



Tech Tips

Xilinx technical support answers your DSP application questions.

by Jeffrey Harriman
DSP Tools Product Applications Engineer
Xilinx, Inc.
jeffrey.harriman@xilinx.com

These tips offer insight into popular questions from Xilinx® System Generator for DSP users trying to run hardware co-simulation for hardware verification and simulation acceleration.

I hope these tips will give you some guidance next time you're thinking about using System Generator for DSP for hardware co-simulation. Xilinx also provides additional examples in the System Generator for DSP documentation.

Q: How can I use hardware co-simulation with System Generator for DSP to accelerate my simulation?

A: Hardware co-simulation is an excellent tool for increasing the simulation speed of complex System Generator for DSP models. There are several options when setting up a hardware co-simulation that affect how data is transferred between the FPGA hardware and your computer. Depending on your design, one may be more appropriate than another.

In the most common type of hardware co-simulation, you will take your System Generator model "as is" and select your hardware co-simulation target as the compilation target for generation. The Gateway In and Gateway Out blocks become input and output ports on the generated hardware co-simulation block. For this type of setup, "single stepped" simulation mode is the best simulation mode. In this mode, the FPGA hardware is kept in sync with the

Simulink simulation by System Generator, which drives the clock along with the data at each time step.

By adding the newly generated hardware co-simulation block to your model, you can now simulate a system-level Simulink model with FPGA hardware running the Xilinx System Generator for DSP portion of your design. You can leave the System Generator blocks from which the hardware co-simulation block was generated and run them side-by-side with hardware to verify that they match. Or you can remove all of the software simulation blocks to give your simulation a speed boost, offloading the System Generator portion of your design completely to hardware for simulation.

Q: Which development board and type of interface is the best for running hardware co-simulation?

A: One of the most important factors affecting simulation speed is the type of interface used to communicate with the board. The available interfaces are JTAG, PCI or PCMCIA, and Ethernet. JTAG is the most flexible option, as it can be used with any board with a JTAG connection to a Xilinx FPGA. However, JTAG typically has the lowest bandwidth. The System Generator board description builder is available to help generate the necessary board support files to add custom boards to the list of compilation targets in System Generator. In the GUI, you can specify board details such as clock rate, JTAG chain information, and board-specific I/O ports (Figure 1).

The Nallatech XtremeDSP™ kit boards have built-in support using PCI, while other companies use a PCMCIA

interface for hardware co-simulation. Both of these options will offer increased simulation throughput over JTAG when running hardware co-simulation. These development platforms often include DSP-specific hardware such as ADCs and DACs.

Ethernet offers the fastest hardware co-simulation speeds. The XtremeDSP Spartan™-3A DSP edition supports Ethernet connections at 10 and 100 Mbps, while the ML506 and ML402 can support up to 1 Gbps connection rates. These boards also allow you to run a point-to-point or network-based Ethernet connection. Because network-based Ethernet co-simulation has to share traffic and deal with additional Ethernet protocols, the point-to-point setup has much higher performance.

Many factors can result in some performance overlap of the various interface options:

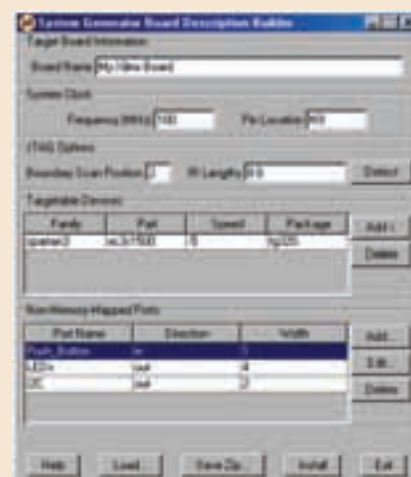


Figure 1 – The System Generator board description builder makes it easy to add hardware co-simulation support for any board with a Xilinx FPGA in the JTAG chain.

data width, the number of ports, and the number of simulation steps, for example. Note that there is some overhead associated with hardware co-simulation – device configuration and initializing the interface – so the more data samples collected from a simulation, the more advantages you have by using hardware co-simulation.

Q: How can free-running mode and shared memories be used with hardware co-simulation in System Generator?

A: The free-running simulation mode has two common applications. The first involves interaction with other hardware on your board. For instance, you may want to process a real-world signal that comes in from an ADC and send the processed signal back out to a DAC to view on an oscilloscope. In this case, you can use

co-simulation bandwidth. To use shared memories in free-running mode, they need to be “lockable,” which means using additional control signals to request and grant access to the shared memory space.

Because the design running on the FPGA uses the on-board clock in free-running mode, it is important to use data-valid signals throughout the design so that the hardware can be paused when there is no data to consume from the shared memory buffers. For this type of simulation, the Simulink test bench should buffer the data so that it matches the size of the shared memory buffer in hardware. To maintain efficiency on the hardware side, it is important to use memory buffers sized appropriately for your design.

The “Simulink System Period” set in the System Generator token determines how often the memories on the FPGA and

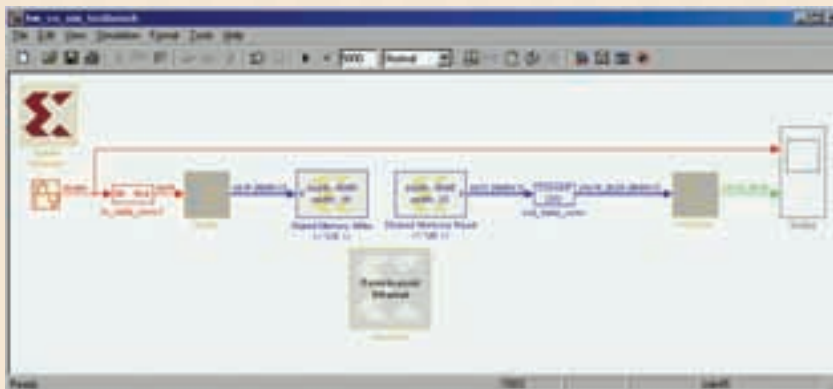


Figure 2 – Shared memories are used to transfer blocks of 4,096 samples between Simulink and the FPGA hardware to increase the hardware co-simulation bandwidth.

non-memory-mapped ports (such as those in the XtremeDSP kit block set) with a free-running hardware co-simulation to connect your design to the hardware-wired FPGA pins instead of communicating with the Simulink simulation.

Another scenario using free-running simulation involves the shared memory blocks in the Xilinx block set. The frame-based nature of the shared memory interface makes it a good fit for frame-based designs such as video processing, where you can leverage the shared memory blocks using large block transfers down to the FPGA for great increases in your hardware

CPU are synchronized. By setting this so that a vector is transferred only once each time the buffer is filled up, you will get a more efficient simulation and optimize your shared memories.

For example, in the design in Figure 2, with 4k deep shared memory, the “Simulink System Period” is set to 4,096 times the Simulink sources sample period (the sine wave generator on the left). In this way, after the appropriate number of samples are buffered up in software, a block transfer is performed down to the hardware memory instead of transferring a single sample per simulation step. ●●

**Supporting Your Future
HUNT ENGINEERING**

USB connected Programmable FPGA systems

V-II Pro PowerPC

- Virtex-II Pro XC2VP7
- 256 Mbytes DDR Memory
- Configurable digital I/Os
- PowerPC boot FLASH
- USB 2 or Standalone

Software Defined Radio

- Virtex-II FPGA 1M gates
- 2 ch 125Mps A/D and D/A
- TI C6203 DSP
- 32Mbytes SDRAM
- Configurable Digital I/O
- USB 2 or Standalone

Imaging with Virtex-4FX

- Virtex-4 FPGA FX12
- 128Mbytes DDR Memory
- CameraLink connection
- VHDL and PowerPC Imaging Libs
- USB 2 or Standalone

Programmable hardware with cables, device drivers, loading tools, examples and Power Supply. Systems can be used connected to a PC using USB, or can function standalone (without USB) using the initialisation PROMs.

sales@hunteng.co.uk
+44 (0)1278 760188
www.hunt-rtg.com