

# *Getting Started with System Generator for DSP*

## *Lab 2 - Design Creation Basics*

# Design Creation Basics

---

## Introduction

---

This lab will introduce you to the basic concepts of creating a design using System Generator within the model-based design flow provided through Simulink. The design is a simple multiply-add circuit.

---

## Objectives

---

After completing this lab, you will be able to:

- Understand the basics of building a design in System Generator
- Simulate a design in System Generator
- Run the System Generator token to generate a Xilinx FPGA bitstream
- Create a subsystem
- Improve performance using dedicated Xilinx FPGA math functions

---

## Lab Setup

---

Please check the System Generator for DSP release notes to insure that the proper versions of ISE Design Suite and MATLAB are installed on your machine. Failure to have the proper tool versions installed may result in unexpected behavior.

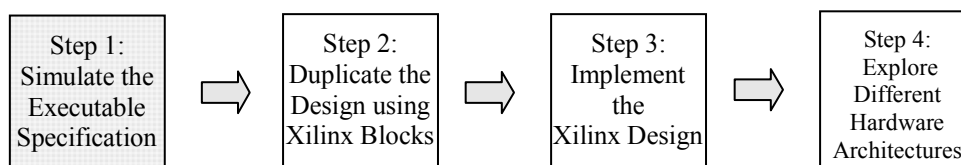
## Procedure

This lab has four primary steps. In Step 1, you open and simulate a Simulink blockset-based design that serves as an “executable specification.” In Step 2, you re-create the Simulink design using the Xilinx blockset. In Step 3, you take the Xilinx executable specification through the full implementation flow. Finally, in Step 4, you explore different hardware architectures to achieve the best performance.

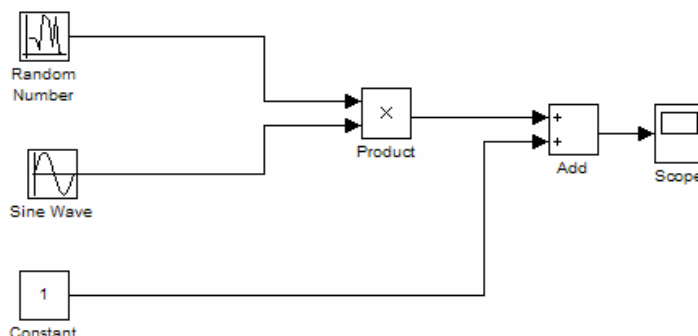
## Simulate the Executable Specification

## Step 1


### General Flow for this Lab:

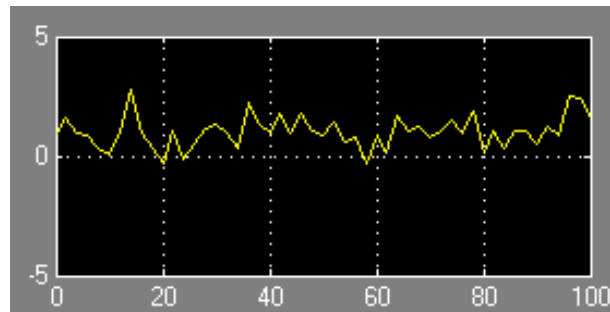


- ❶ Launch the MATLAB program and change the working directory to:  
...sysgen/examples/getting\_started\_training/lab2
- ❷ Open the file **lab2.mdl** and observe the following design.



**Note:** This design is an executable specification created in Simulink using the standard Simulink blockset. It is a simple multiply-add circuit but serves to demonstrate many of the key concepts of model-based design. You are going to design a Xilinx FPGA to this spec.

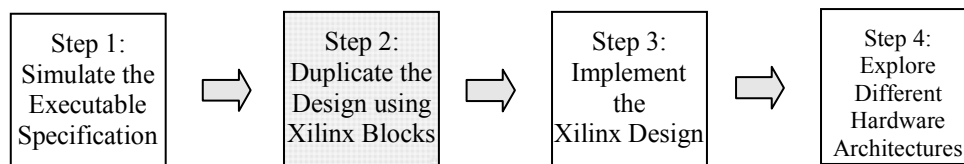
- ③ Simulate the design for 100 cycles by pressing the play button  on the toolbar. View the waveform by double-clicking on the **Scope** block.



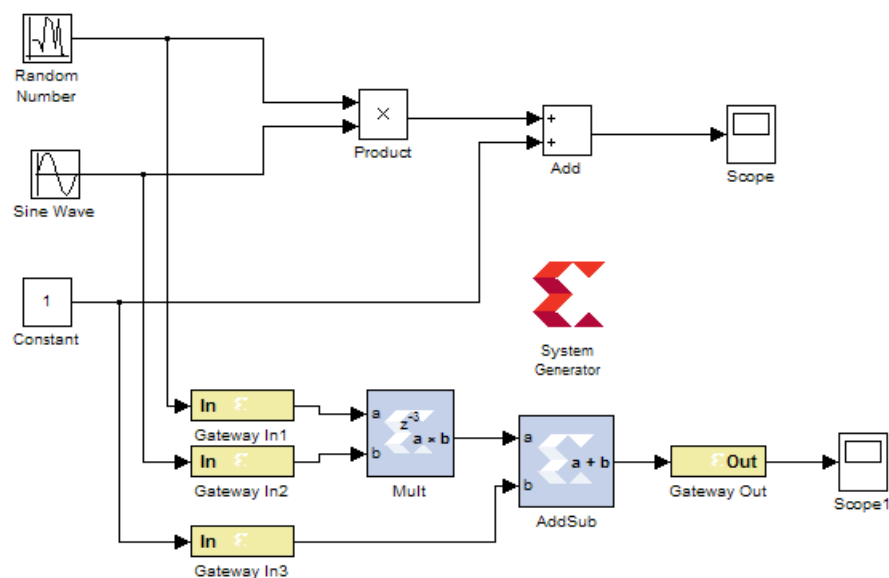
## Duplicate the Design using Xilinx Blocks

## Step 2

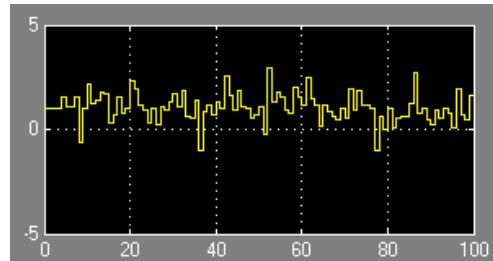
### General Flow for this Lab:



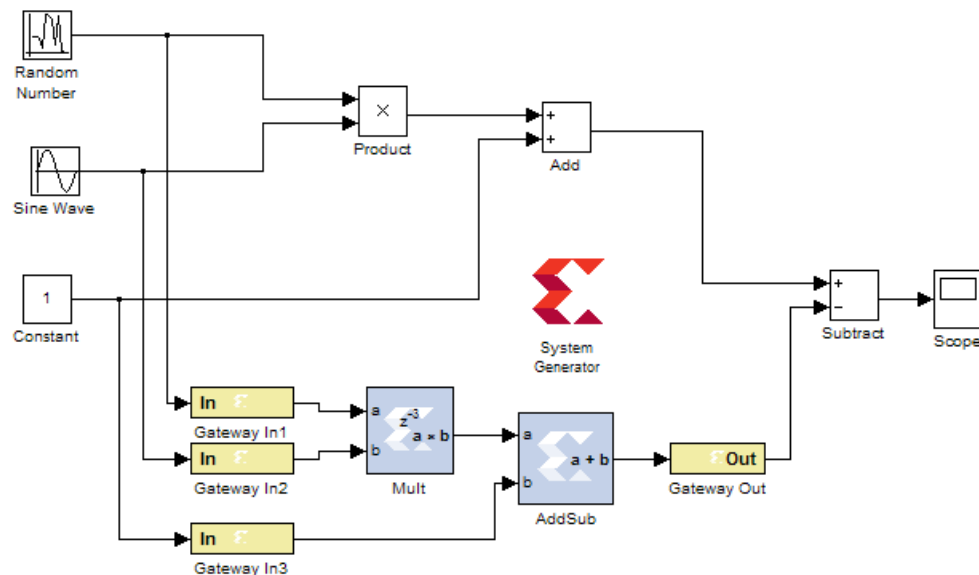
- ① From the Simulink Library Browser, open the Xilinx Blockset Library and the **Index** sub-library to access the blocks. Create a Xilinx version of the multiply / add design using Xilinx blocks. Remember you must use Xilinx **Gateway In** / **Gateway Out** blocks to define the FPGA boundary and you must also place a **Xilinx System Generator** token in the design, as shown in the figure below. Leave all the block settings at their default values.



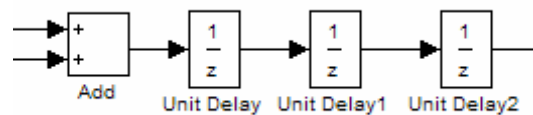
- ② Simulate the design and view the waveform on the **Scope** attached to the Xilinx implementation. Notice that the waveforms have square edges. This is because the System Generator blockset forces a discrete sampling of the input signals which represents the behavior of the actual hardware that operates on synchronous clock edges.



- ③ Compare the results from the executable spec vs. the Xilinx implementation using a **Subtractor** from the “Simulink/Math Operations” library as shown below. This is an important model-based design concept.



- ④ Add 3 **Unit Delay** blocks to the output of the Add block as shown below and re-simulate the design.

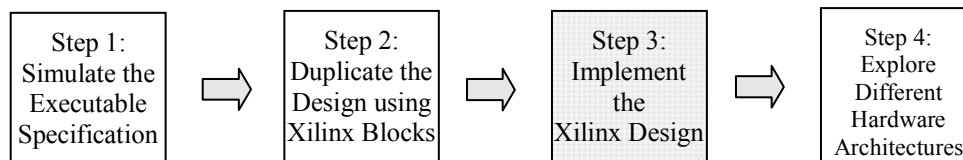


The design is now functionally matching the executable spec. The next step is to perform the FPGA implementation steps that include RTL generation, RTL synthesis and Place and Route.

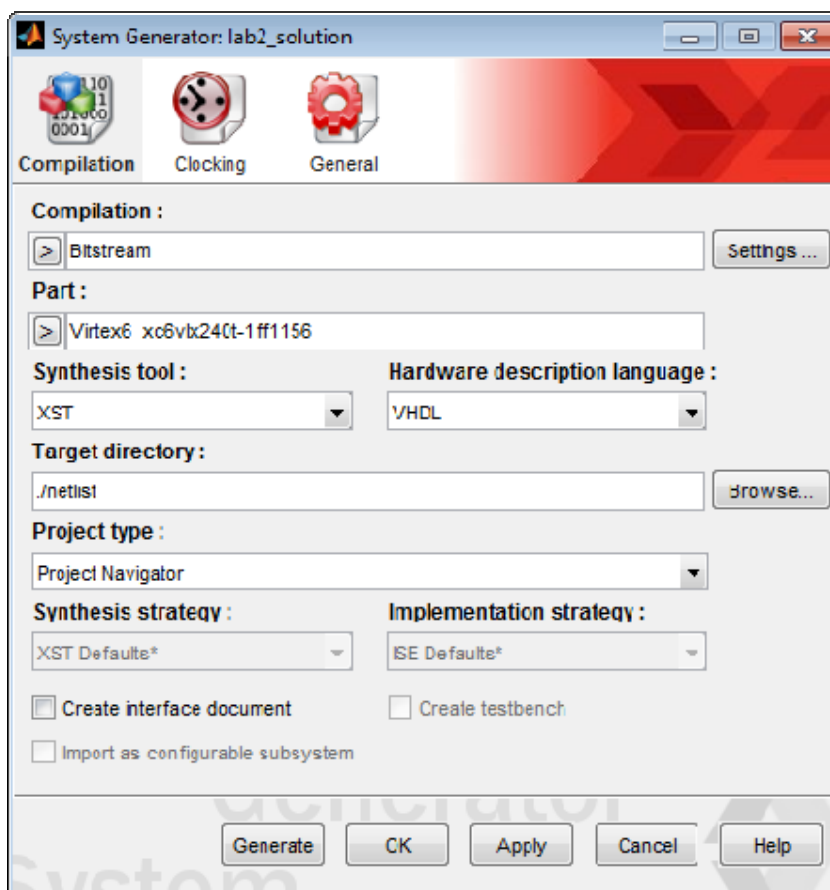
## Implement the Xilinx Design

## Step 3

### General Flow for this Lab:



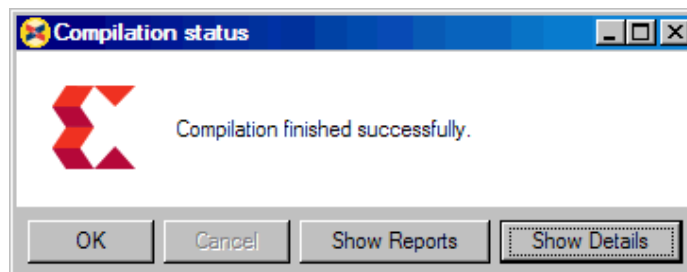
- ❶ Double-click on the **System Generator** token and set the Compilation target to **Bitstream**, as shown below. Also, verify that the output device is set to **Virtex6 xc6vtx240t-1ff1156**.



- ❷ Click the **Generate** button to initiate the implementation process.

System Generator will automatically execute the RTL generation, logic synthesis and place and route programs to create an FPGA programming file. Optionally, you can select to generate an intermediate format such as HDL (logic synthesis) or NGD (place and route) and run these steps interactively.

- ③ When the generation is complete, click on the **Show Details** button on the **Compilation status** dialog box as shown below. This will display the implementation transcript.



Record the following results from the transcript:

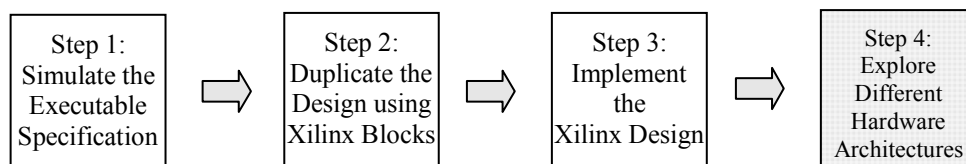
# of DSP48 \_\_\_\_\_  
# of slices \_\_\_\_\_

You now have an initial implementation. The remainder of this lab will focus on some common techniques used to explore different hardware architectures.

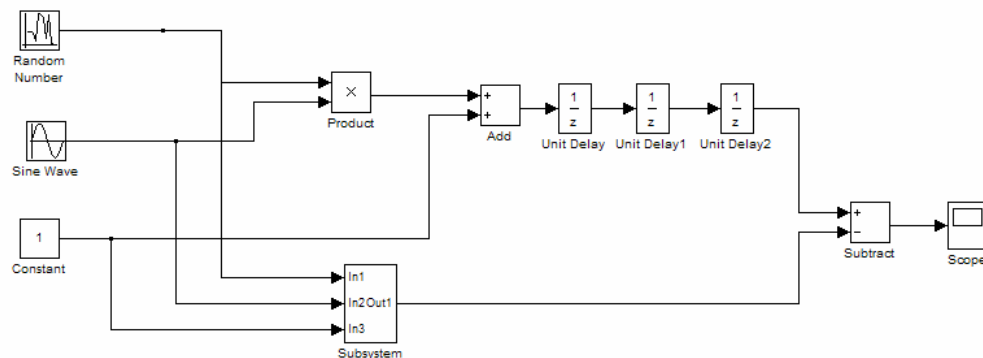
## Explore Different Hardware Architectures

## Step 4

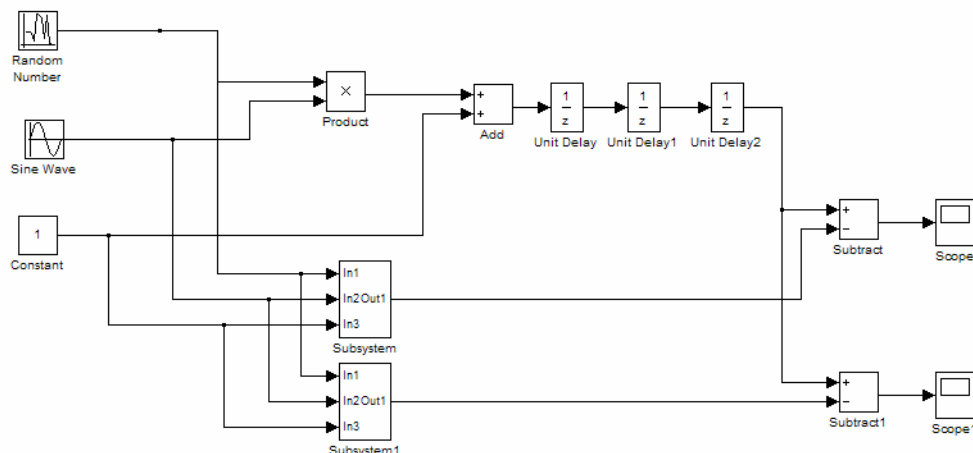
### General Flow for this Lab:



- ① Select all the Xilinx components, including the **System Generator** token and push them into a subsystem by pressing **Ctrl-G**. The diagram should look like the following figure:

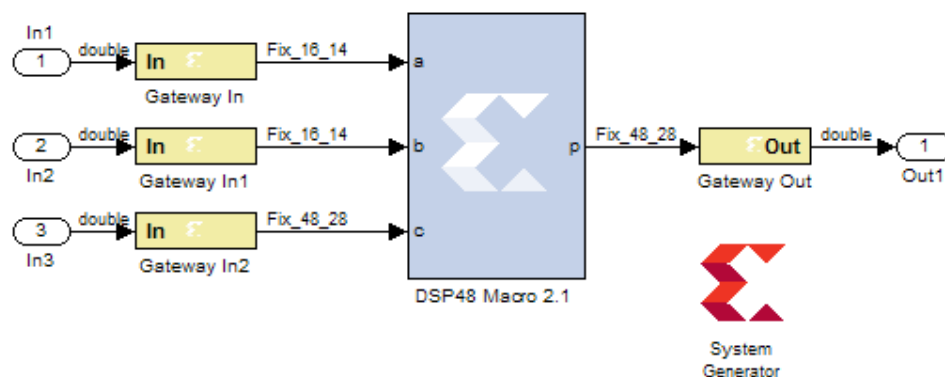


- 2 Copy the subsystem to create a second subsystem and connect it to the design as shown in the figure below:



You now have a Simulink diagram that contains two subsystems each with a **System Generator** token. This represents two FPGAs or two blocks of a single FPGA within a larger DSP system. Each **System Generator** token creates a top-level entity from the subsystem from which it is associated. It will not merge with the other subsystem. Creating subsystems can be a useful technique when exploring hardware architectures for a given design.

- 3 Push into the copy of the subsystem and modify the design to implement the same function using the Xilinx DSP48 2.1 macro block.



Using this block allows improved control over the hardware implementation. The DSP48 macro will force the use of the DSP48 primitives in the final netlist. Port c on the DSP 48 Macro 2.1 block requires an input quantization of Fix\_48\_28. Double-click on the Gateway In2 block, set the **Number of bits** to 48 and the **Binary point** to 28, then click OK.

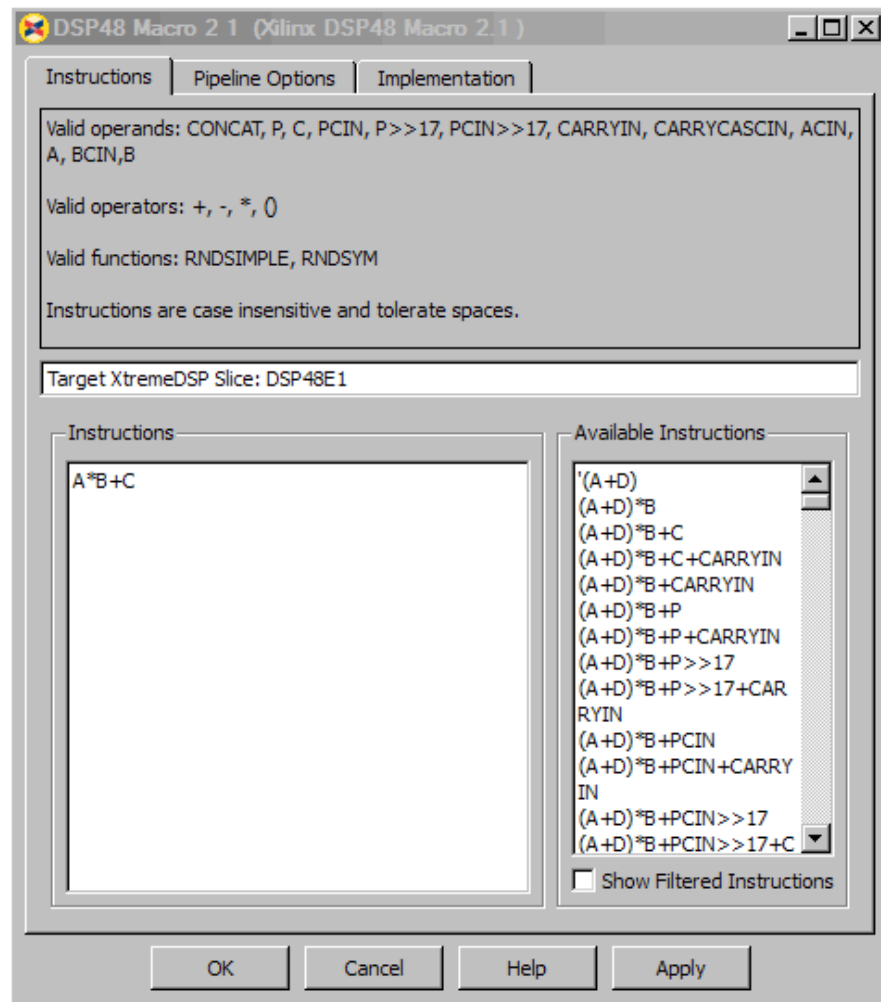
To view the port quantization values as shown above, execute the following pulldown menus:

**Format > Port/Signal Displays > Port Data Types**

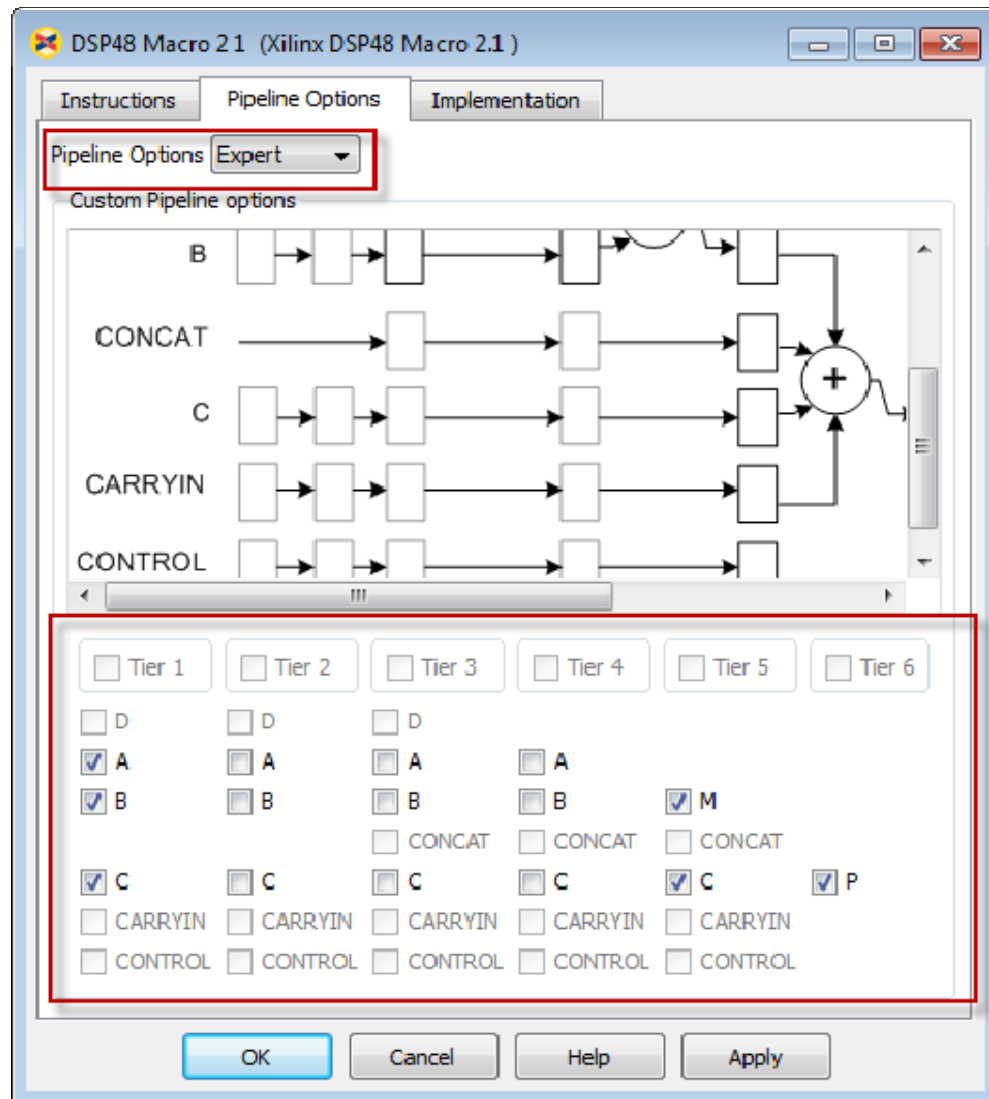
**Format > Port/Signal Displays > Signal Dimensions**



- ④ Double-click on the DSP48 2.1 Macro block to bring up the properties dialog box editor. From the **Instructions** tab, verify the equation to be  $A*B+C$ , as shown below. Observe the other implementation options but leave them at their default values.



Click on the **Pipeline Options** tab and set the Pipeline Options to **Expert**, as shown below.



As shown above, click the **A**, **B**, and **C** boxes in Tier 1, **M** and **C** boxes in Tier 5, and the **P** box in Tier 6. Un-click all other boxes, then click **OK**.

- ⑤ Re-simulate the design to insure functional correctness
- ⑥ Double-click on the **System Generator** token to generate a bitstream. Record the following:

# of DSP48 \_\_\_\_\_  
 # of slices \_\_\_\_\_

## Solution

The complete solution to this lab is in the following location:  
**...sysgen/examples/getting\_started\_training/lab2/solution**

