Tutorial - Using Xilinx System Generator 14.6 for Co-Simulation on Digilent NEXYS3 (Spartan-6) Board

Shawki Areibi

August 13, 2023

1 Introduction

Xilinx System Generator provides a set of Simulink blocks (models) for several hardware operations that could be implemented on various Xilinx FPGAs. These blocks can be used to simulate the functionality of the hardware system using Simulink environment. The nature of most DSP applications requires **floating point format** for data representation. While this is easy to implement on several computer systems running high level modeling software such as Simulink, it is more challenging in the hardware world due to the complexity of the implementation of floating point arithmetic. These challenges increase with portable DSP systems where more restricting constraints are applied to the system design. For these reasons Xilinx System Generator uses **fixed point format** to represent all numerical values in the system. System generator provides some blocks to transform data provided from the software side of the simulation environment (in our case it is Simulink) and the hardware side (System Generator blocks). This is an important concept to understand during the design process using Xilinx System Generator.

1.1 Objectives

This tutorial will demonstrate the process of creating a simple DSP system using Xilinx System Generator 14.6 The System Generator runs within the Simulink simulation environment which is part of MATLAB mathematical package. In this tutorial a simple DSP system will be simulated using Simulink and then a Co-simulation is performed using NEXYS3 (Spartan-6) Board. Co-simulation integrates Simulink simulation capabilities with a hardware implementation to verify the functionality of the system.

The following steps are described in this tutorial:

- Starting System Generator with MATLAB.
- Creating a DSP system using Simulink and System Generator.
- Simulating the DSP system using Simulink.
- Preparing System Generator for Co-Simulation on NEXYS3 (Spartan-6) Board.
- Performing Hardware/Software Co-Simulation for the DSP system.

1.2 System Requirements

You must have the following software installed on your PC to complete this tutorial:

- Windows 7 OS.

- ISE 14.6
- System Generator 14.6
- MATLAB R2012a with Simulink.

Besides the following is required to complete this tutorial:

- Familiarity with Simulink Simulation Environment with MATLAB.
- Familiarity with Xilinx ISE and FPGA design flow.
- NEXYS3 (Spartan-6) Board and Digilent USB JTAG download cable

2 Starting System Generator

- To Start Xilinx System Generator, select ***Start*** → ***All Programs*** → ***Development*** → ***Xilinx*** → ***ISE Design Suite 14.6*** → ***System Generator***→ ***Xilinx System Generator 14.6***.
- This will start MATLAB and Simulink simulation environment as shown in Figure 1.

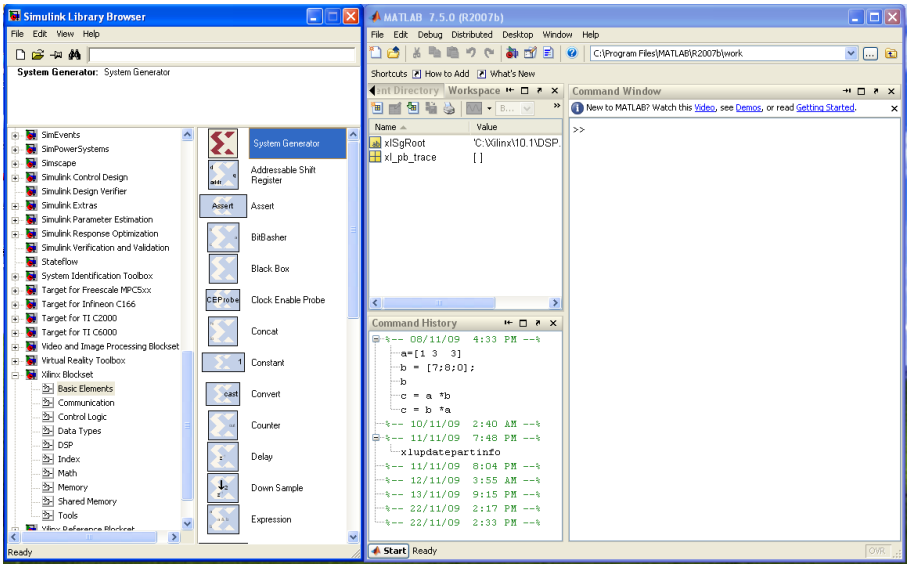

Figure 1: Simulink and MATLAB

- The Simulink library browser shows a list of all the different Toolboxes installed within MATLAB. Xilinx System Generator components will appear under three categories:

  1. Xilinx Blockset
  2. Xilinx Reference Blockset
  3. Xilinx XtremeDSP Kit

  The category **Xilinx Blockset** contains all the basic blocks used in various number of applications and will be used in this tutorial.

- Create a new Simulink model by selecting ***File*** → ***New*** → ***Model***.

3 Creating a DSP system

In this part of the tutorial we will create a simple DSP system that will be used to evaluate Equation 1:

$$z = (5 \times x) + (3 \times y) \tag{1}$$

The main operations required to implement this system are:

- Two multiplication operations.
- One addition operation.
- Two storage elements to store the factors (5, 3). In this tutorial we will use a constant block for these two factors.

For each of these operations a System Generator Block exists. Besides, any System Generator model requires a **System Generator block** to perform various hardware operations on the model.

3.1 Building the Hardware Model

For the new model created in Section 2 we will place several block from the **System Generator Blockset** category in Simulink library browser:

- **Xilinx System Generator Block**: The first block to be used in any System Generator model is the System Generator Block. This block can be found in the following Simulink category: ***Xilinx Blockset→Basic Elements→System Generator***. Place the System Generator block in the new model window as shown in Figure 2.

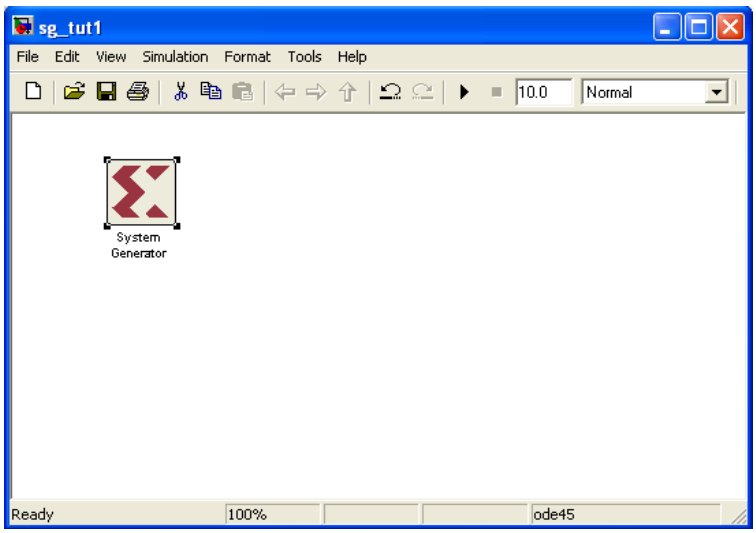

Figure 2: Creating a new Design

- **Input/Output Gateways**: are System Generator blocks that are used to convert data received from Simulink in the floating point format to fixed point format used inside the hardware system modeled using System Generator. Later, they convert the system output back to floating point. There are two types of gateways provided by System Generator:

1. **Input Gateways**: are used at the input of the System Generator system to convert floating point data info fixed point format. As show in Figure 3 the input gateway properties specify the details of the fixed point format. Three important properties are:

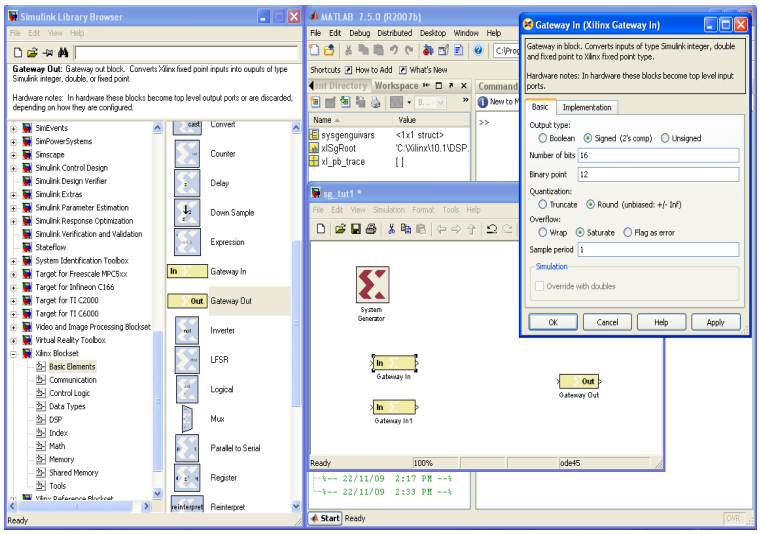

Figure 3: Adding IO Gateways

   - Output type: Three values can be assigned to this property: Boolean which is a single bit data representation, two's complement data representation and unsigned data representation.
   - Number of bits: Number of bits to represent the data. The higher the number of bits, the higher the resolution of the system.
   - Binary point: This is the position of the binary point in the fixed point format.

   Select ***Xilinx Blockset→Basic Elements→Gateway In***. Place two input gateways in the model as shown in Figure 3. Double click on the input gateway to change its properties. The **"Gateway In"** properties window will show up. Set the output type to two's complement, the number of bits to 16 and the binary point to 12 as shown in Figure 3. You can change the block name by clicking on it. We will leave the block names unchanged in this tutorial.

2. **Output Gateways**: are used to transform the data generated from the System Generator in fixed point format into floating point format required by Simulink. Output gateways automatically detect the fixed point format from the system output and do not require any modification. Place one output gateway by selecting ***Xilinx Blockset→Basic Elements→Gateway Out***.

- **Multipliers:** Our model requires two multiplication operation. Xilinx System Generator Blockset provides several blocks for arithmetic operations. To place a hardware multiplier block select ***Xilinx Blockset→Math→Mult***. Place two multipliers as shown in Figure 4. Double click on the **Mult** block to change its properties. As shown in Figure 4 there are two **basic options** for output precision in the basic properties: i) full and ii) user defined. In the full precision option, the multiplier block uses the input fixed point format to determine the format of the output. In our case the full precision requires 32 bits with the binary point at bit position 24. In the user defined precision mode, the designer can specify a different format. In this case the designer needs to specify a rounding method for excessive data values. In this tutorial we will use full precision option. Also

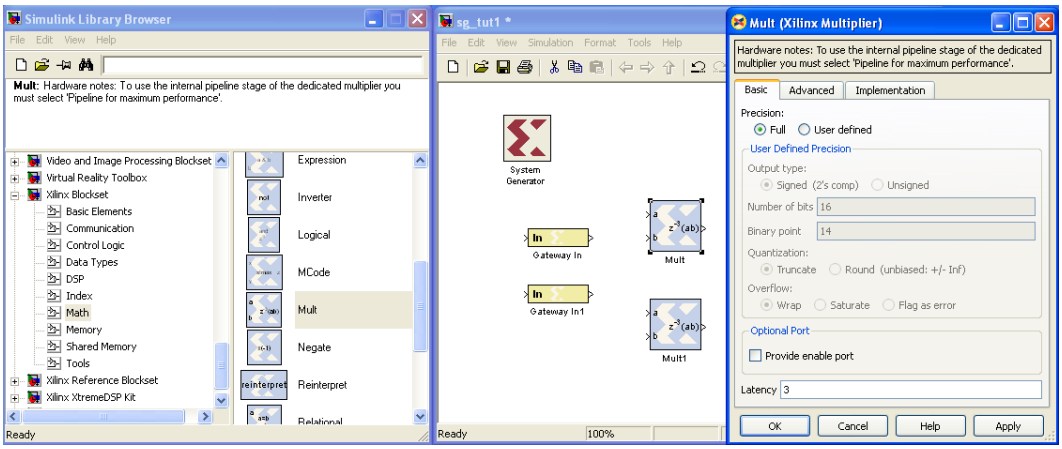

Figure 4: Adding Multipliers

in the **Implementation Menu** the user has the option of optimizing for speed/area and also using the dedicated Multipliers embedded in an FPGA or just the Look Up Tables (LUTS) instead.

- **Constants:** The system we are building performs two multiplications between the two inputs (variables) and two factors. For simplicity we will use constants to represents these two factors. Although in a typical DSP application we might need to change these factors and we may need to use memory elements to do so. Constants are usually implemented using hardwired configurations. Place two constants by selecting ***Xilinx Blockset→Basic Elements→Constant***. Double click on the constant block to display the constant properties and change the constants values to 5 and 3 as shown in Figure 5. If you use Fixed-point (Signed (2's comp)) then make sure you have enough bits to represent your number (Number of bits 16, Binary point 12) would allow you to represent fraction numbers +7 to -8. Connect the input gateways and the constant to the inputs of each multiplier. To

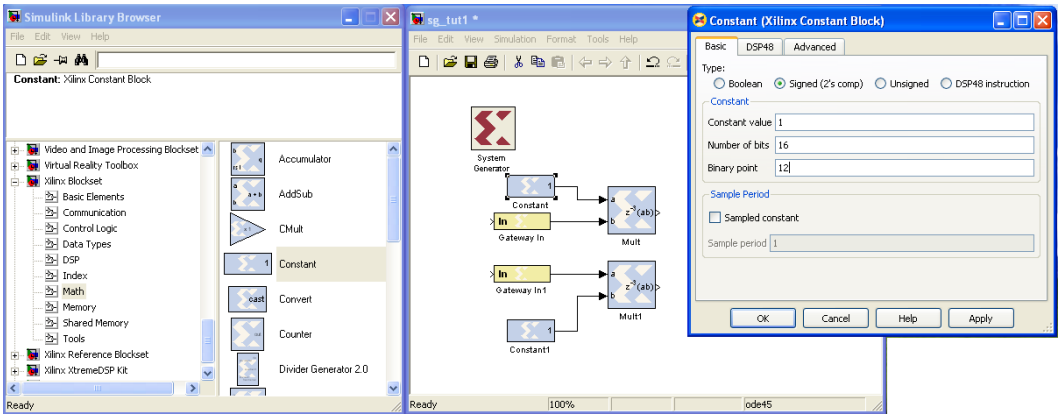

Figure 5: Adding Constants

connect the input gateways to the multipliers input click on the output terminal of the input gateway and with the mouse button down drag the connection link to one of the multiplier's input terminals. Using the same approach connect the constants output to the input of the multipliers as shown in Figure 5.

- **Adder/Subtractor:** The final block we need to place in order to complete the system is the

adder/subtractor block. Select ***Xilinx Blockset→Math→AddSub*** to place AddSub block. The AddSub block performs addition and subtraction operations for two operands. The fixed point format of the output is determined from the inputs format as shown in Figure 6. Connect the outputs of the two multipliers to the inputs of the Adder. Connect the adder to the Gateway out module.

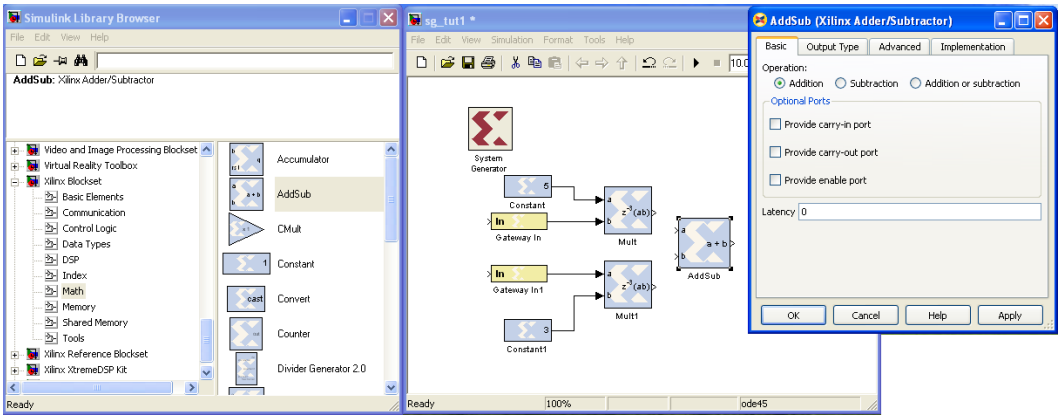

Figure 6: Adding an Adder/Subtractor

3.2 Estimate Resources

You can estimate the resources used by the design using the Resource Estimator block with multiplier block implemented in LUTs/Multipliers with latency set to 2 and 3 by following these steps:

1. Double click the **System Generator token** and select **HDL Netlist** for Compilation and set the **Part** related fields as:
   - Compilation: **HDL Netlist**
   - Part: **Spartan6 xc6slx16-2csg324**
2. Add the **Resource Estimator** block from Xilinx **Tools** library (Xilinx BlockSet Tools)

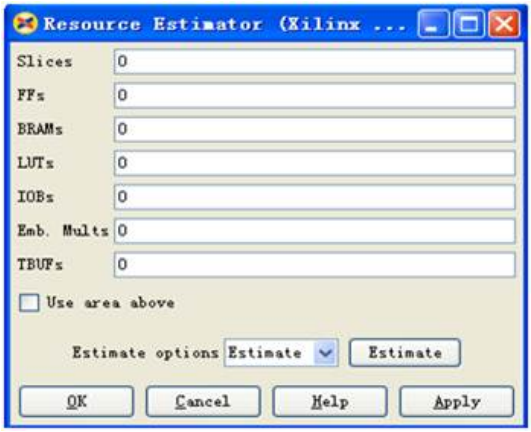

Figure 7: Resource Estimator Block

3. Open the **Resource Estimator** block (As seen in Figure 7)

4. With Estimate selected from the drop-down box, click the Estimate button.

5. The Estimate Resource tool will have different results based on whether the Multipliers were implemented using dedicated multipliers or the Look Up Tables of the FPGA.

3.3 Preparing the Simulation Environment

After completing the hardware system we will use Simulink environment to verify its functionality. Simulink offers a very flexible simulation environment which allows building different testing scenarios. For simplicity, we will build a testing scenario for our system by applying a constant input to the system and display the result on a single value display.

- Inputs: Simulink provides several blocks that can be used as an input to models generated and simulated using Simulink environment. These blocks can be found under the following category: ***Simulink→Sources***. From this category select ***Constant*** and place two constants in the model (the values should be 3.1 and 4.5 respectively). Connect these constants to the two input gateways of the system. We chose two arbitrarily values for the two constants as shown in Figure 8.

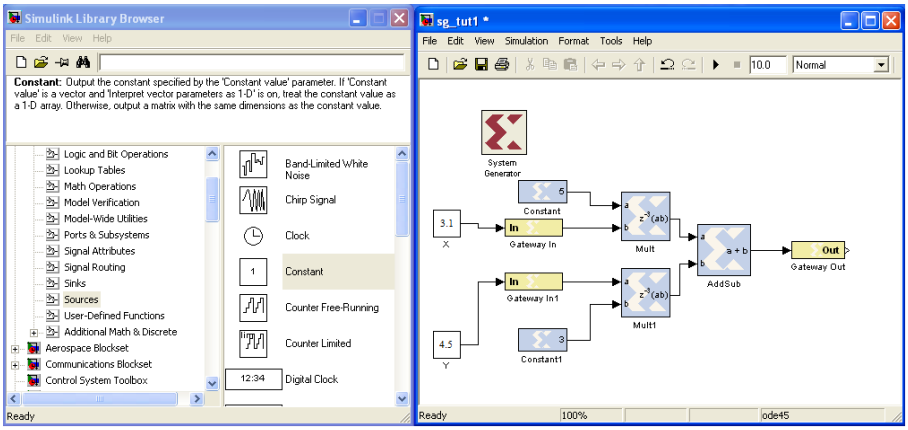

Figure 8: Adding a Simulink Constant to apply an Input to the System

- Output: Simulink provides several blocks to display the simulation results of the model under investigation. These blocks can be found in the category: ***Simulink→Sinks***. From this category select ***Display*** which is used to display the value of a single output. Connect this block to the output of the system as shown in Figure 9.

Figure 9 shows the complete model ready for simulation. Save your model to the file **sg_tut_1.mdl** by selecting ***File→Save*** in the model window.

4 Simulating a DSP system using System Generator and Simulink

The simulation process can be started by clicking the **Start Simulation** button in the toolbar of the model window. The execution of the model can be performed using several methods. Simulink can be used to perform real time simulations (functional simulation). In our case we need to verify the functionality of the model. For that reason no changes are required in the execution model. After starting the simulation process, System Generator starts to process each block in the model and generate a simulation model according to the specific configurations of each block as shown in Figure 10.

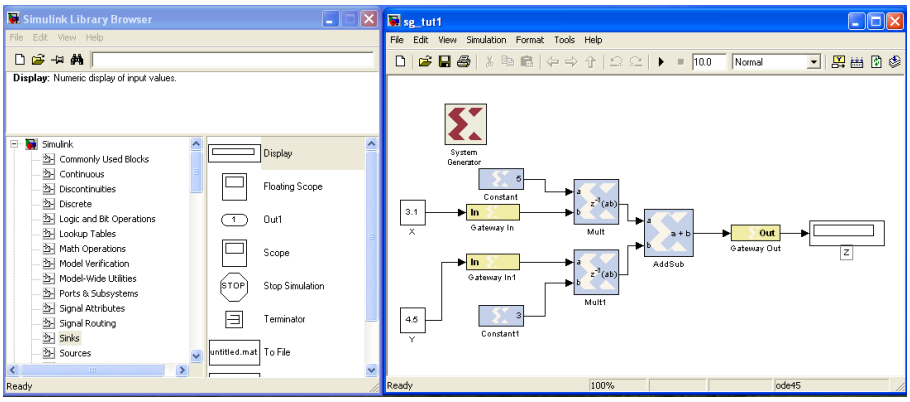

Figure 9: Display the Results in Simulink

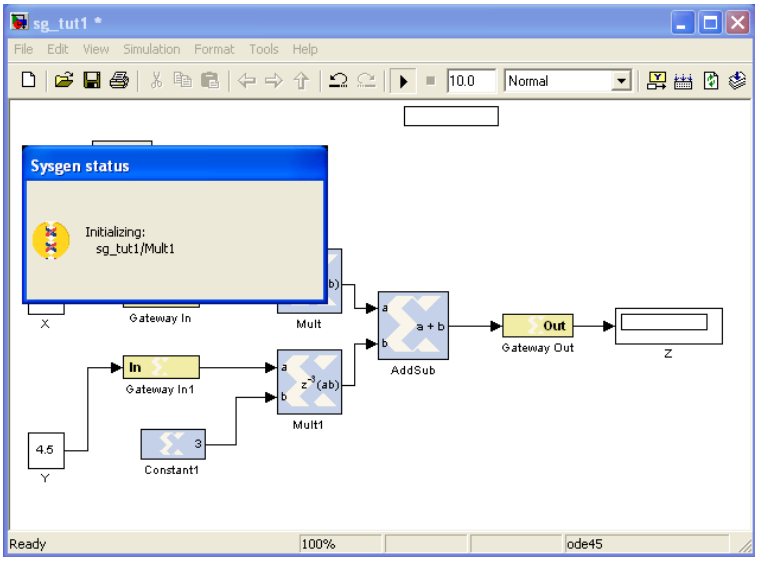

Figure 10: Simulating the DSP System in System Generator

This step is performed only once as long as the configurations for each block does not change. Using the values of **3.1** and **4.5** as inputs for both multipliers, the expected output of the system should be **29**. This can be verified by inspecting the output of the model as can be seen in Figure 11. This verifies the functionality of the DSP model generated by using Xilinx System Generator and Simulink.

5 Preparing System Generator for Hardware/Software Co-Simulation

In the previous section we verified the functionality of our simple DSP system using Simulink simulation models for different hardware component. Usually several issues may arise when the model is transformed into hardware. System Generator provides several methods to transform the models built using Simulink into hardware. One of these methods is called Hardware/Software Co-simulation. Hardware/Software Co-simulation enables building a hardware version of the model and using the flexible simulation environment of Simulink we can perform several tests to verify the functionality of the system in hardware. HW/SW Co-simulation supports FPGAs from Xilinx on boards that support JTAG or Ethernet connectivity. Several boards are predefined on System Generator for Co-simulation including the NEXYS3 (Spartan-6) board

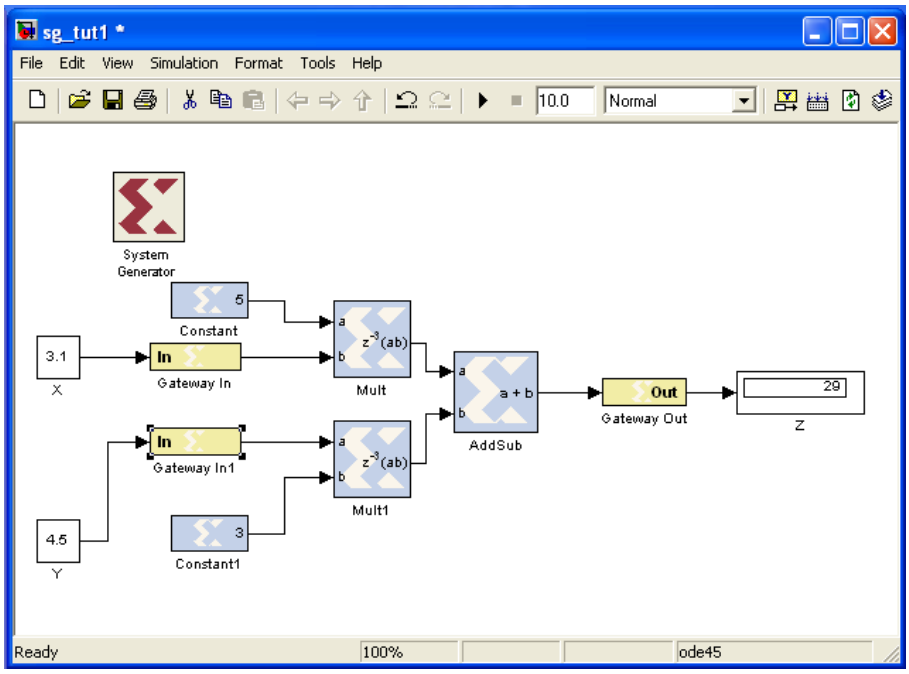

Figure 11: Verifying the Simulation Results

we are using in this tutorial.

5.1 Board Requirements for Co-Simulation

For a specific FPGA board to be used for Co-simulation, the following is required:

- A Xilinx FPGA which has enough resources for JTAG/Ethernet communication.
- Support for either JTAG/Ethernet communication.
- A free running clock.
- Xilinx Parallel/USB Programming cable for JTAG configuration/communication.

5.2 Generating Co-Simulation Module

Double click on the System Generator block. A dialog box will show up as shown in Figure 12. This dialog box allows you to select the type of the hardware generated using system generator. If the board is supported it will appear and you should follow the steps below. If the board is not supported then refer to Appendix A which describes in more detail how to configure System Generator to add your board.

- In the compilation list select ***Hardware Co-Simulation→NEXYS3_Board_Plugin_JTAG***. A new dialog box will show up highlighting the NEXYS3.
- Click **Generate** to build the hardware system. This step will generate a bit-stream that will later be used to configure the FPGA. ISE flow is used by System Generator to build this bit-stream. The progress of the process is displayed in the Compilation Statue window as shown in Figure 13.

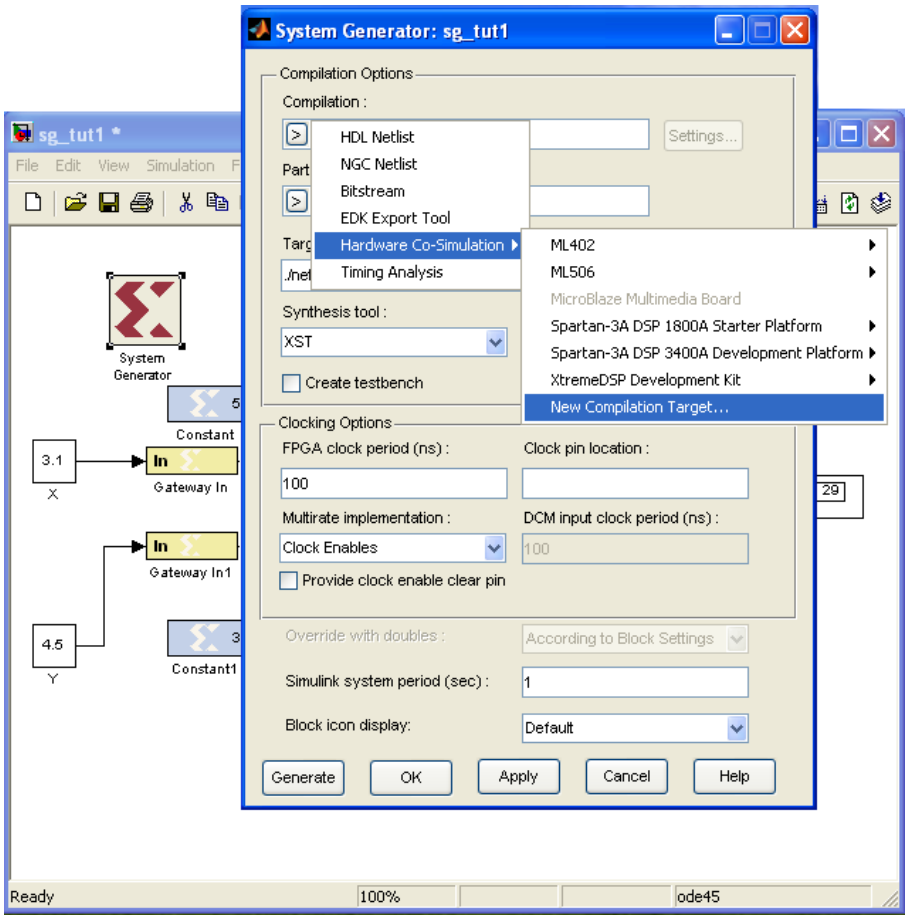

Figure 12: Configure System Generator for HW/SW Co-Simulation - Compiling a new Board

- When the compilation is complete, a new library is created including one block as shown in Figure 14. The library name should be **"sg_tut_1_hwcosim_lib"** and the block name should be **"sg_tut_1_hwcosim"**. The block has two inputs and one output as required by the DSP system. This block includes all the functionality required for the system to be executed on the FPGA.
- Now we are ready to perform HW/SW Co-Simulation for our DSP system.

6 Hardware/Software Co-Simulation

In the previous section two steps were performed:

- We configured System Generator for HW/SW Co-Simulation using NEXYS3 (Spartan-6) Board
- We generated a library with a new block that encapsulate the hardware implementation of the DSP system. This block is linked to a bit-stream that will be downloaded into the FPGA during Co-Simulation.

In this section we will modify the DSP model to use the new Co-Simulation block and replace the simulation models used before.

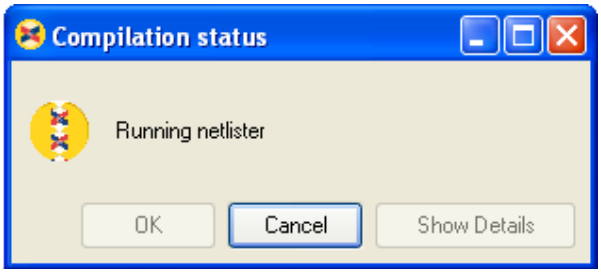

Figure 13: Building the Design Netlist

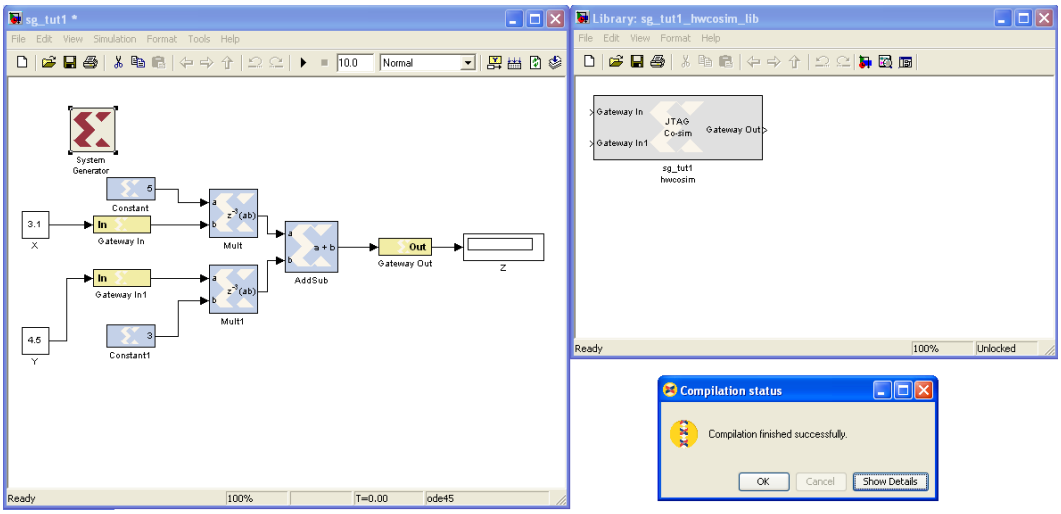

Figure 14: Netlist building Complete

- Make a copy of the model generated in Section 3.1 by Selecting ***File→Save as*** and use the file name **sg_tut1_co**.
- In the model **sg_tut1_co** replace all the hardware components with the **"sg_tut_1_hwcosim"** from the library **"sg_tut_1_hwcosim_lib"** as shown in Figure 15.

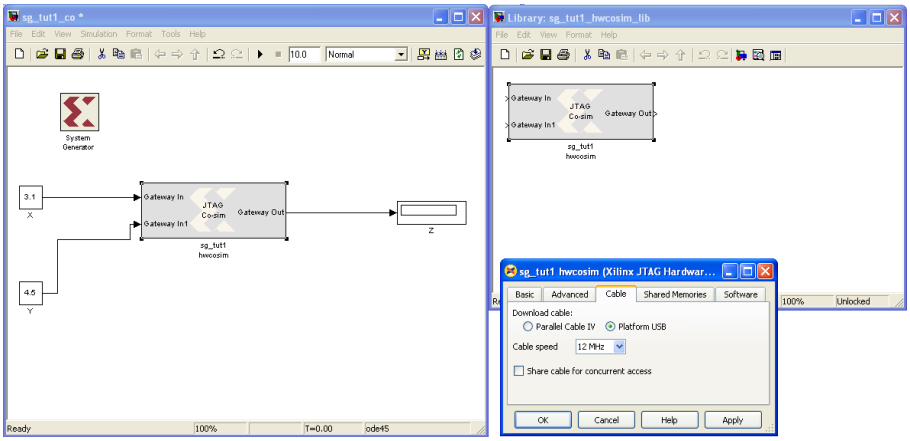

Figure 15: Modifying the Design for Co-Simulation

- Connect thr FPGA 'USB Prog' cable and 'UART' Cable to USB Ports and power it on. Wait for all Microsoft Windows drivers to be loaded.
- Double click on the **"sg_tut_1_hwcosim"** block. The block properties window will appear as shown in Figure 15. For the download cable select **Digilent USB JTAG Cable** as the NEXYS3 (Spartan-6) boards uses Digilent USB JTAG download cable. Click **OK**.
- Now the design is ready for Co-Simulation. Click the **Start Simulation** button in the model window toolbar to start the Co-Simulation. The System Generator will first download the bitstream associated with the block **"sg_tut_1_hwcosim"** as shown in Figure 16.

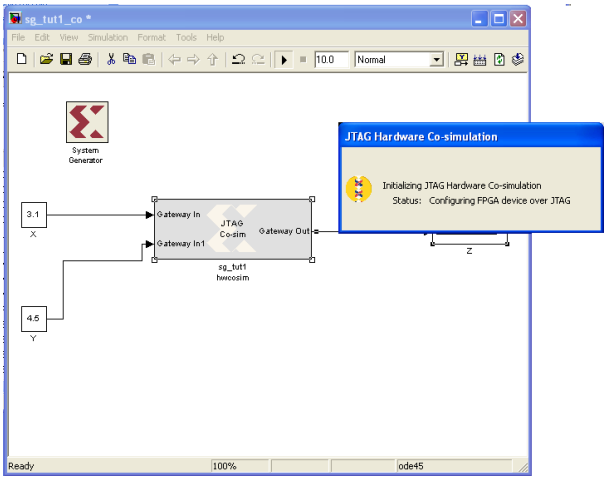

Figure 16: Starting HW/SW Co-Simulation

- When the download completes, System Generators reads the inputs from Simulink simulation environment and send them to the design on the board using the JTAG connection. System Generator then reads the output back from JTAG and sends it to Simulink for displayed.

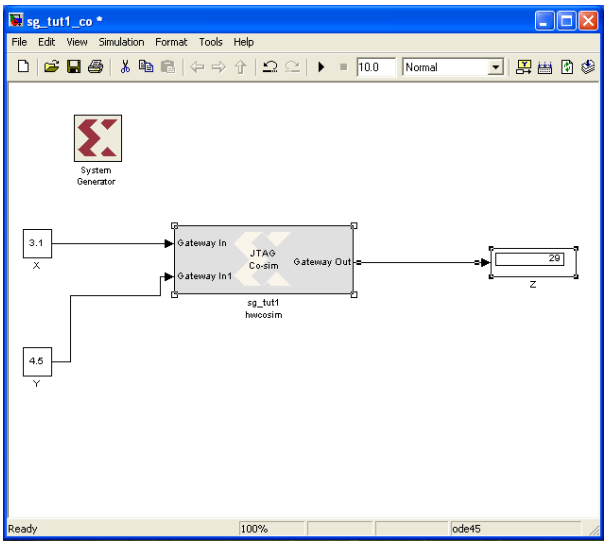

Figure 17: Verifying the Results for HW/SW Co-Simulation

- After the simulation is completed the results should be displayed as shown in Figure 17. We can verify the results by comparing the simulation output to the expected output (the expected output is **29**) as shown in Figure 17.

7 Generating the HDL Code

One of the advantages of Xilinx System Generator is the capability of generating HDL code from your designs. **Make sure you use the original model sg_tut1 developed and not the one used for co-simulation**. By following the steps below you should be able to generate the VHDL code and analyze the implemented design using reports generated in ISE Foundation.

1. Make sure the latency of the multiplier block is set to 2 and under the "Implementation tab" check that the embedded multiplier usage is checked, and the "Use behavioral HDL" option is checked.

2. Doubel click the **System Generator** icon (See Figure 18) and specify the following settings:

   - Compilation: **HDL Netlist**
   - Part: **Spartan6 xc6slx16-2csg324**
   - Synthesis Tool: **XST**
   - Hardware Description Language: **VHDL**
   - Target Directory: **./ise**
   - Create Testbench: **unchecked**
   - FPGA System Clock Period (ns): **10**

Figure 18: System Generator Parameters for Generating VHDL Code

3. Click **Generate** to generate the HDL code and ISE Project files.

4. Select **Start** → **All Programs** → **Development** → **Xilinx** → **ISE Design Suite 14.6** → **ISE Design Tools** → **Project Navigator**

5. Open the generated project by selecting **File** → **Open Project** and select **sg_tut_1.xise** in the **ise** project directory. The file can be found in your current directory you setup in Matlab.

6. Highlight the top-level file, called sg_tut_1.vhd, and Double click on **Implement Design**.

8 Summary

In this tutorial we demonstrated the use of Xilinx System Generator to perform the following tasks:

- Build a simple DSP system using a basic DSP blocks from Xilinx System Generator Blockset within Simulink simulation environment.
- Simulate the DSP system using System Generator and Simulink.
- Configure System Generator for Co-Simulation using NEXYS3 (Spartan-6) Board.
- Perform Hardware/Software Co-Simulation for the proposed DSP system.

By the end of this tutorial you should be familiar with Xilinx System Generator design flow.

9 Appendix A – Setup A New Board for Co-Simulation

There are two methods that can be followed to add your board to the System Generator.

9.1 Copying a Directory available from Xilinx

Users can obtain the "Nexys3_board_plugin.zip" from Xilinx. Unzip this file in temp. The plugin files are required to enable JTAG Co-simulation targeting the NEXYS3 board. Unzip Nexys3_board_plugin.zip file in

`<xilinx14.6>\ISE_DS\ISE\sysgen\plugins\compilation\Hardware Co-Simulation directory`

9.2 Manual Addition of a Board

By following the steps below you will be able to setup NEXYS3 (Spartan-6) Board for HW/SW Co-Simulation if it is not defined:

- Double click on the System Generator block. A dialog box will show up as shown in Figure 19. This dialog box allows you to select the type of the hardware generated using system generator.
- In the compilation list select ***Hardware Co-Simulation→New Compilation Target***. A new dialog box will show up allowing you to configure a new board (System Generator Board Description Builder). Note that there are several boards predefined in the list.

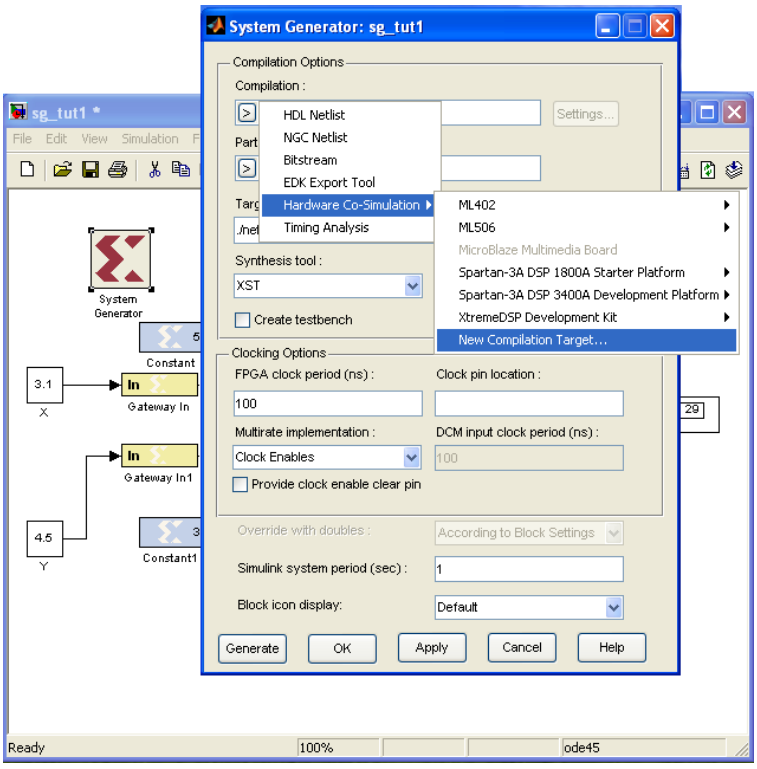

Figure 19: Configure System Generator for HW/SW Co-Simulation - Compiling a new Board

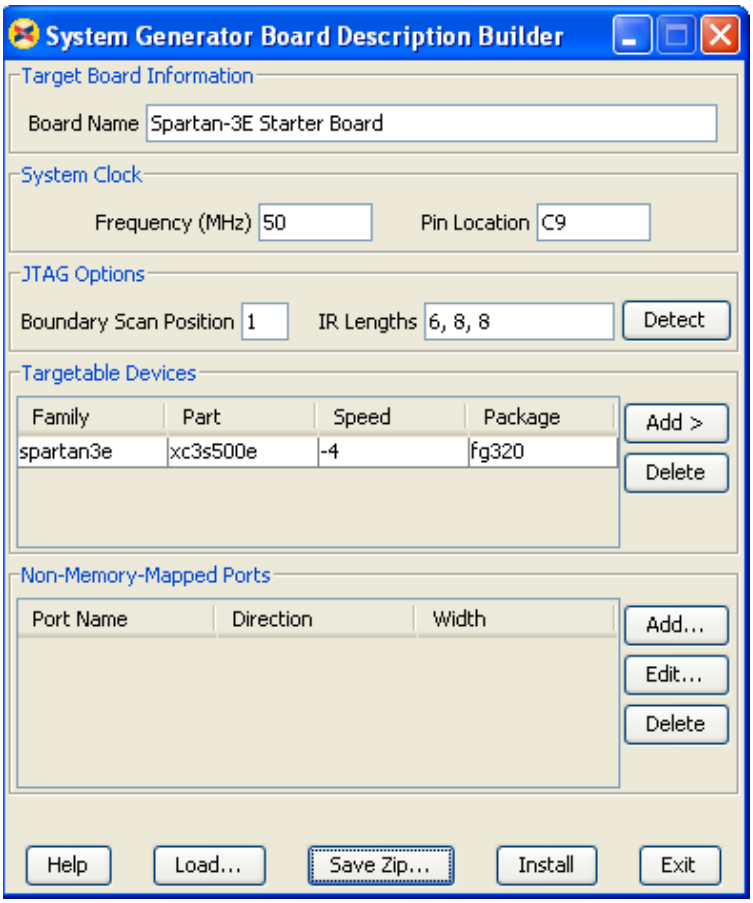

Figure 20: Configure System Generator for HW/SW Co-Simulation - Setup the NEXYS3 Board

- The **System Generator Board Description Builder** dialog box is used to configure a new board to be used for Co-Simulation. For each board the following information is required to define the board for Co-Simulation using JTAG (See Figure 20):
  - Board Name: The name of the board that will appear in the System Generator Co-Simulation list.
  - Clock Pin Location: The FPGA pin number that is connected to the free running clock.
  - Clock Frequency: The frequency of the free running clock.
  - The FPGA part number: The FPGA part number.
  - The FPGA position in the JTAG chain: The position of the FPGA in the JTAG chain.

  Note: If this is the first time to use the System Generator you need to build the parts list and store them into MATLAB. Execute the command xlupdatepartinfo in the MATLAB command window as shown in Figure 21.

  For the NEXYS 3 Board use the following values:

  - Board Name: NEXYS3 (Spartan-6) Board
  - Clock Pin Location: V10

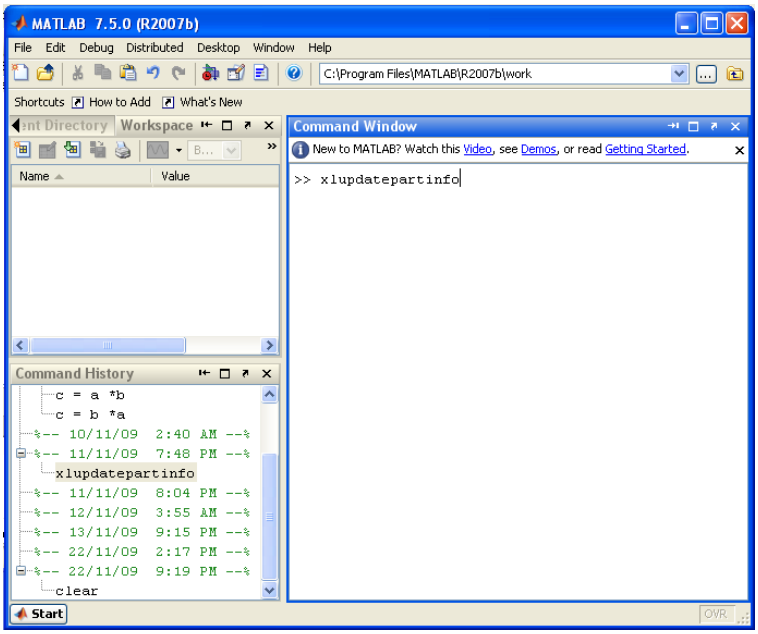

Figure 21: Initialize Parts List

- Clock Frequency: 100 MHz
- The FPGA part number:
  - Family: spartan6
  - Part: xc6sLX16
  - Speed: -3
  - Package: csg324
- The FPGA position in the JTAG chain: 1

Click **Install** to build the required files to configure the board for System Generator.

Note: You can save the configurations file into a compressed archive so that you can use it later to configure the board in case System Generator was reinstalled. Click "Save Zip" to create the compressed file.

- After installing the new board configuration, the board name **"NEXYS 3 Board"** should appear in the Co-Simulation list in the System Generator properties as shown in Figure 22. Select the board from the list.

  Note: Board configuration for Co-Simulation is required only once for each new board. The board name will always appear in Co-Simulation list every time you use System Generator.

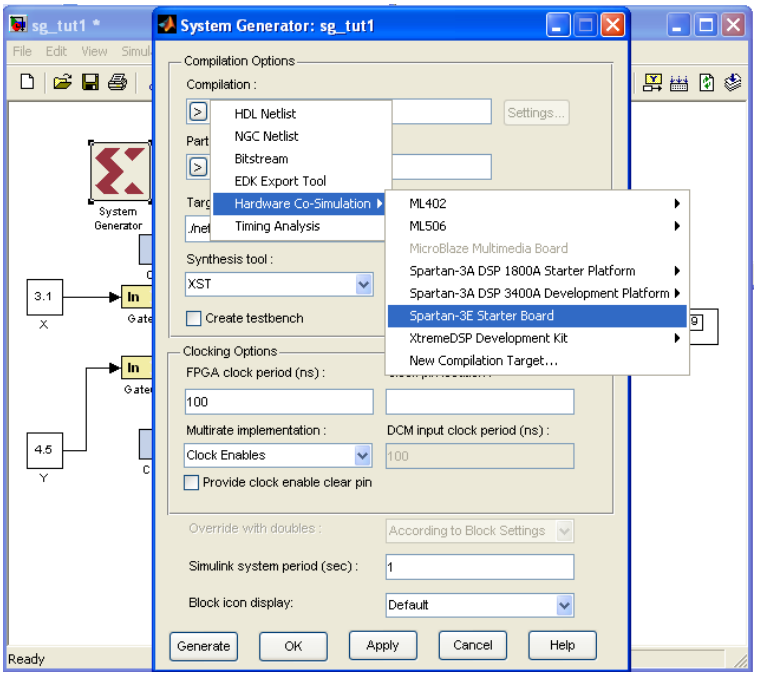

Figure 22: Configure System Generator for HW/SW Co-Simulation - Selecting the new Board