Lab 2: Creating a 12 x 8 MAC Using the Xilinx System Generator

Creating a 12 x 8 MAC Using the Xilinx System Generator

Introduction

In this lab, you will create the same 12-bit x 8-bit MAC (Multiplier Accumulator) that you created in Lab 1 by creating a multiplier and an accumulator in the Xilinx System Generator, and estimate the resource utilization using the Resource Estimator block of the System Generator. The System Generator allows you to use blocks that represent Xilinx LogiCOREs. After design and verification in the software environment of System Generator, you will generate VHDL code and cores from this design, and implement the MAC in the Xilinx ISE 6 (Project Navigator) software environment

Note: There is a completed example in c:\xup\dsp_flow\labs\lab2\lab2_soln.

Objectives

After completing this lab, you will be able to:

- Handle the basics of Simulink
- Understand the basics of building a design in System Generator
- Simulate a design in System Generator
- Run the System Generator token to generate VHDL code
- Run designs through the System Generator design flow
- Estimate the resources using Resource Estimator
- Implement the design in Xilinx ISE 6 software environment and generate a bit stream file

Design Description

Use System Generator under Simulink environment running under MATLAB to create a 12 x 8 MAC:

- Multiplier input data widths of 12 bits and 8 bits of signed data
- Multiplier output width of 20 bits
- Accumulator output width of 27-bits

Procedure

This lab comprises nine primary steps: the first four steps introduce you to System Generator and the next five steps guide you through how to create a design starting from System Generator, and finally, implement the design using the Xilinx ISE 6 software environment. You will be introduced to Simulink in step 1, and the Xilinx blockset in step 2. In step 3, you will evaluate the precision, and, in step 4, you will analyze the effect of the sample period upon the output. Step 5 requires you to design a 12 x 8 MAC core using the System Generator software environment, and step 6 requires you to estimate the resources using the Resource Estimator block. In step 7, you will simulate the design in Simulink, and in step 8 you will generate the VHDL core using the System Generator block. In step 9, you will implement the design using the Xilinx ISE 6 software environment. Below each general instruction for a given procedure, you will find accompanying step-by-step directions and illustrated figures providing more detail for performing the general instruction. If you feel confident about a specific instruction, feel free to skip the step-by-step directions and move on to the next general instruction in the procedure.

Note: If you are unable to complete the lab at this time, you can download the lab files for this module from the Xilinx University Program site at http://university.xilinx.com

Introduction to Simulink — Step 1

Become familiar with the MATLAB, Simulink environments (software tools from The MathWorks) and the Xilinx blockset running under the Simulink environment. Create a blank worksheet, add **Sine Wave** source element, add **Scope** sink element, and wire them up as shown in Figure 2-1.

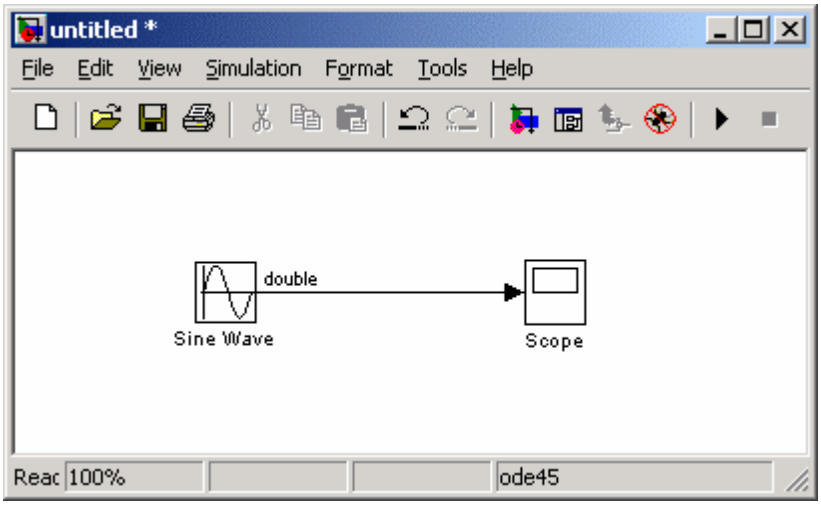


Figure 2–1. Double Precision Design.

❶ Open the MATLAB command window by double-clicking the MATLAB icon on your desktop, or go to **Start Menu → Programs → MATLAB 7 → MATLAB 7**

Figure 2-2. MATLAB Icon.

❷ Change directory to **c:\xup\dsp_flow\labs\lab2**: Type **cd** c**:\xup\dsp_flow\labs\lab2** in the command window

You may navigate to product directories by typing a "cd**"** command in the MATLAB command window. Type "ls**"** to see the directory contents. Many UNIX shell commands work from the MATLAB command Window

❸ Launch **Simulink**: Type **simulink** at the MATLAB command prompt; or open the **Simulink Library Browser** by clicking the corresponding button on the MATLAB toolbar

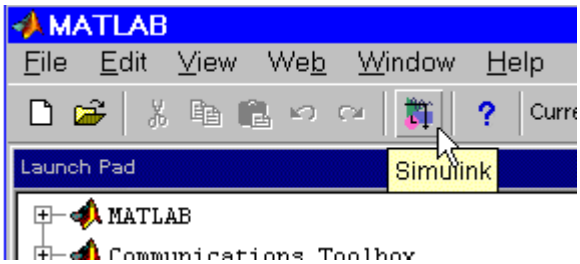


Figure 2–3. Simulink Library Browser.

❹ Look at the blocks available in the Simulink Library Browser. The following elements, among others, should appear:

- **Simulink** (sources and sinks)
- **Signal Processing Blockset**
- **Xilinx Blockset**
- **Xilinx Reference Blockset**

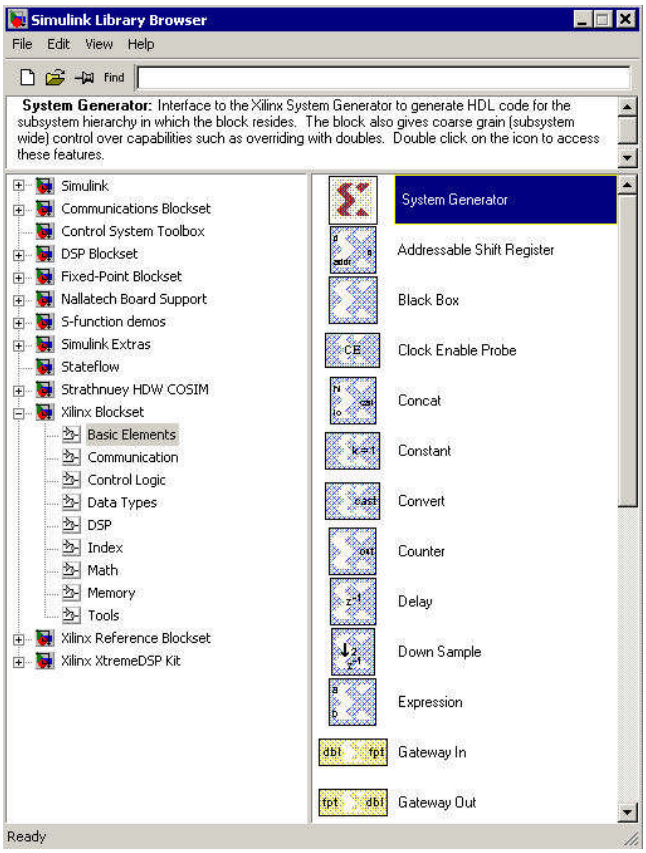


Figure 2–4. Simulink Library Browser.

❺ Right-click any block in the library browser, and choose Help from the MATLAB menu

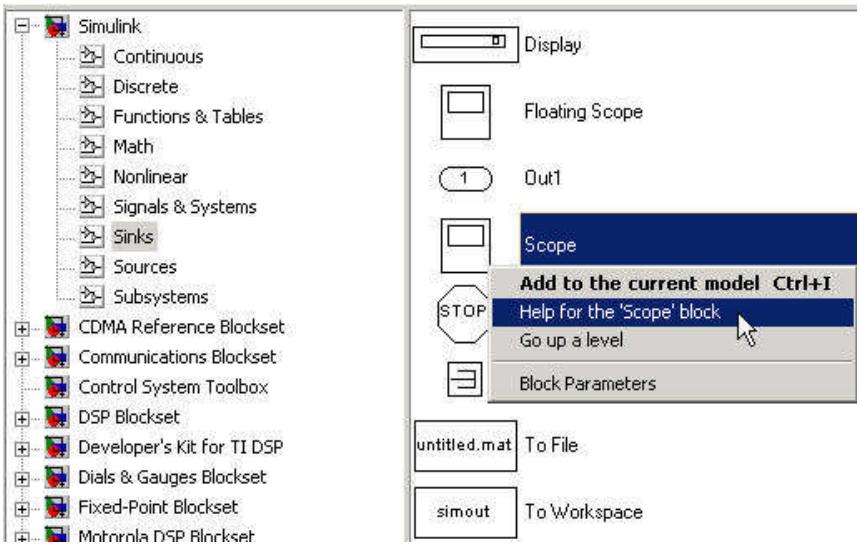


Figure 2–5. Choose Help.

Note: This provides details about the block. You can also use Help with the Xilinx Blockset elements

❻ Create a "new model" blank sheet by clicking the **Create a new model** button of the **Simulink Library Browser**

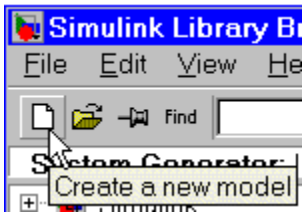


Figure 2–6. Create New Model.

❼ In the **library browser** window, expand the **Simulink Library**, and click **Sources**

❽ Add the **Sine Wave** source on the worksheet: Scroll through the library to find the **Sine Wave** source, left click **Sine Wave**, and drag it onto the worksheet

❾ Add the **Scope** sink element and wire it to the **Sine Wave** source on the worksheet: From Simulink **Simulink → Sinks,** add the **Scope** block, and draw a wire from the **Sine Wave** to the **Scope** block

Note: To draw a wire, left-click source and drag mouse to destination

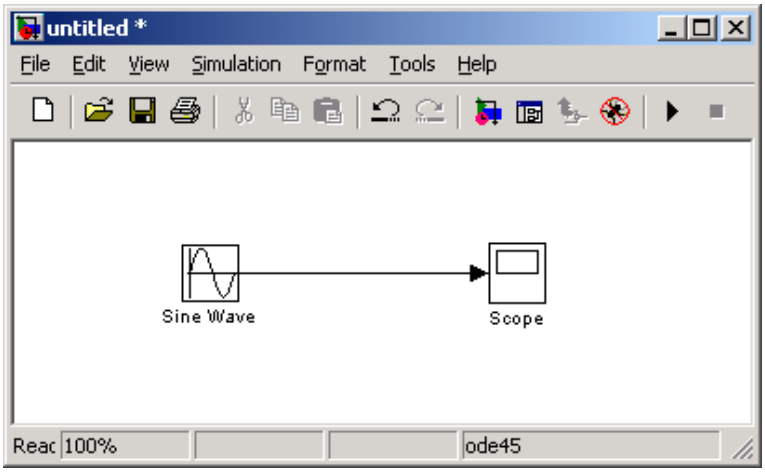


Figure 2–7. Wire the Elements.

Assign 2*pi*(1/150) frequency to the Sine Wave element, show port data types, and change **simulation parameter**'s **stop time** to **inf** (infinity)

❶ Double-click the **Sine Wave** block

The **Block Parameters** dialog box opens.

❷ Change the frequency to **2*pi*(1/150)** and click **OK** to close the dialog box

Figure 13–8. Change the Frequency.

❸ On the worksheet, go to **Format → Port/Signal Displays** and click **Port Data Types**

The signal width is displayed on the wire as shown in the following figure.

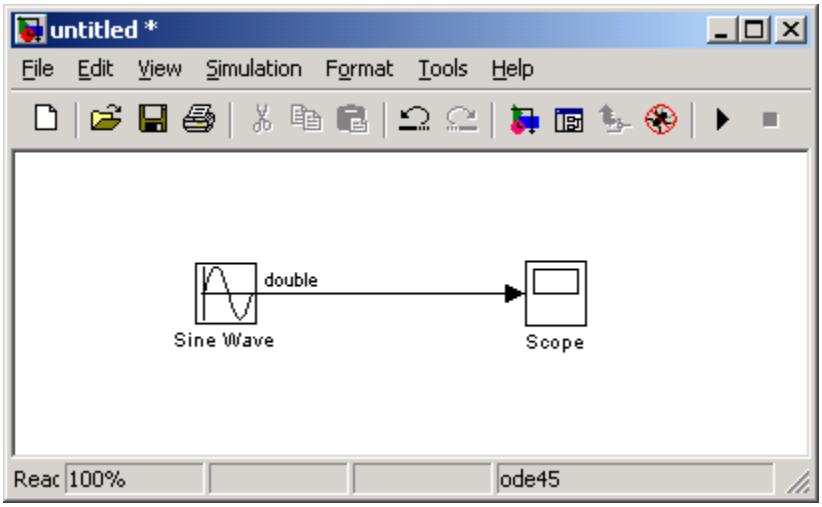


Figure 2–9. Double Precision.

❹ From your project sheet, pull down the **Simulation** menu and select **Configuration Parameters**

❺ From the **Simulation Parameters** dialog box, select **Solver** in the left hand window and change the stop time to **inf**, and click **OK**

This allows your simulation to run to infinity (until you manually stop the simulation).

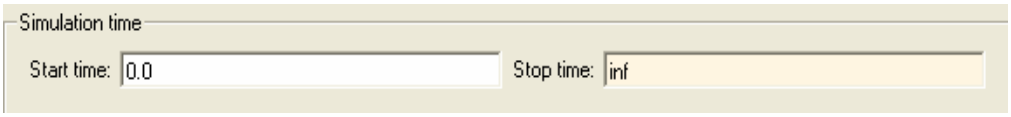


Figure 2–10. Simulation Parameters Dialog Box.

Parameterize the **Scope** block, and run the simulation

❶ Double-click the **Scope** block

❷ Click the **Scope Parameters** button

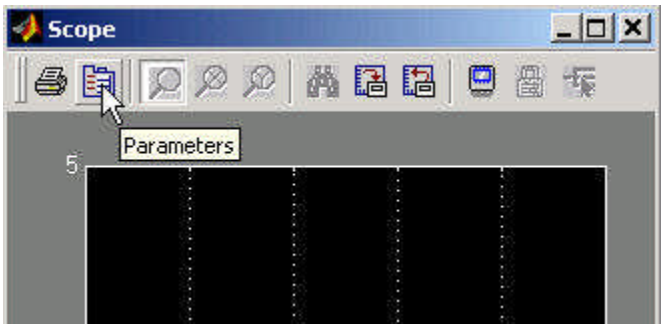


Figure 13–11. Scope Parameters Button.

❸ In the **Scope Parameters** box, set the time range to 500, and click OK

❹ Run the simulation: From your **Simulink** project worksheet, click **Start Simulation** button, or use **Simulation → Start**

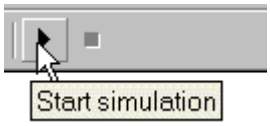


Figure 2–12. Simulation Button.

❺ On the **Scope** display, click the **Autoscale** button so the output will fit into the **Scope**

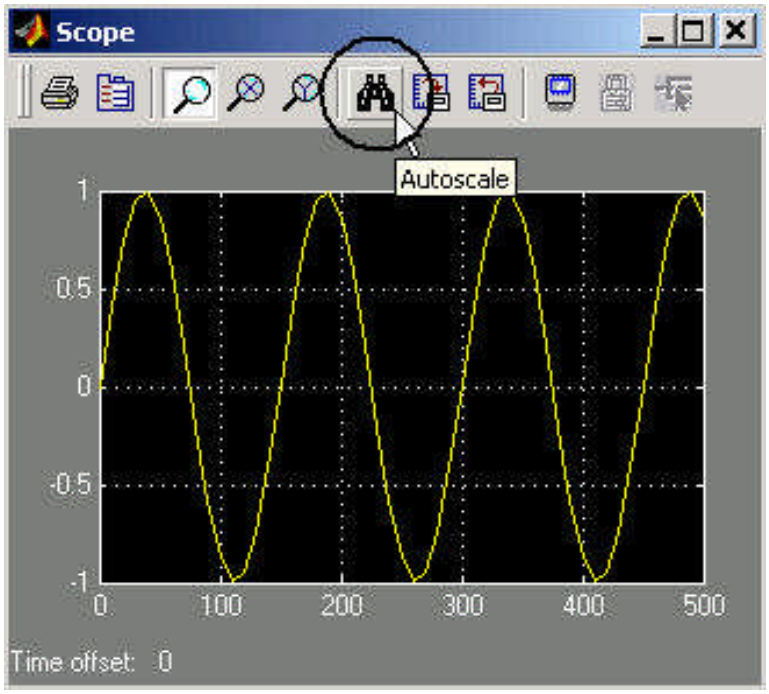


Figure 2–13. Autoscale Button.

❻ View the **Scope** output

A *smooth* sine wave should fit into your scope window. This is what you would expect because you are running a double-precision software simulation.

❼ Stop the simulation as shown below

Figure 2–14. Stop Simulation.

Introduce the Xilinx Gateways and MUX Blocks Step 2

Use the Xilinx **Gateway In**, **Gateway Out, System Generator,** and **MUX** blocks, as shown below, which provide interface to the Xilinx Blocksets in Simulink.

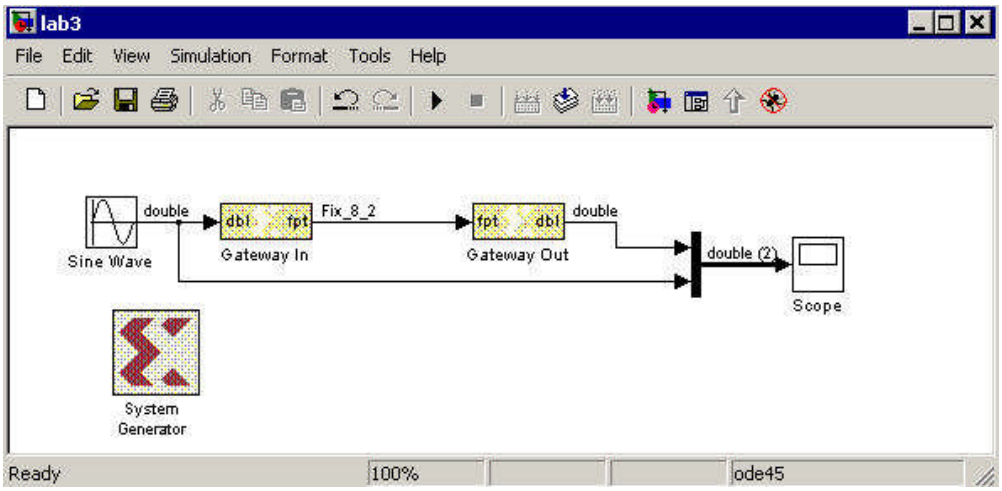


Figure 2–15. Gateway In and Out.

❶ Using **Simulation → Configuration Parameters** dialog box, set the stop time to **500**, and click **OK**

❷ From the **Xilinx Blockset** (in the **Simulink Library Browser**), open **Basic Elements** and drag the **Gateway In** block onto the design sheet. Drop it on the connection between the **Sine Wave** and the output **Scope**. It will automatically insert itself

❸ Double-click **Gateway In** to open the **Block Parameters**

❹ Set the **Number of bits** to **8** and **Binary Point** to **2**

❺ Similarly, drag a **Gateway Out** block onto the sheet, and drop it between the **Gateway In** block and the output **Scope** block

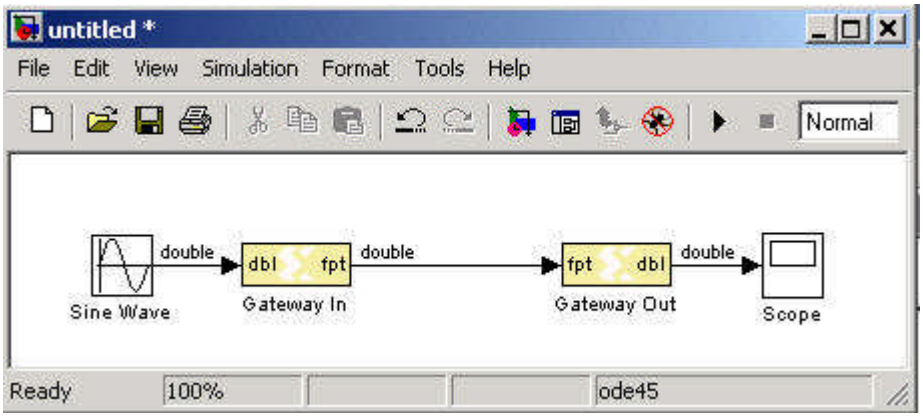


Figure 2–16. Drag Gateway In Block.

❻ Add a Simulink **MUX** between the **Gateway Out** and the **Scope** by using **Simulink → Signal Routing** (see Figure 2-15)

❼ Add an additional net between the **Sine Wave** and the **MUX**

Note: This will make the scope display both the double-precision sine wave and the user-defined precision sine wave that has gone into and back out of the Xilinx gateways

❽ Add a **system generator** token  from the **Xilinx Blockset → Basic Elements** library to the design

❾ To view the number of signals going into the **MUX**, select **Wide nonscalar lines**, **Signal dimensions**, and **Port data types** under the **Format → Port/Signal Displays** menu

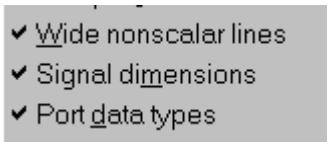


Figure 2–17. Format Menu.

❿ Update the diagram: select **Edit → Update Diagram**

Note: Now look at your port types. Notice that the gateway in block has changed the signals from double-precision to fixed-point types. Fixed-point looks like Fix_8_2 in this case.

Evaluate the Precision — Step 3

Run the simulation with default settings and understand the output. Change the quantization and overflow options one at a time, run the simulation for each

change, and analyze the simulation output. Change the sampling period from one to five and see effect on the quantization.

❶ Run the simulation and observe a jagged sine wave adjacent to the smooth sine wave that is not going through the Xilinx blocks, indicating quantization effects (the difference between the double precision floating point of MATLAB and the fixed point FIX_8_2 of the block)

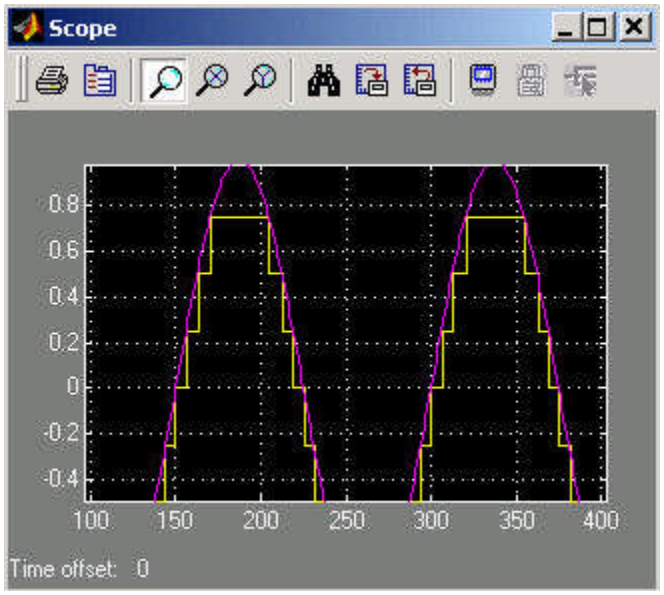


Figure 2–18. Quantization Effects.

❷ Double-click **Gateway In,** change the **Output Data Type** parameter to **Unsigned** from **2s complement**, and click **Apply**

❸ Run the simulation and observe the scope output as shown in Figure 13-19

Note: Because the value is unsigned, you have some overflow (the negative part of the sine wave). Because overflow is set to Wrap, the negative portion is wrapping.

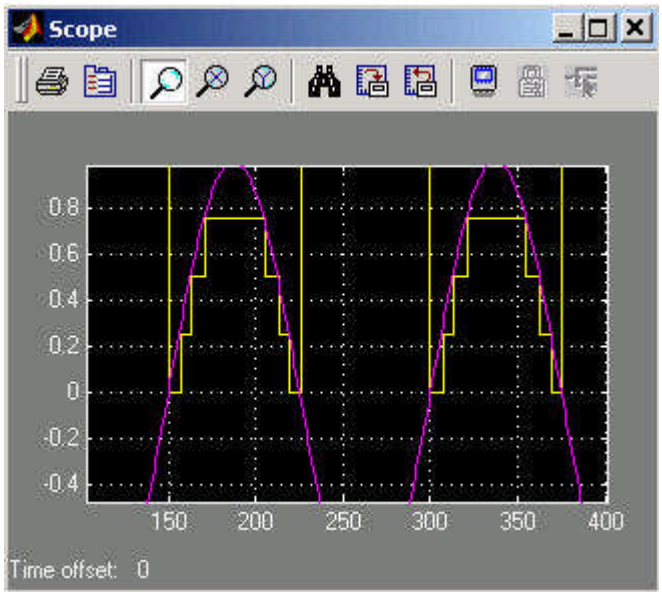


Figure 2–19. Unsigned Data, Wrap Overflow, Truncate Qantization Output.

❹ Change the **overflow** to **Saturate**, click **Apply**, and run the simulation

Note: The negative part of the sine wave is saturated to 0

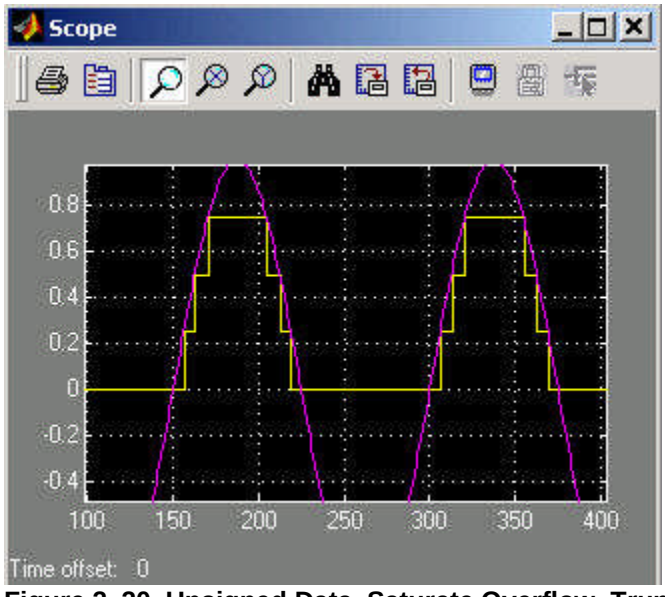


Figure 2–20. Unsigned Data, Saturate Overflow, Truncate Quantization Output.

❺ Change the data type to **2s complement**, **overflow** to **Saturate**, **quantization** to **Round**, click **Apply**, and run the simulation

Note: You will see the sine wave is rounded up to the peak value

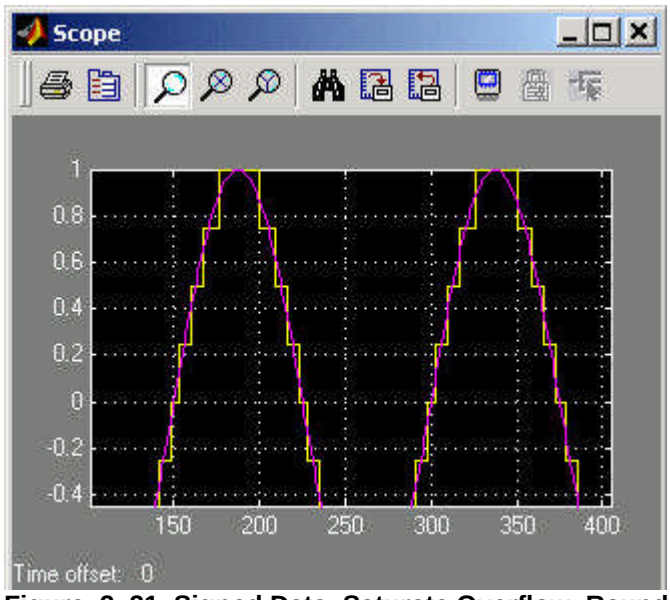


Figure 2–21. Signed Data, Saturate Overflow, Round Quantization Output.

❻ Change the **quantization** to **Truncate**, click **Apply**, and run the simulation

Note: Now, instead of rounding up, the effect of quantization is to truncate the peak value

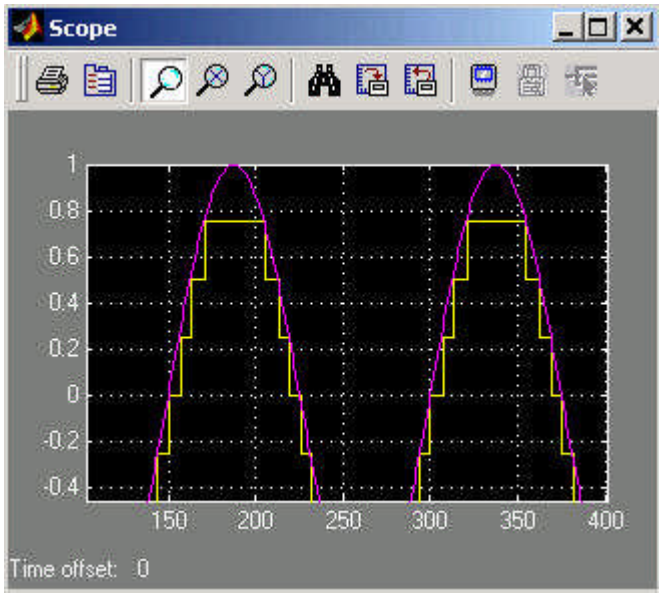


Figure 2–22. Signed Data, Saturate Overflow, Truncate Quantization Output.

❼ Reduce some of the quantization error by changing the **binary point** from **two** to **six**, click **apply**, and run the simulation

Note: You will see a smoother sine wave because more of the quantization error has been removed. The number of fractional bits was increased from two to six.

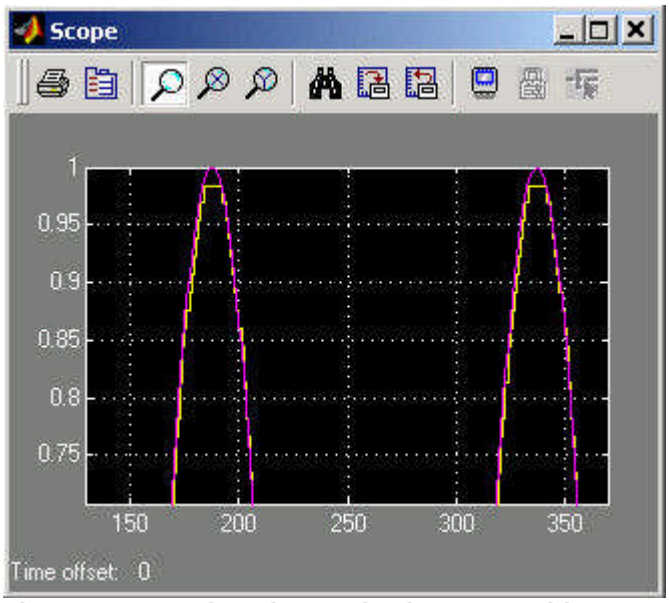


Figure 2–23. Reduced Quantization Error with Increased Number of Fractional Bits.

Analyze Sample Period — Step 4

Run the simulation with slower sampling period (sample period of 5) with the sine wave source and analyze the quantization effect.

1. Change the **sample period** from **one** to **five**, click **Apply**, and run the simulation

   Note: As the sample period is increased (that is, the sampling is fewer times), the quantization error is increased.

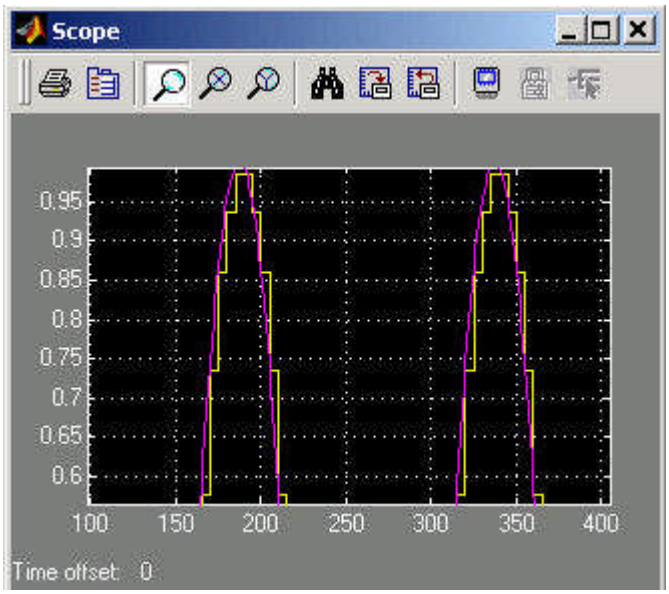


Figure 2–24. Signed Data, Saturate Overflow, Truncate Quantization Output.

Change the source to ramp function and analyze the sampling period (sample period of 10) effect on the quantization

❶ Replace the **sine wave** source with the **ramp** function from the **Simulink Sources**

Figure 2-25. Ramp Icon.

❷ Open the **Configuration Parameters** dialog box

❸ Change the **Stop Time** to 100, and click **OK**

❹ In the **Gateway In Block Parameters** dialog box, change the **Binary Point** to **0** and the **Sample Period** to **10**

❺ Click **Apply**, and run the simulation

Note: You will see the ramp is coarse due to slow sampling period

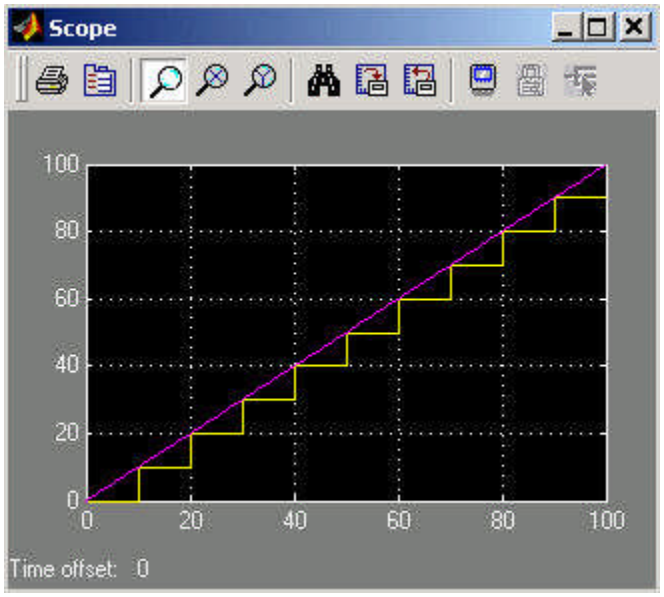


Figure 2-26. Ramp Source, Sampling Period 10.

❻ Change the **Sample Period** to **1** in the **Gateway In Block Parameters**, and click **Apply**

❼ Run the simulation

Note: You will see the ramp is smoother as sampling period has improved.

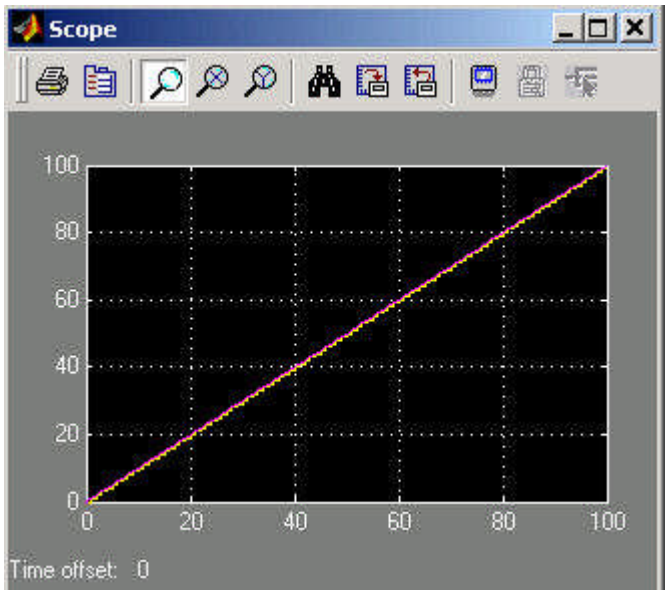


Figure 2-27. Ramp Source, Sampling Period 1.

❽ Close the worksheet

Note: You do not need to save the worksheet.

Design a 12 x 8 MAC Using Xilinx Blockset Step 5

Create a 12 x 8 MAC in Simulink with the following characteristics and basic structure shown in Figure 2-28:

- **Separate multiplier and accumulator**
- **Multiplier input data widths of 12-bits and 8-bits of signed data**
- **Multiplier output width of 20 bits**
- **Accumulator output width of 27-bits**
- **Use third input as reset signal (positive logic) for the accumulator**
- **Set multiplier block properties to implement in LUT**

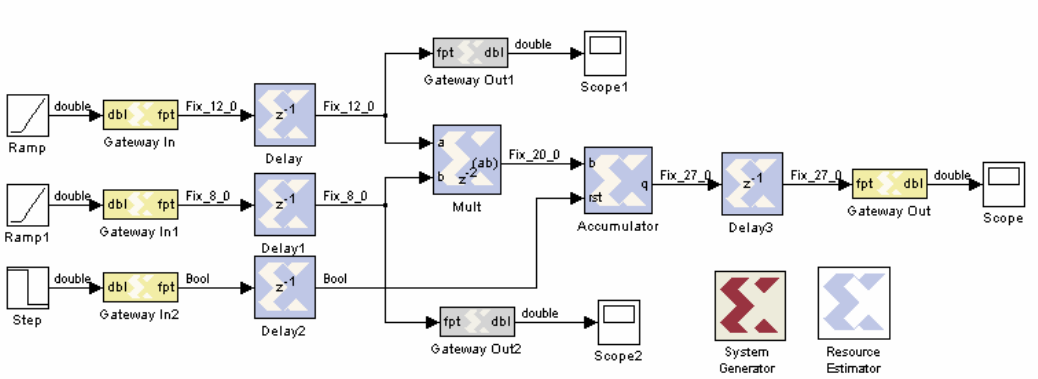


Figure 2-28. The 12 x 8 MAC Using Xilinx Blockset.

❶ In the project sheet, select **File → New → Model**

A new Simulink project opens.

❷ Add three inputs to the worksheet: the 12-bit input data (**ramp** function), the 8-bit input data (**ramp** function), and a third input that will be reset for the accumulator (**step** function)

❸ Change **step** function input to have **initial value** to **1** and **final value** to **0**

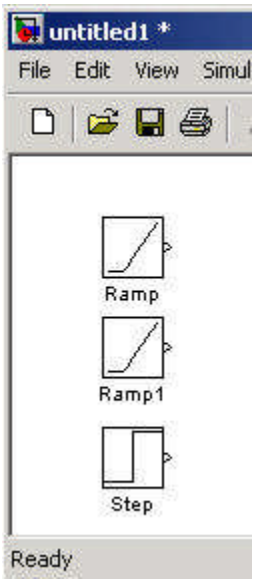


Figure 2-29. Input Sources for the MAC.

❹ Add Xilinx **Gateway In** blocks to the three inputs and wire them up

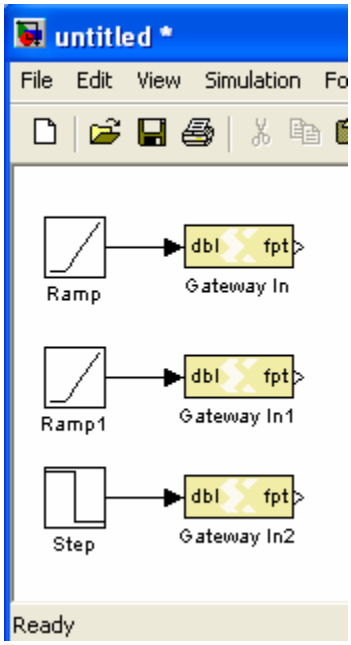


Figure 2-30. Gateway Ins for the Three Inputs.

❺ Add registers (**delay** block from the **Basic Elements** library of the Xilinx Blockset) to improve the performance. Note that registered inputs in the FPGA may improve timing.

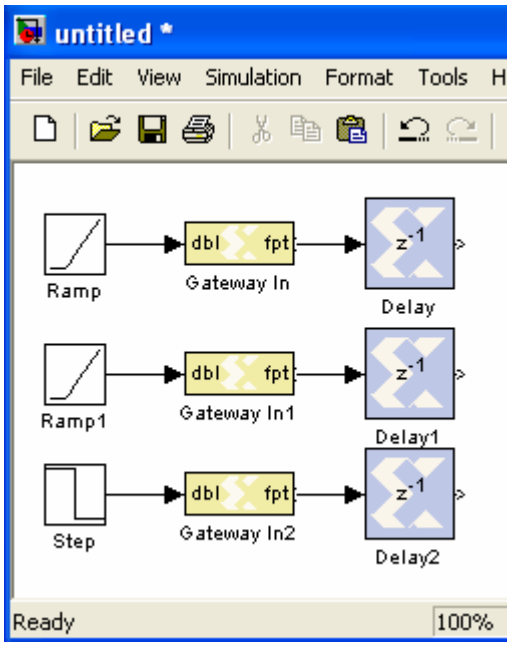


Figure 2-31. Add Registers for Performance.

❻ Add the **multiplier** and the **accumulator** blocks located in the Xilinx Blockset's **Math** library and wire them up

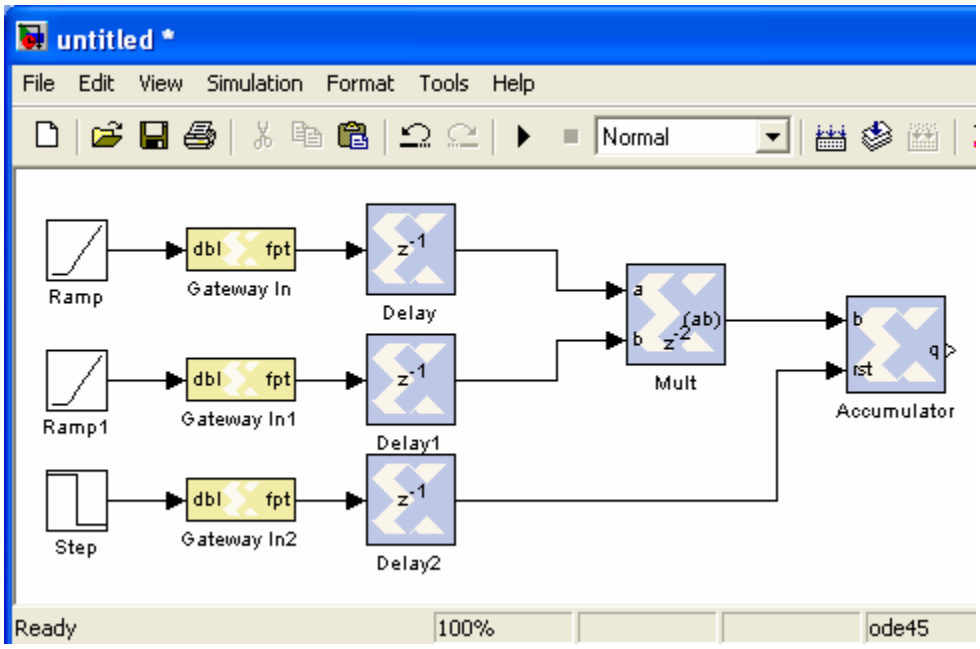


Figure 2-32. Add the Multiplier and the Accumulator.

❼ Add the output **delay** block at the output of the accumulator to improve the performance

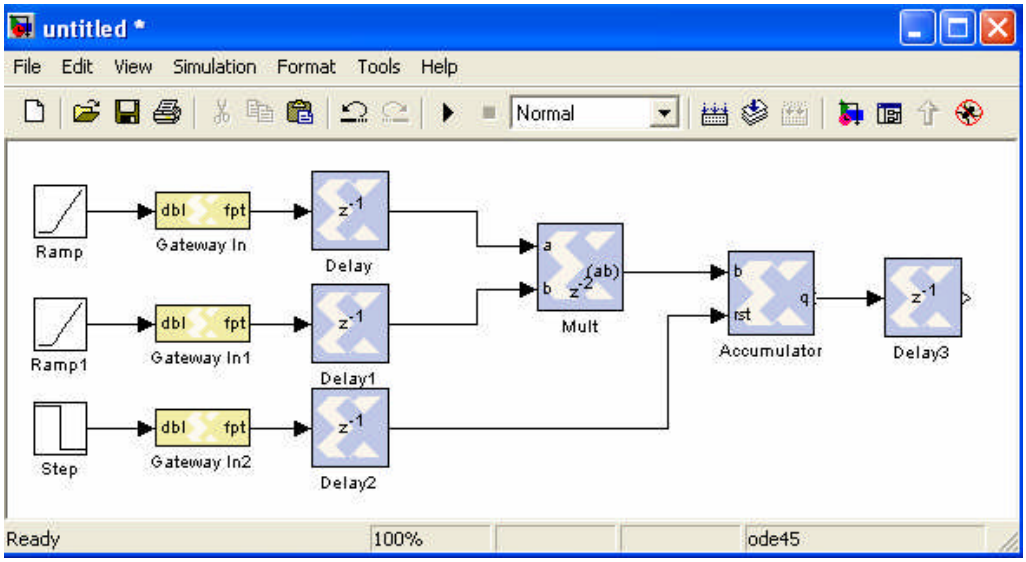


Figure 2-33. Add Output Delay for Performance.

❽ Add **Gateway Out** for the output and also for the ramp inputs (Ramp input, do not place it on the Step input) so you can monitor both input and output

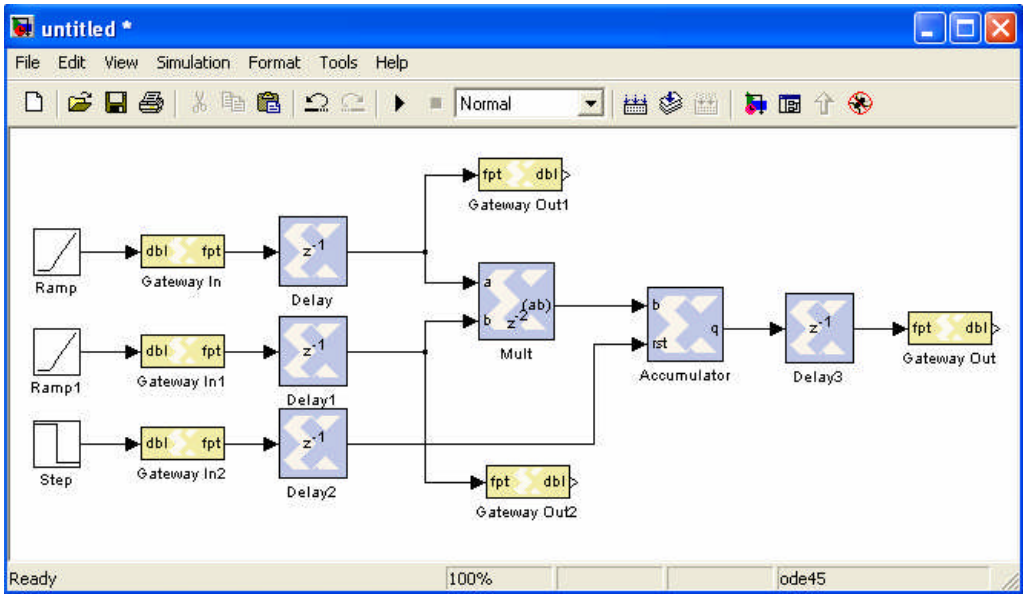


Figure 2-34. Add Gateway Outs.

❾ Add **Scope** to the **Gateway Out** blocks to watch in oscilloscope form

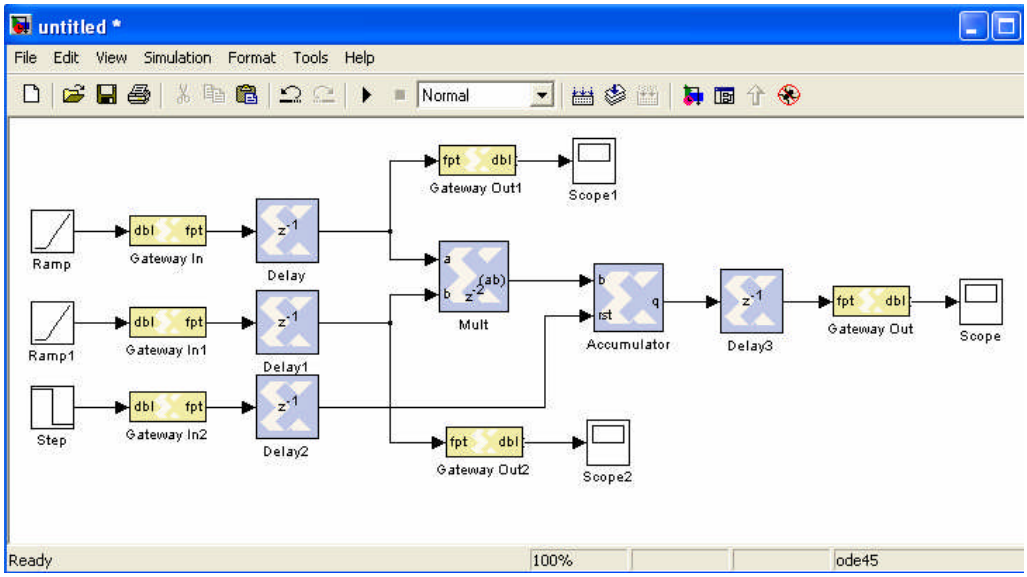


Figure 2-35. Input Sources for the MAC.

❿ Wire all the blocks and add **System Generator** block.

❿ Update the latency of the reset delay block to 3 to make sure that the accumulator is reset after each output sample is generated.

The final diagram should look like the figure below.

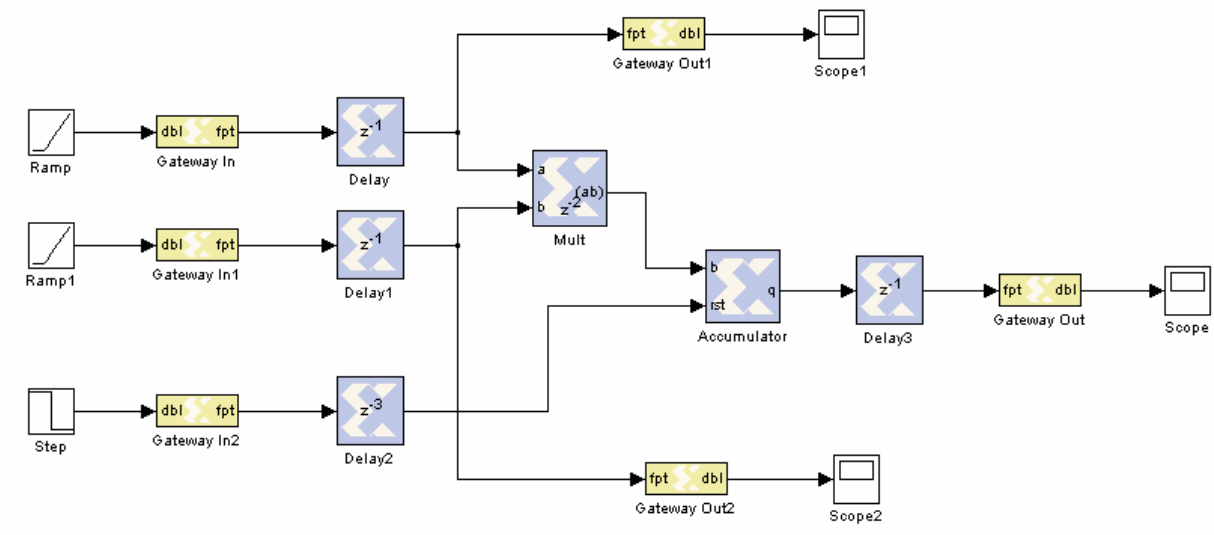


Figure 2-36. The 12 x 8 MAC Using Xilinx Blockset.

Simulate the 12 x 8 MAC Using Simulink Step 6

Configure the inputs and simulation parameters as specified below. Next, simulate the 12 x 8 MAC in Simulink and verify the design functionality

- **12-bit input: signed data, binary point 0, overflow method saturate, and sampling period 1**
- **8-bit input: signed data, binary point 0, overflow method saturate, and sampling period 1**
- **3rd input: Boolean type**
- **Gateway Out**
  - **Accumulator: "Translate in to output port" box checked**
  - **Ramp sources: "Translate in to output port" box unchecked**
- **Multiplier: parameter set to Pipeline to Greatest Extent Possible**
- **Accumulator: output width of 27 bits, overflow method to wrap**
- **Simulation parameters: stop time to 2500**

❶ Configure the 12 and 8-bit inputs by double-clicking each of the **Gateway In** blocks to open their **Parameters** dialog boxes

❷ For the first input (12-bit) set **Data Type** to **Signed Data**, **Number of bits** to **12**, **Binary Point** to **0**, **Sampling Period** to **1**, **Overflow** method to **Saturate**

❸ For the second input (8-bit) set **Data Type** to **Signed Data**, **Number of bits** to **8**, **Binary Point** to **0**, **Sampling Period** to **1**, **Overflow** method to **Saturate**

❹ For the third input set the **Data Type** to be **Boolean**

❺ Double-click the **Gateway Out** blocks connected to the multiplier input (for monitoring purpose) and **uncheck** the **Translate into output port** box. The box should turn gray if done correctly

❻ Configure the multiplier block: Double-click the **Multiplier** block, **check** the **Pipeline to Greatest Extent Possible** box, **uncheck** the **Use Embedded Multipliers** box so the multiplier will be implemented with LUTs, and set the **latency** to 2.

❼ Configure the accumulator to the same bit widths required by the specifications (27-bits), and set the **Overflow** method to **Wrap**

❽ Open the **Configuration Parameters** dialog and set the **Stop Time** to **2500**

❾ Run the simulation

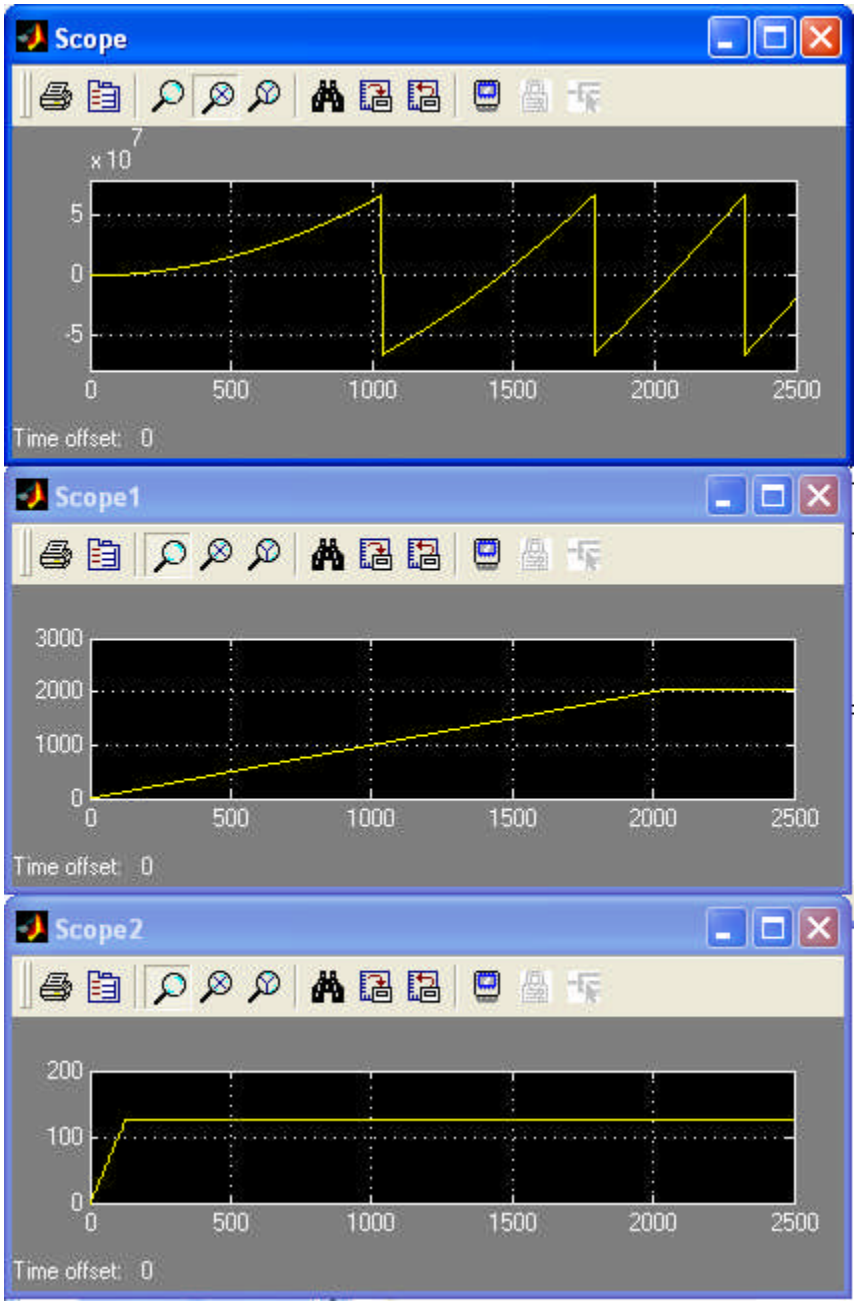


Figure 2-37. Simulink Output with Multiplier Having Maximum Pipelining.

❿ Zoom in around first transition of the output and determine where it occurs

? 1. At what time the first transition (sharp) occurs? _____

Change the multiplier block latency to 3 and re-simulate. Compare the results and save the design as mac_fir_test

❶ Open the multiplier's **Block Parameters** dialog box, and change the **latency** to **3**

❷ Click **OK**

❸ Run the simulation

2. At what time the first transition (sharp) occurs now? _____

❹ Save the design with the name **mac_fir_test**

Resource Estimations Step 7

Estimate the resources used by the design through the **Resource Estimator** block with **multiplier** block implemented in **LUTs** with **latency** set to **2** and **3** for the following target:

- Compilation: **HDL Netlist**
- Part: **Virtex2p xc2vp30-7ff896**

❶ Double-click the **System Generator** icon and Select **HDL Netlist** for Compilation and set the **Part** related fields as:

- Compilation: **HDL Netlist**
- **Part:** Virtex2p xc2vp30-7ff896

❷ Make sure that the latency of the **multiplier** block is set to **2** and check **Use Placement Information for Core** check-box

❸ Check the **Use Placement Information for Core** check-box of the **accumulator** block

❹ Add the **Resource Estimator** block from Xilinx Blockset's **Index** set

❺ Double-click the **Resource Estimator** block

The following GUI appears.

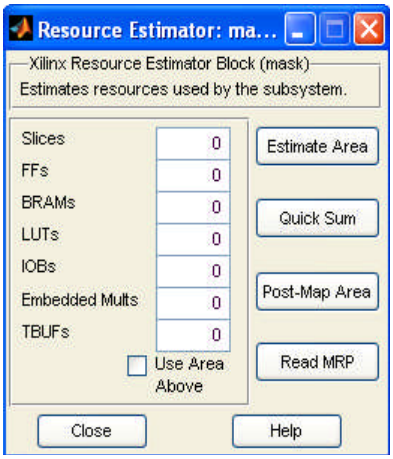


Figure 2-38. Resource Estimator Block.

❻ Click the **Estimate Area** button

3. What is the resource estimation for the design using the multiplier implemented using LUTs and latency of 2?

Number of Slices _____
Number of FFs _____
Number of LUTS _____
Number of IOBs _____

❼ Change the latency of the **multiplier** block to **3** in the multiplier block parameter and click the **Apply** button

4. What is the resource estimation for the design using the multiplier implemented using LUTs and latency of 3?

Number of Slices _____
Number of FFs _____
Number of LUTS _____
Number of IOBs _____

Estimate the resources used by the design through the **Resource Estimator** block with **embedded multiplier** and **latency** set to **2** and **3**

❷ Set the **latency** of the **multiplier** block to **2** in the multiplier block parameters GUI

❷ Click the **Use Embedded Multipliers** check-box to select the embedded multiplier

❸ Click the **Apply** button

❹ Click the **Estimate Area** button

5. What is the resource estimation for the design using the embedded multiplier with latency of 2?

Number of Slices	_____
Number of FFs	_____
Number of LUTS	_____
Number of Multiplier	_____
Number of IOBs	_____

❺ Change the latency of the **multiplier** block to **3** in the multiplier block parameter and click the **Apply** button

6. What is the resource estimation for the design using the embedded multiplier with latency of 3?

Number of Slices	_____
Number of FFs	_____
Number of LUTS	_____
Number of Multiplier	_____
Number of IOBs	_____

❻ Click the **Use Embedded Multiplier** check-box to **de-select** the embedded multiplier and implement it in LUTS

❼ Make sure that the **latency** is set **2**

❽ Save the design

Code Generation Using System Generator — Step 8

Generate the VHDL code for the design, with multiplier having latency of 2 and implemented in LUTs, using the System Generator block with the following target information

- **Compilation: HDL Netlist**
- **Part: Virtex2p xc2vp30-7ff896**
- **Synthesis Tool: XST**
- **Target Directory: c:\xup\dsp_flow\labs\lab2\ise**

- **Create Testbench:** Checked
- **FPGA System Clock Period (ns): 10**

❶ Make sure the latency of the **multiplier** block is set to **2** and the embedded multiplier usage is **un-checked**

❷ Double-click the **System Generator** icon

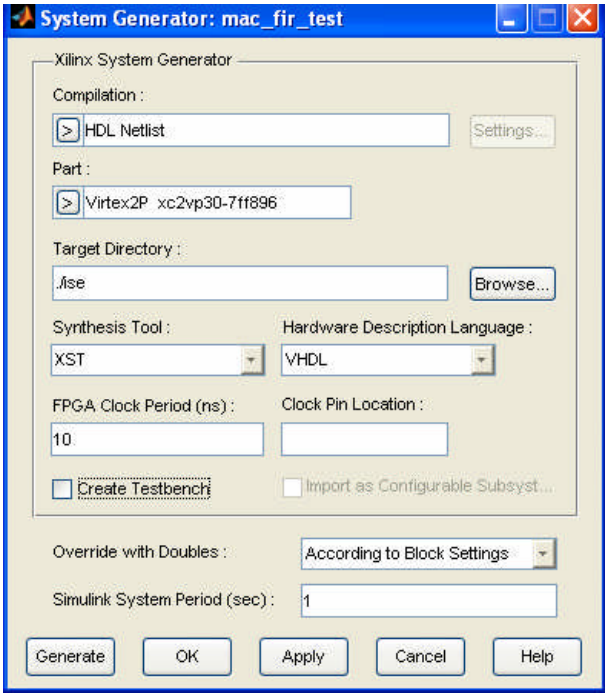


Figure 2-39. System Generator Parameters for Generating VHDL Code.

❸ Specify **c:\xup\dsp_flow\labs\lab2\ise** as the **Target Directory** by entering **./ise** in the Target Directory field.

❹ Select **HDL Netlist** for Compilation and set the **Part** related fields as:

- **Compilation: HDL Netlist**
- **Part: Virtex2p xc2vp30-7ff896**
- **Synthesis Tool: XST**

❺ Make sure that the **Create Testbench** check-box is **not checked**

❻ Enter **10** for the **FPGA System Clock Period (ns)** and click **Apply.** This translates into a clock period constraint for the ISE Foundation software.

❼ Click **Generate**

Note: A warning message window may pop up asking if you want to continue generating testbench and design files for the simulation run-time of 2500. If so, click **Yes**.

Note: This will create VHDL files possibly using the CORE Generator system in the target directory. It will also have the top-level VHDL file (mac_fir_test_clk_wrapper.vhd), the generated multiplier core (mac_fir_test_mult.vhd) and the accumulator core (mac_fir_test_accumulator.vhd), the testbench (mac_fir_test_clk_wrapper_testbench.vhd), the project file (mac_fir_test_clk_wrapper.ise), and several project files and MTI simulation files.

❽ Open the generated project in the ISE software by double-clicking on **mac_fir_test_clk_wrapper.ise** in the project directory

❾ Highlight the top-level file, called **mac_fir_test_clk_wrapper.vhd**, and double-click on **Implement Design**.

7. Open the Place and Route report file and fill in the following information

Number of Slices: _____
Number of BUFGMUXs: _____
Number of External IOBs: _____

8. Open the post-Place and Route Timing report and fill in the following information

Maximum clock frequency: _____

On-Chip Verification with Chipscope-Pro (optional) Step 9

Modify the design by adding input stimulus, reset & enable signals, and a Chipscope block to the design as illustrated in Figure 2-41. Create the stimulus by adding counters that feed the inputs of the MAC. Push the multiplier, accumulator and output delay into a sub-system and name it **MAC**. Add the reset and enable signals.

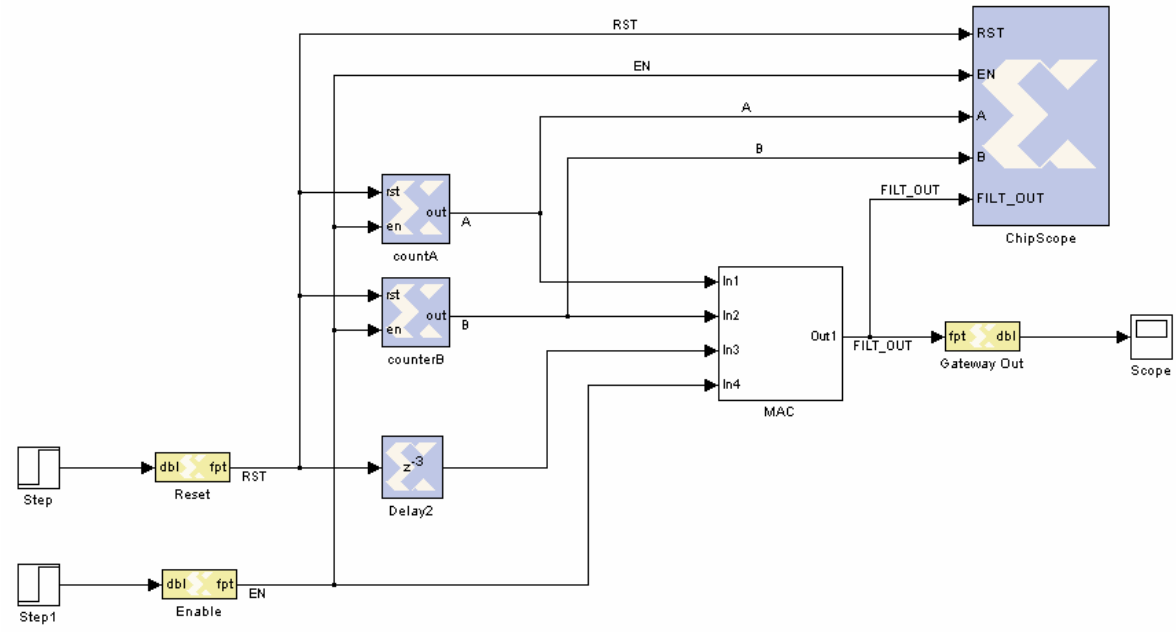


Figure 2-41. MAC unit with Chipscope Block

❶ Drag-and-Drop a counter into the design and specify the following parameters, and rename to **countA**.

- Counter Type: Count Limited
- Number of Bits: 12
- Binary Point Position: 0
- Arithmetic Type: Signed (2's complement)
- Initial Value: 0
- Count to Value: inf
- Step: 1
- Count Direction: Up
- Provide Reset Port: Checked
- Provide Enable Port: Checked

❷ Drag-and-Drop a second counter into the design and specify the following parameters, and rename to **countB**.

- Counter Type: Count Limited
- Number of Bits: 8
- Binary Point Position: 0
- Arithmetic Type: Signed (2's complement)
- Initial Value: 0
- Count to Value: inf
- Step: 1
- Count Direction: Up
- Provide Reset Port: Checked
- Provide Enable Port: Checked

❸ Connect the **countA** and **countB** counters to the inputs of the MAC, respectively. Your design should look similar to that of the figure below.

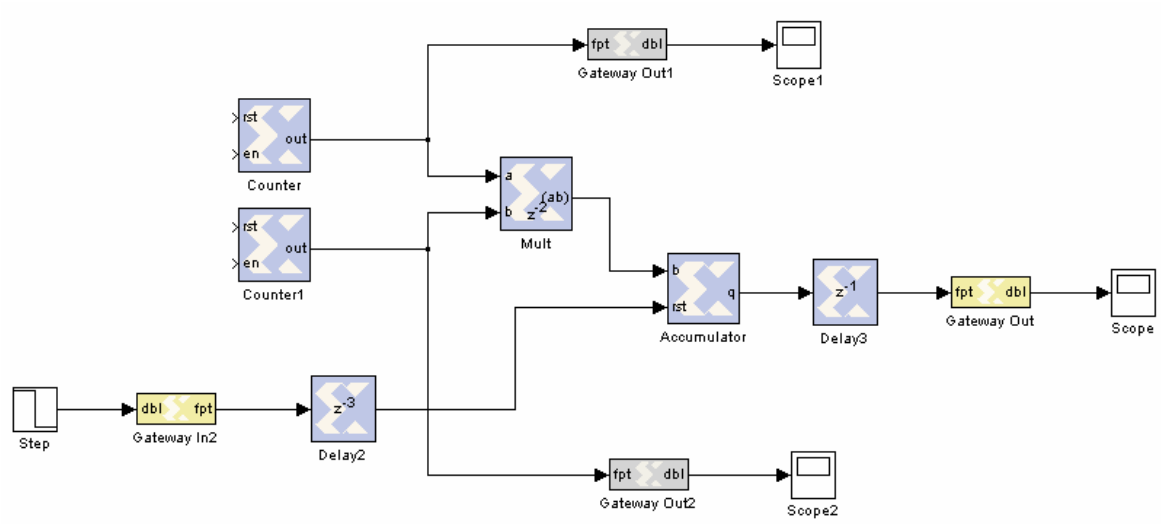


Figure 2-42. Adding the Input Stimulus

❹ Add enable ports to both the multiplier and accumulator by double-clicking on each block and clicking so that a check appears next to the Provide Enable Port option.

❺ Add a new gateway in block of type **boolean**, rename it as **Enable**, and connect it to the enable ports of the blocks in the design.

❽ Add a Simulink step function and connect it to the input of the **Enable** gateway in block.

❼ Rename the gateway in block that connects to the reset port of the accumulator as **Reset** and connect it to the remaining blocks that have reset ports.

❺ Drag the cursor around the multiplier, accumulator, and output delay so that they are selected, and create a subsystem called **MAC**. At this point, your design should look similar to the figure below.

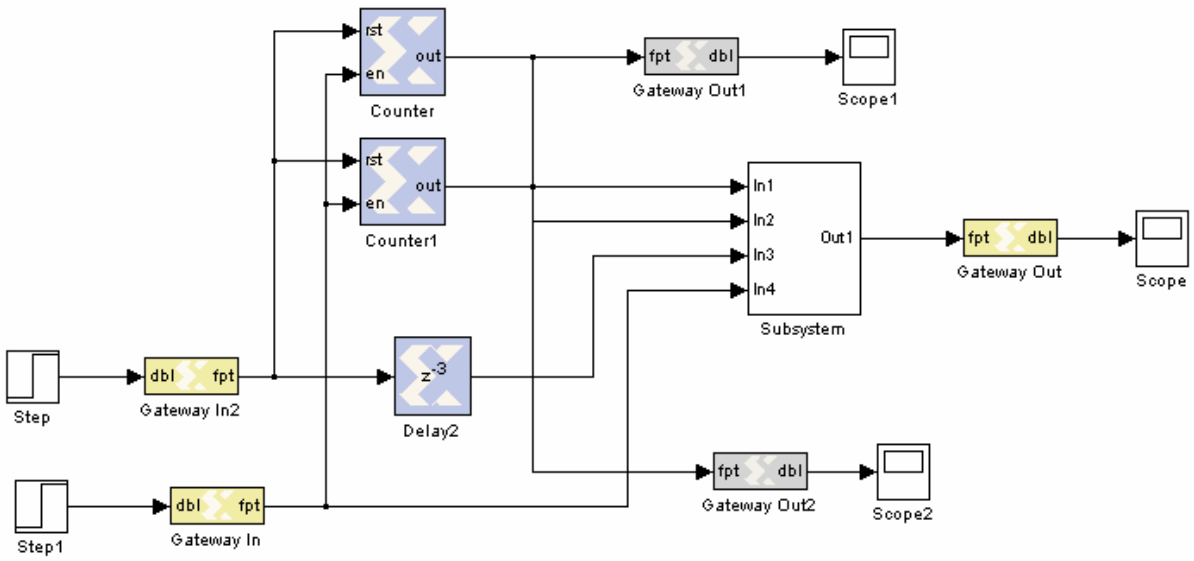


Figure 2-43. Connecting the Enable and Reset Signals

Add the Chipscope block to the schematic, specify parameters to setup the data and trigger, and connect the Chipscope block to the design.

❶ Drag the **Chipscope** block from the **Tools** library to the schematic sheet and double-click to specify the following parameters
- Number of Trigger Ports: 2
- Display Options for Trigger Port 0
  - Number of Match Units: 1
  - Match Type: Extended
- Display Options for Trigger Port 1
  - Number of Match Units: 1
  - Match Type: Extended
- Number of Data Ports: 3
- Depth of Capture Buffer: 2048

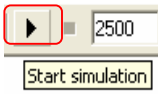
❷ Make the following connections to connect the Chipscope block to the design
- **Reset_Delay2** to the **trig0** port, name wire connection as **RST**
- **Enable_Delay2** to the **trig1** port, name wire connection as **EN**
- **CountA** to **Data0** port, name wire connection as **A**
- **CountB** to **Data1** port, name wire connection as **B**
- Output of **MAC** to **Data3** port, name wire connection as **FILT_OUT**

❽ Go to **edit → update diagram** and notice how the ports of the Chipscope block will be updated to reflect the name of the signals connected to it.

These signal names will be preserved in the Chipscope Analyzer.

❾ Save the design as **mac_fir_test_cs.mdl**

Simulate the design to verify MAC operation. Next, you will specify the pinouts for the reset, enable, and clock inputs according to the XUP Virtex-II Pro board. Finally, you will generate the HDL code that contains the Chipscope ILA and ICON cores.

❶ Click the 

button to run the simulation. You should see the following results

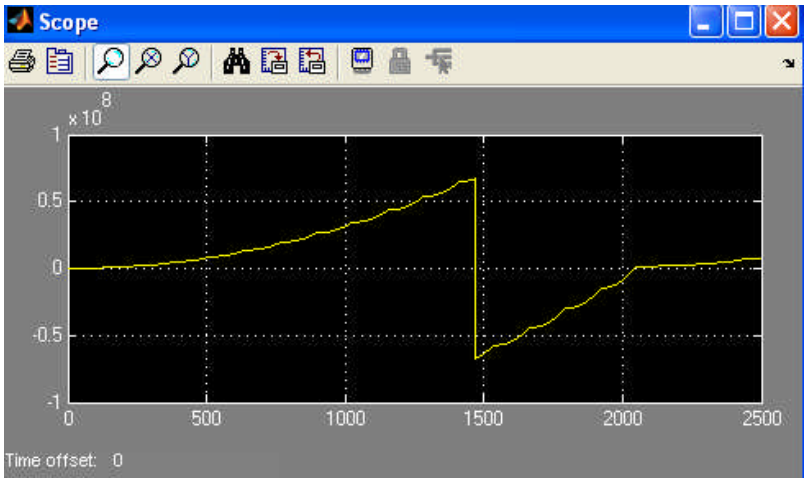


Figure 2-44. Accumulator Output

❷ Double-click the **Reset** gateway in block and click next to **Specify IOB Location Constraints** so that a check mark appears, and then enter **AG5** (center push button 0 on XUP Virtex-II Pro board) as the I/O location.

❸ Similarly, enter the I/O location of **AC11** for the **Enable** input, which corresponds to switch 0 on the XUP Virtex-II Pro board.

❹ Double-click the System Generator token and specify the following parameters and then click generate

- Compilation: **HDL Netlist**
- Part: **Virtex2p xc2vp30-7FF896**
- Target directory: **./cs**
- FPGA Clock: **10 ns** (50 MHz oscillator on board)
- Clock pin location: **AJ15** (100 MHz system clock connected to pin AJ15 of Virtex-II Pro device)
- Create Testbench: **unchecked**

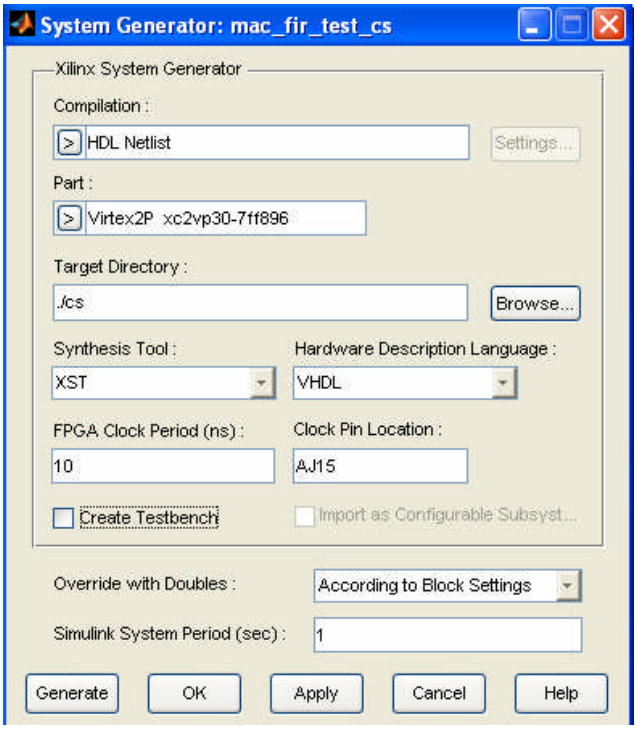


Figure 2-45. System Generator token parameters

Using the Windows Explorer, browse to the target directory (**c:\xup\dsp_flow\labs\lab2\cs**) and double-click on the project file (**mac_fir_test_cs_clk_wrapper.ise**) to open the ISE 7 Project Navigator. Simulate and implement the design

❶ Using the windows explorer, browse to the target directory (**c:\xup\dsp_flow\labs\lab2\cs**)

❷ Double-click the project (**mac_fir_test_cs_clk_wrapper.ise**) file

The ISE Project Navigator opens, and it will read the System Generator project. The System Generator project is already set up to run your on-chip verification with the Chipscope Analyzer.

You have many options within Project Navigator for working on your project. You can open any of the Xilinx software tools: the FPGA Editor, Constraints Editor, report viewers, etc.

For now, you will only instruct Project Navigator to run your design all the way from synthesis to bitstream, resulting in the invocation of Chipscope Analyzer.

❻ In the **Sources in Project** window, select the top-level VHDL module (**mac_fir_test_cs_clk_wrapper.vhd**)

❼ In the **Processes for Current Source** window, right-click **Analyze Design Using Chipscope** and select **Run**

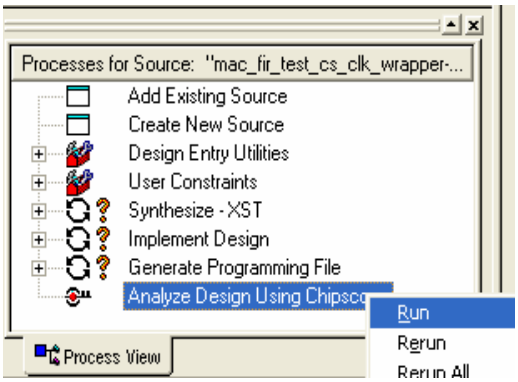


Figure 2–46. Select Run.

In the Messages console window, you will see that Project Navigator is synthesizing, translating, mapping, routing, and generating a bitstream for your design.

If you wish to examine any of the intermediate reports or tools used in implementing your design, you can return to the Processes window and select any of the completed reports or tools. For example, to see how your design was placed on a Xilinx FPGA, you can select the FPGA Editor view, underneath the Place & Route option in the Processes window.

? 9. Open the Place and Route report file and fill in the following information

Number of Slices: _____
Number of BUFGMUXs: _____
Number of External IOBs: _____

? 10. Open the post-Place and Route Timing report and fill in the following information

Maximum clock frequency: _____

Connect up the JTAG download cable and power on the Spartan-3 board. In Chipscope Analyzer, import the Chipscope project file that contains the grouped data signals as specified in the Simulink design. Next, you will specify values for the match units and trigger Conditions to capture data when reset is '0' and the counter that feeds the A input of the MAC is 0x000. Finally, you will perform on-chip verification.

❶ Connect the JTAG download cable and power up the board

❷ Click the Open Cable/Search JTAG Chain button 

❸ Chipscope-Pro Analyzer will automatically detect two devices in the JTAG Chain. Click <OK>.

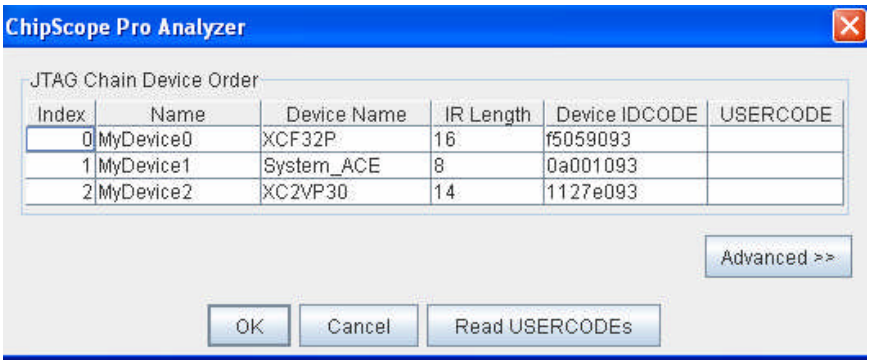


Figure 2-47. Devices Detected in JTAG Chain

❹ Right-click on the xc2vp30 device, select configure, and configure the device using the bitstream file generated in the project directory called **mac_fir_test_cs_clk_wrapper.bit**.

❺ Go to **File → Import**, browse to **…/cs/temp** directory and select **mac_fir_test_cs_chipscope.cdc** as the Import File. Click <OK>

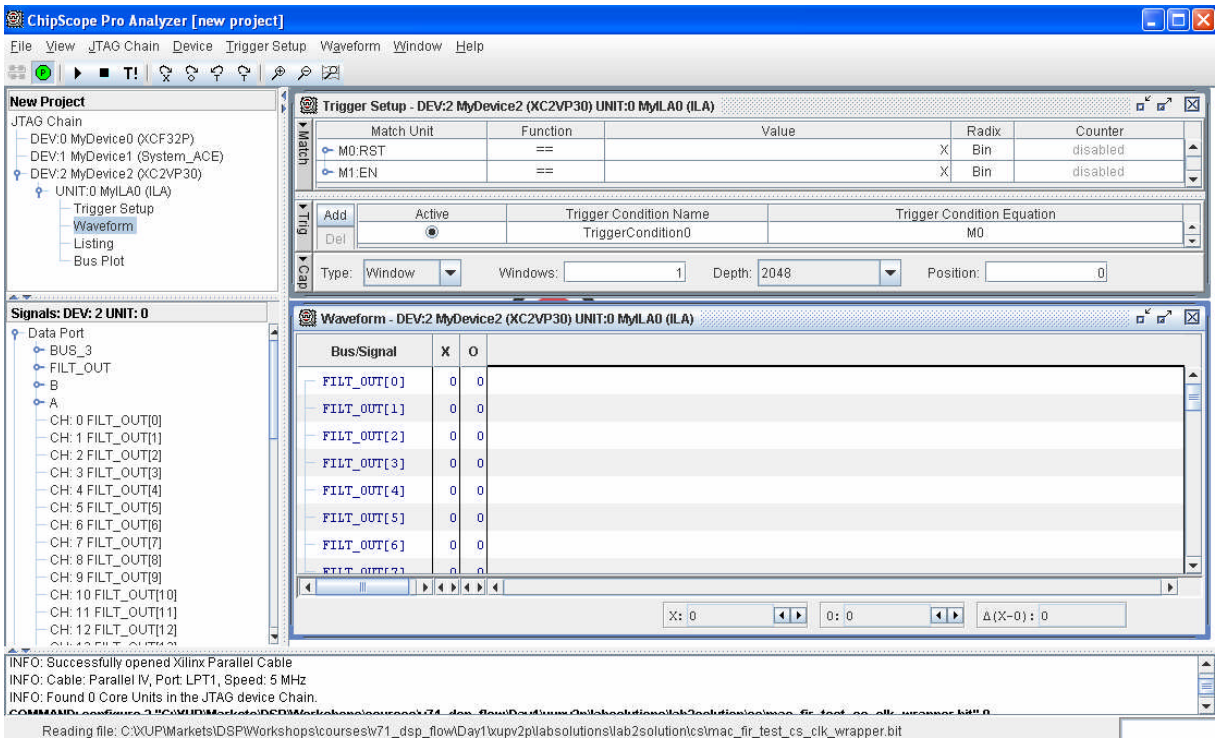


Figure 2-48. Chipscope Analyzer Project Generated from System Generator

Note how the port names indicated in the signal list and match units match those specified in the System Generator design. The buffer depth has also been set to 2048, which is in accordance with the parameters specified for the Chipscope block. In addition, the radix for the busses are also set to signed format, which is in accordance with the design specifications.

❺ Set the value fields of the Match Units to **0** for **RST** and **1** for **EN**, and the trigger condition equation to **M0 && M1**.

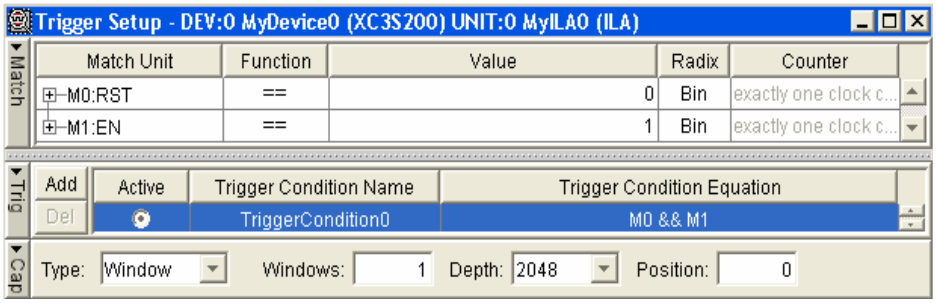


Figure 2-49. Specify Match Unit Values and Trigger Condition Equation

❻ Click the Apply Settings and Arm Trigger button 

❼ Push the Switch 0 on the XUP Virtex-II Pro board to the on position and press the center push button

❽ Double-click **Bus Plot** under the **New Project** pane and click to put a check mark next to **FILT_OUT** under the **Bus Selection** options.

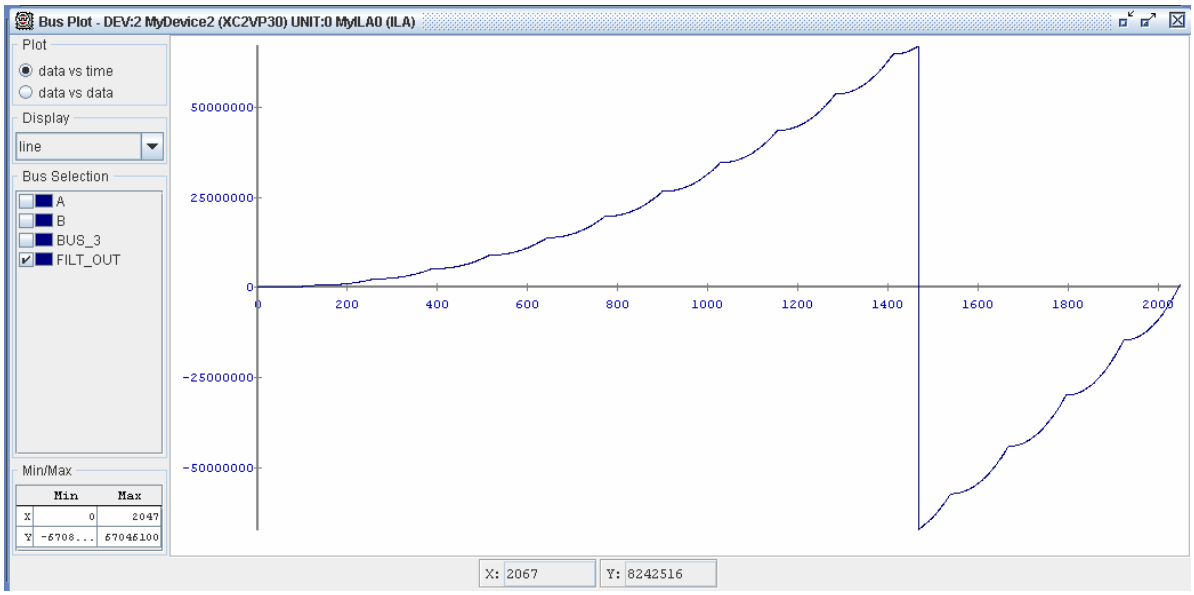


Figure 2-50. On-Chip Verification of MAC

Notice how the results of the on-chip verification are identical to the results of the Simulink simulation. You may optionally view the captured data for the **A** and **B** inputs of the MAC unit.

Conclusion

In this lab, you learned the basic design flow involved in incorporating the System Generator blockset in Simulink. You verified the design using MATLAB simulator through the Simulink environment and ran the System Generator token to generate VHDL code and the Xilinx CORE generator cores. Once the design was verified using Simulink, you were able to implement the design, view how the design was implemented, and generate the configuration file.

Answers

1. At what time the first transition (sharp) occurs? 1036

2. At what time the first transition (sharp) occurs now? 1037

3. What is the resource estimation for the design using the multiplier implemented using LUTs and latency of 2?

Number of Slices	93
Number of FFs	164
Number of LUTS	132
Number of IOBs	48

4. What is the resource estimation for the design using the multiplier implemented using LUTs and latency of 3?

Number of Slices	94
Number of FFs	184
Number of LUTS	132
Number of IOBs	48

5. What is the resource estimation for the design using the embedded multiplier and latency of 2?

Number of Slices	49
Number of FFs	95
Number of LUTS	48
Number of Multiplier	1
Number of IOBs	48

6. What is the resource estimation for the design using the embedded multiplier and latency of 3?

Number of Slices	59
Number of FFs	115
Number of LUTS	48
Number of Multiplier	1
Number of IOBs	48

7. Open the Place and Route report file and fill in the following information

Number of Slices:	70
Number of BUFGMUXs:	1
Number of External IOBs:	49

8. Open the post-Place and Route Timing report and fill in the following information

Maximum clock frequency: ~146 MHz

9. Open the Place and Route report file and fill in the following information

Number of Slices:	365
Number of BUFGMUXs:	2
Number of external IOBs:	30

10. Open the post-Place and Route Timing report and fill in the following information

Maximum clock frequency:	~146 MHz