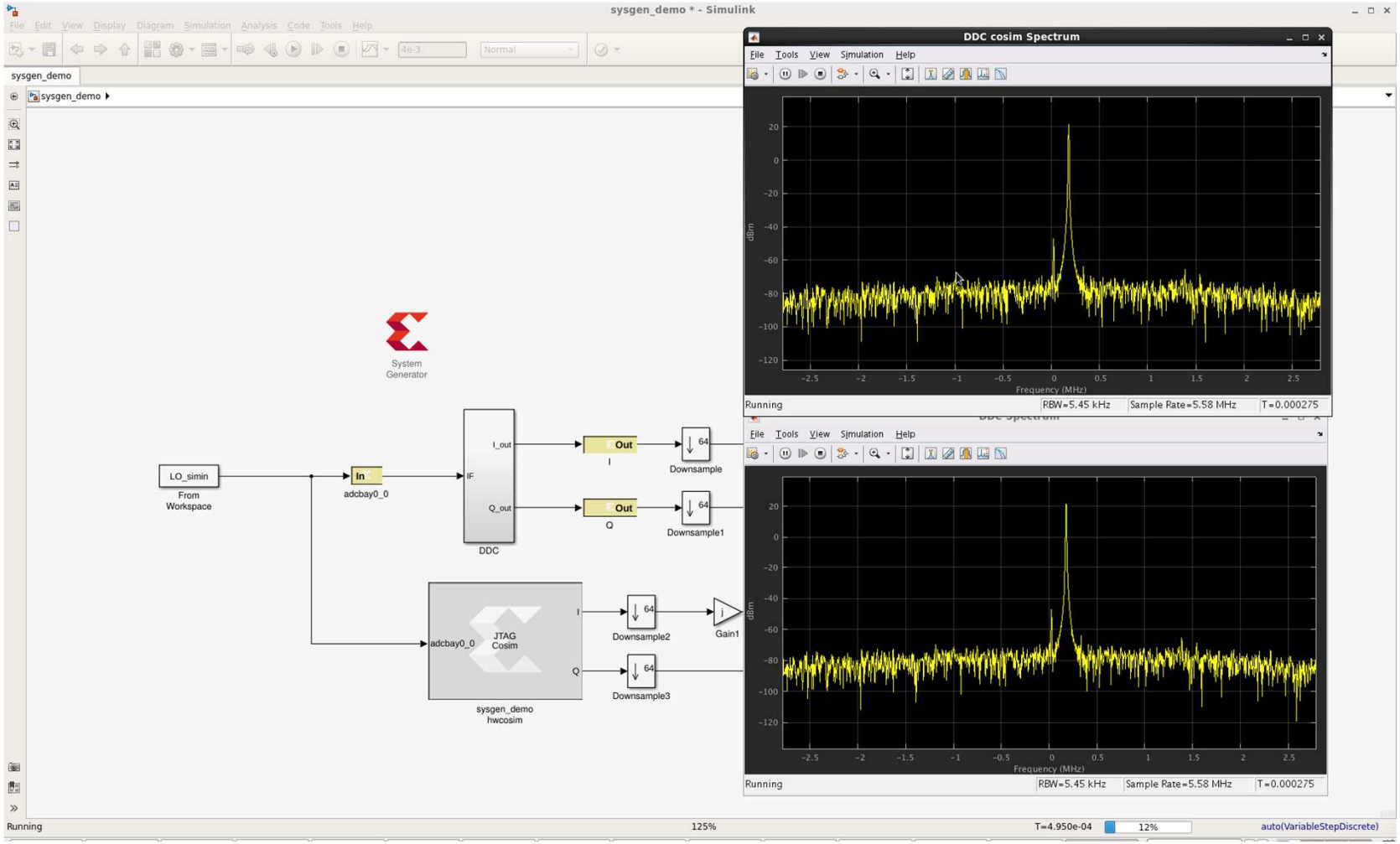


Example Digital Down Converter (DDC) Design

- System generator co-simulation

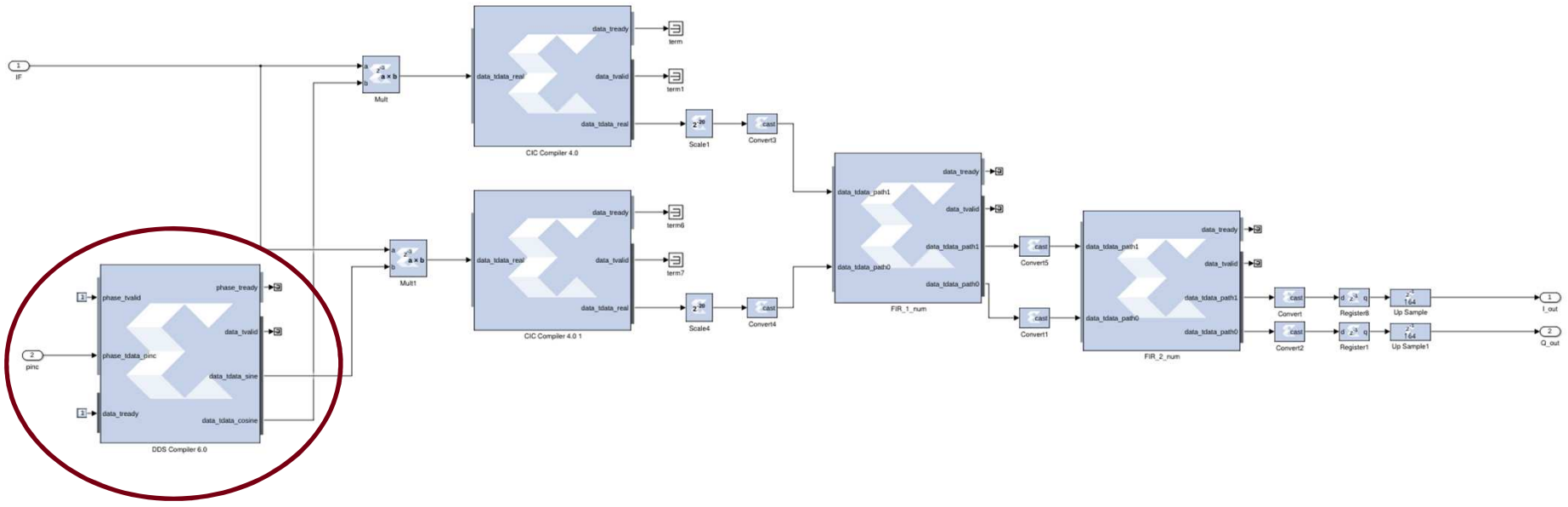


Example Digital Down Converter (DDC) Design



Example Digital Down Converter (DDC) Design

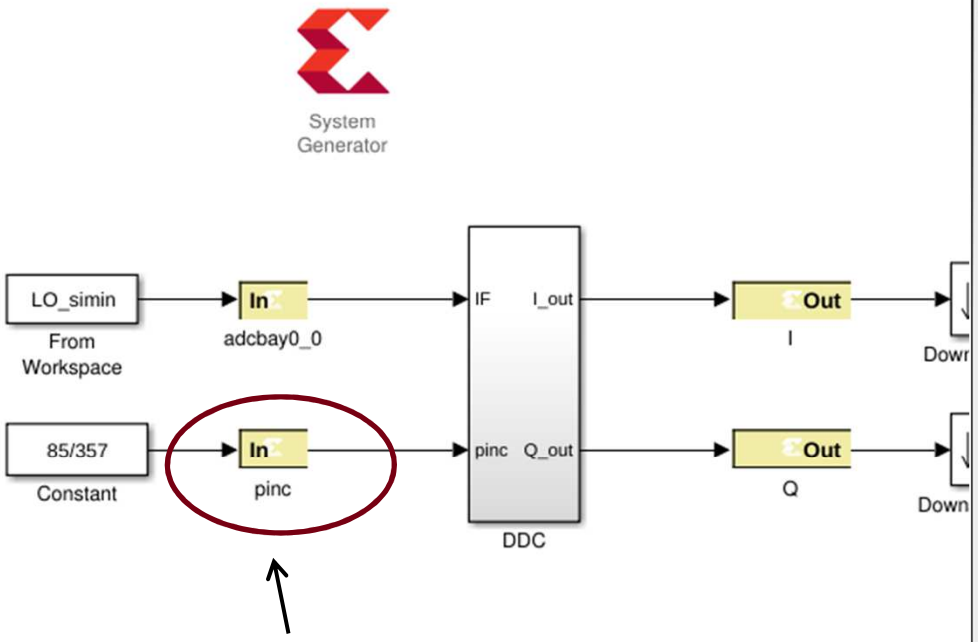
- Modify design to add configuration for digital LO



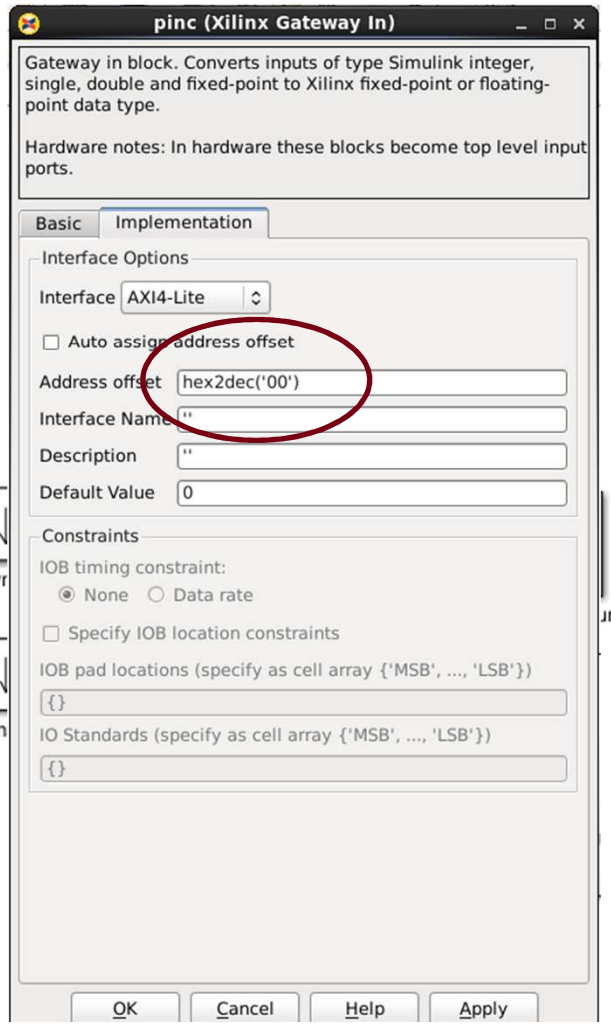
Tunable DDS

Example Digital Down Converter (DDC) Design

- Modify design to add configuration for digital LO



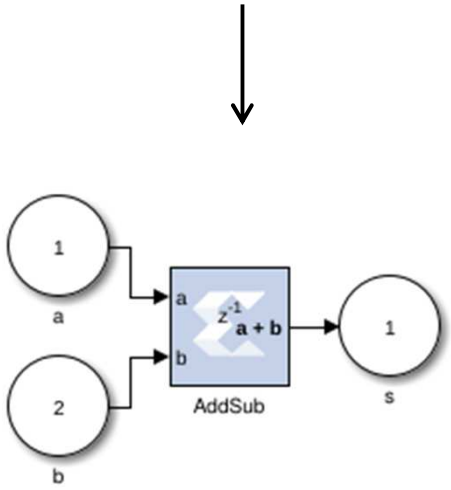
AXI4-Lite register for PINC



Conclusion

- Simulink/System generator allows for rapid algorithm development
  - Know the FPGA target
- System generator is used as a portion of a larger design
- System generator integration with common platform
  - Define interface to DSP core
  - Build system generator model
    - Synthesized checkpoint
  - Build system generator wrapper

System Generator API for Programmatic Generation

- MATLAB interface to build system generator subsystm
  - Used to instantiate and connect xBlock, xSignal, xInport, and xOutport objects

```
[a, b] = xInport('a', 'b');
s = xOutport('s');
adder = xBlock('AddSub', struct('latency', 1), {a, b}, {s} );
```