Getting Started with Xilinx System Generator
Getting Started with Xilinx System Generator

Introduction

This lab will introduce students 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.

Note: There is a completed example in lab2/solutions folder

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

The following software is required to be installed on your system to successfully complete this lab

1. System Generator 8.2
2. ISE 8.2.01
3. ISE CORE Generator IP Update 1
4. MATLAB / Simulink R2006a
**Procedure**

1. Launch the MATLAB program and once invoked change the working directory to:
   
   C:\SysGen_Training_Labs\lab2

2. Open the file “lab2.mdl” and observe the following design

   ![Simulink Executable Spec for a multiply-add circuit](image)

   **Figure 1 – Simulink Executable Spec for a multiply-add circuit**

   **Note:** This design is an executable specification creating in Simulink using the standard blockset. It is a simple multiply-add circuit but serves to demonstrate many of the key concepts of model based design. We 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.

   ![Multiply/Add Waveform](image)

   **Figure 2 – Multiply/Add Waveform**

4. From the Simulink Library Browser, open up the Xilinx Blockset Library and the “Index” sub-library to access the blocks. Once open 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. See **Figure 3**. Leave all the block settings at their default values.
5. Simulate the design and view the waveform on the scope attached to the Xilinx implementation. You will 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 which operates on synchronous clock edges.

6. Compare the results from the executable spec vs. the Xilinx implementation using a subtractor from the “Simulink/Math Operations” library as shown in Figure 5. This is an important model based design concept.
The waveforms do not match. What has happened is that the Xilinx blocks have introduced three clock cycles of latency into the design through the multiplier which is pipelined. If you double-click on the Xilinx multiplier block you will see the latency specification for that block is set to three. The executable spec will need to be adjusted to account for this hardware latency.

7. Add 3 unit delay blocks to the output of the adder as shown in Figure 6 and resimulate the design.

![Figure 6 – Delaying the output of the executable spec adder](image)

The simulation results should now closely match. There will be some slight discrepancies due to the reduced precision effects of the fixed-point arithmetic being performed by the Xilinx DSP blocks.

The design is not 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.

8. Double-click on the System Generator token and set the output target to “Bitstream”. Also, set the output device to Virtex4 xc4vsx25-12ff668.

![Figure 7 – System Generator settings](image)

9. Once complete click the “Generate” button to initiate the implementation process.

System Generator will automatically execute the RTL generation, synthesis and place and route programs to create an FPGA programming file. Optionally users could select to generate an intermediate format such as HDL (synthesis) or NGD (place and route) and run these steps interactively.
10. Once the generation has completed click on the “show details” button from the “Compilation Status” dialog box, which will display the implementation transcript,

![Compilation status dialog box]

Record the following results from the log file

- # of DSP48
- # of slices

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

11. Select all the Xilinx components, including the System Generator token and push then into a subsystem by hitting “cntl<g>” on the keyboard. The diagram should look like Figure 7.

![Figure 7 – System Generator Subsystem]

12. Copy the subsystem to create a second subsystem and connect it to the design as shown in Figure 8.

![Figure 7 – Simulink diagram showing two subsystems]

We now have Simulink diagram that contains two subsystems each with a System Generator token. This would represent two FPGAs or two blocks of a single FPGA within a larger DSP system. Each System Generator token will create 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.

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

![Figure 8 – Multiply / Add using the DSP48 Macro block](image)

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.

14. Double click on the DSP48 macro block to bring up the properties dialog box editor and modify the equation to be “Xo*Yo+No”. Observe the other implementation options but leave them in their default values.

![Figure 9 – DSP48 Macro Properties Editor](image)

15. Resimulate the design to insure functional correctness.

16. Double click on the System Generator token to generate a bitstream. Record the following results:

<table>
<thead>
<tr>
<th># of DSP48</th>
<th># of slices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

End of Lab2