Marrying SoC Platform Designs with System Generator for DSP

by Daniel E. Michek
Senior Manager, System-Level Product Marketing
Xilinx, Inc.
daniel.michek@xilinx.com

The Vivado System Generator tool easily connects into platform designs to leverage board interfaces and processing systems.

FPGA applications are ever evolving, and the FPGA design flows are evolving along with them. No longer do we use FPGAs as simple glue logic or even as the heart of a signal-processing chain that marries intellectual property (IP) to proprietary back-end interfaces. Instead, FPGAs are transitioning into programmable systems-on-a-chip, a combination of hardware designed to act as processor peripherals alongside high-level software running on powerful APUs. This is an architecture we refer to as Xilinx® All Programmable SoCs.

In order to take advantage of this new flow, we need to shift the design methodology from top-down RTL, as in the earliest days of the FPGA, to a bottom-up flow centered around IP development and standardized connections such as ARM®'s Advanced eXtensible Interface (AXI). As the interfaces evolve from custom to common, we reduce the effort expended on verifying interactions between the data path and the platform design.

Xilinx's System Generator for DSP has evolved too. This tool, which is part of the Vivado® Design Suite, integrates the new bottom-up design methodology by incorporating DSP data paths into platform designs constructed with the Vivado IP Integrator tool. Let's take a closer look at how design automation using System Generator is enabling high-performance designs to take advantage of platform connectivity.

BUILDING AN ALL PROGRAMMABLE PLATFORM FRAMEWORK

We start our new design flow by defining the platform framework in which we want to house the data path. The Vivado tool suite is board aware and we will take advantage of available board automation to build our new platform design.

A platform design or platform framework, as shown in Figure 1, is the basic collection of processor- and board-level interfaces, along with the logic combining them. We use the platform framework as the basis for our system-level design, the shell, and this leaves us with the room for our data path. Block and connectivity automation will link the processing systems through IP peripherals to the board-level interfaces. DSP data paths or software accelerators packaged in the IP Catalog can then take advantage of Xilinx's Designer Assistance automation to easily tie into our platform framework of processors and, by extension, interfaces to external devices.

CREATING THE DATA PATH AS IMPORTABLE IP

Our ultimate goal is to make the data path accessible to the All Programmable platform framework. If we wanted to start from scratch, we could create the data path with standardized interfaces. By quickly marking a gateway port as an AXI4-Lite interface as shown in Figure 2, or simply naming our ports to match a standard connection like AXI4-Stream on our Simulink® diagram, System Generator will take care of adding the extra logic to our design and collecting the common signals into interfaces as it packages the design for the Vivado IP Catalog.

But a new method allows us to use the platform framework to tailor-fit a plug-in that marries to the All Programmable design. We use automation to determine which interfaces exist within a platform design, which interfaces associate to the board and which interfaces create a plug-in for the DSP data path. Since the goal is to convert the data path into IP that connects to the platform framework, we do not need to focus on board-level interfaces but instead, on standardized AXI interfaces. Each interface not associated at the board level converts into System Generator gateways. While acting as

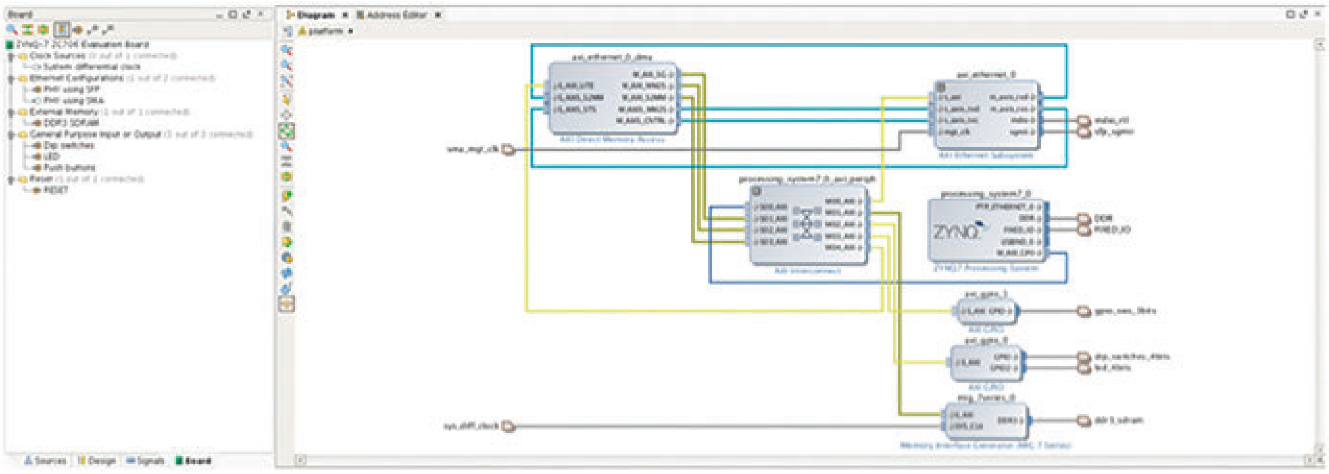
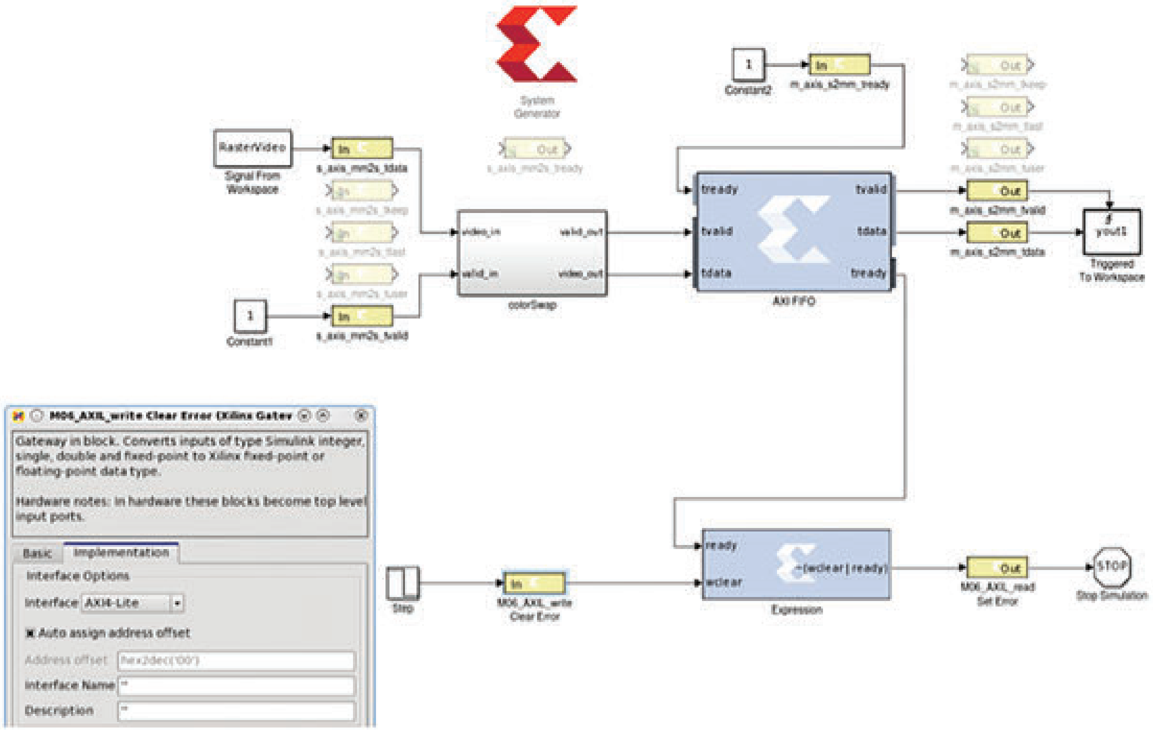


Figure 1 – Example of a platform framework connecting a processing system to board-level interfaces

simple signals in the System Generator world, these gateways produce AXI interfaces to connect to the platform design when we export it to the IP Catalog.

As one example, AXI4-Lite interfaces create independent read and write signals that share a commonly addressable register interface when exported to the Vivado tool suite. A trivial copy-and-paste gives us more direct registers available to the processor at an address offset through the same interface. At the same time, we



M06_AXIL_write Clear Error (Xilinx Gatew

Gateway in block. Converts inputs of type Simulink integer, single, double and fixed-point to Xilinx fixed-point or floating-point data type.

Hardware notes: In hardware these blocks become top level input ports.

Basic | Implementation

Interface Options

Interface: AXI4-Lite

☒ Auto assign address offset

Address offset: hex2dec('00')

Interface Name: ''

Description: ''

Figure 2 – Automating gateways into AXI4-Lite and AXI4-Stream interfaces

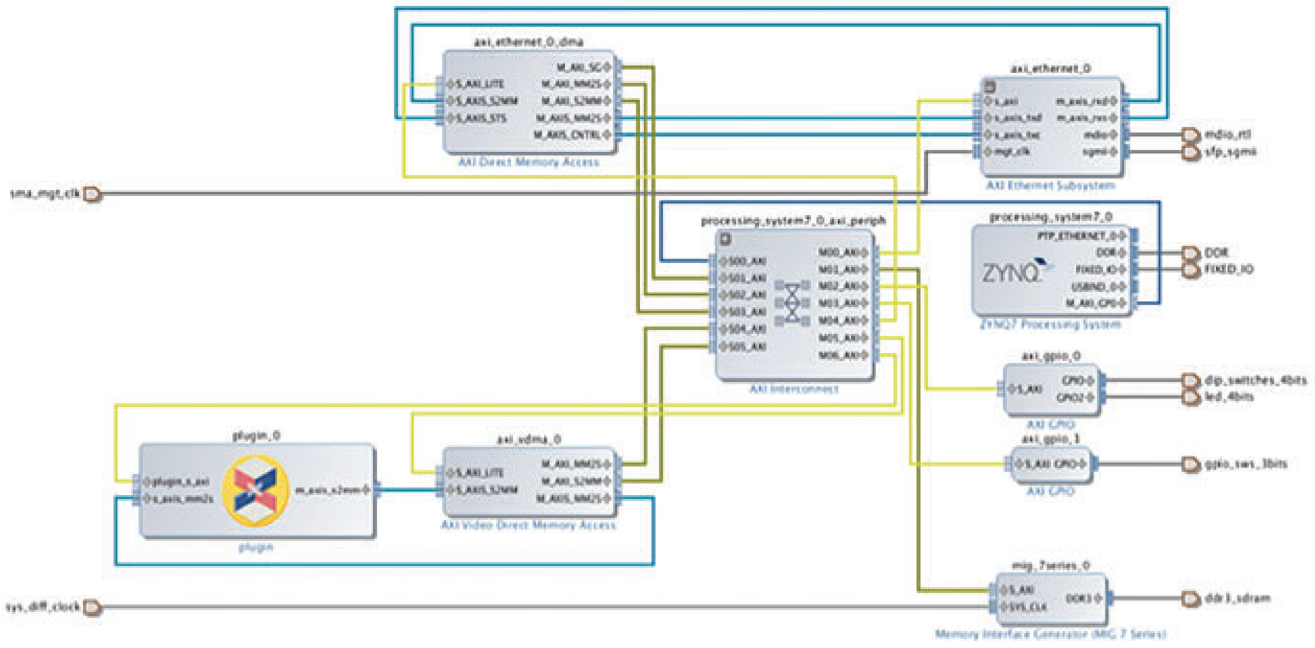


Figure 3 – Platform-based system that connects DSP data path to the platform framework

automatically generate software driver APIs to read or write to those registers.

If an AXI4-Stream interface is available from the platform design, System Generator will add appropriately matched gateways to the model. AXI4-Stream interfaces are extremely flexible and contain many signals. ACLK is the clock source associated with this interface, but this signal is directly implied as the abstracted system clock for this portion of the data path. TVALID is the signal that denotes when the interface is valid. Other signals are optional. System Generator will add those that exist in the originating stream interface to our model, but we can delete or add signals to match our internal requirements.

In the model shown in Figure 2, we see that our data path only cares about TDATA (the data sent across the interface) and TVALID on the S_AXIS interface. To remove unnecessary signals, we comment out the unused gateways for this model, since default values will drive the signal connections in IP Integrator.

The AXI4-Lite and AXI4-Stream signals easily simulate and verify with the bottom-up methodology. The AXI4-Lite interfaces model as simple gateways that we source and sink from the myriad of Simulink blocks, a simple abstraction of passing data from one side of the gateway to the other. Likewise, the AXI4-Stream interfaces are merely a collection of signals that follow simple handshake rules for passing data from one IP core to another.

Our simulation modeling is only challenged by the optional ports we use on our interface. If we accept data on every cycle and process it though our data path without interruption, we will not need the TREADY handshake signal. The simplified model sends each element of a vector through the TDATA gateway from Simulink's Signal to Workspace token. When a full handshake is desired, we model it with an AXI4-Stream FIFO to buffer our data as shown on the M_AXIS interface in Figure 2.

The tailoring automation acts as a starting point to create IP that interfaces to the platform framework. But System Generator is flexible and allows us to add or delete portions of, or even complete, AXI interfaces. The end result is a conversion of the data path into IP for reuse among multiple system-level designs.

After adding our logic, our last step is to export the data path we built with System Generator for DSP back into the Vivado IP Catalog. This action allows easy connection of the interfaces, whether working with RTL or within IP Integrator. Additionally, we generate drivers for use in the SDK and attach models for simulation with golden test vector data to the IP. And because we have prior knowledge of the platform framework when we created our DSP data path, we automate the incorporation of the model and the platform design, as shown in the completed system in Figure 3.

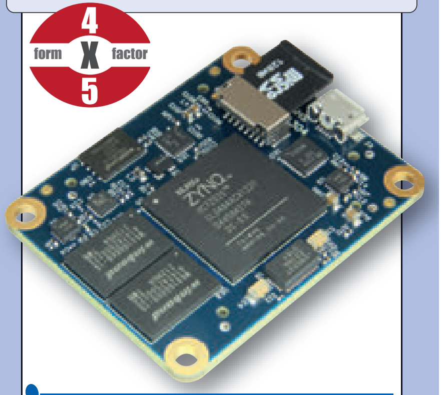
REDUCING SIMULATION RISK

Generating a complete system-on-chip that incorporates hardware accelerators, DSP data paths or custom logic is a challenge. Confirming that the data path will work as expected by simulating in a bottom-up fashion introduces a large risk, and ensuring that we maintain platform interface bandwidth to support the data path is equally daunting.

By utilizing standardized interfaces, we develop IP that reduces simulation risk. That's because the interactions at the interface level abstract away, so we may focus on verifying the internal data path. And finally, by leveraging smart automation for board, block and connectivity, we generate the platform-based system that meets our needs and integrates our custom data path.

More information about System Generator for DSP is available at *www.xilinx.com/products/design-toolsvivado/integration/sysgen.html*. If you have any questions or comments, call the author, Daniel Michek, at (858) 207-5213, or e-mail *daniel.michek@xilinx.com*.

All Programmable
FPGA and SoC modules



rugged for harsh environments
extended device life cycle

Available SoMs:

ZYNQ KINTEX 7
ARTIX 7 SPARTAN 6

Platform Features
- 4×5 cm compatible footprint
- up to 8 Gbit DDR3 SDRAM
- 256 Mbit SPI Flash
- Gigabit Ethernet
- USB option

XILINX
ALLIANCE PROGRAM
CERTIFIED MEMBER — BASE

Design Services
- Module customization
- Carrier board customization
- Custom project development

trenz electronic

difference by design

www.trenz-electronic.de