

Integrating HDL Design and Verification with System Generator

Designing and verifying the CABAC functionality of an H.264 video encoder is easy with System Generator for DSP.

by Justin Delva
Staff DSP Engineer
Xilinx, Inc.
justin.delva@xilinx.com

Ben Chan
Senior Software Engineer
Xilinx, Inc.
ben.chan@xilinx.com

Shay Seng
Engineering Manager
Xilinx, Inc.
shay.seng@xilinx.com

Xilinx® System Generator for DSP is a MATLAB Simulink block set that facilitates system design. Targeting Xilinx FPGAs within the familiar MATLAB environment, System Generator for DSP gives you the ability to functionally simulate a design and use the MATLAB environment to verify bit- and cycle-true models against the golden reference results. These reference results can be produced either externally or inside the MATLAB environment, and you can target a Xilinx FPGA hardware platform all from within MATLAB.

System Generator complements HDL design tasks by providing an easily configured test bench for both functional simulation and hardware verification. You can simulate your HDL code within MATLAB by using the built-in interface to HDL simulators like ModelSim. A System Generator for DSP test bench platform built around the HDL code provides a powerful yet fast simulation environment that interacts seamlessly with ModelSim. Setting up this environment is easy.

You can use this same environment to test your HDL code running in real hardware without any modifications. The hardware co-simulation system uses pre-supported FPGA platforms such as the Xilinx ML506 board for performing either a Simulink-controlled stepped clock hardware run or a real-time data-burst run.

Systematic Design and Verification of a CABAC Module

The H.264/AVC video encoder is the product of years of collaborative efforts that resulted in a standard with good video quality at substantially lower bit rates than previous standards. A reference C source code called the H.264/AVC Joint Model (JM) is available to developers. You can use this source code as a starting point for the functionality implemented in HDL.

Context-adaptive binary arithmetic coding (CABAC) is part of the H.264 video standard. The functionality of the CABAC module is manually translated in HDL using standard communication primitives. The primitives mostly used in this verification are FIFO interfaces. The original JM source code is also used for generating test vector files for the module.

We have also built a test environment that uses the JM model to generate the input stimuli for the CABAC HDL module and verify the results of the output of the HDL with the results produced by the JM reference model. This is a significant improvement over traditional HDL ad-hoc test benches.

Module verification is a three-step process comprising:

1. Functional HDL simulation. MATLAB verification with input and output test vectors feeds the ModelSim simulation and compares results, respectively.
2. Functional hardware verification. An intermediate step for working out any bugs not caught during functional HDL simulations. This stage uses a System Generator for DSP-controlled single-step clocking. The input and output test vectors are extracted in

files from the original execution of the JM source model. Special care is taken to build file interfaces in System Generator for DSP.

3. Real-time hardware verification. Using the input test vectors of the functional HDL simulation, the design is tested in hardware at the targeted input rate and clocking frequency. The output of the hardware is captured into MATLAB and compared with the output test vectors.

Functional HDL Simulation

In this step, we integrate the ModelSim simulation with the System Generator for DSP simulation, as shown in Figure 1.

To simulate the HDL in System Generator for DSP, a black box of the top VHDL entity is created by inserting converters at the boundaries, also shown in Figure 1. These converters translate the ModelSim unknown "X" states to zeros in the Simulink simulation. The test bench setup of the total simulation is shown in Figure 2.

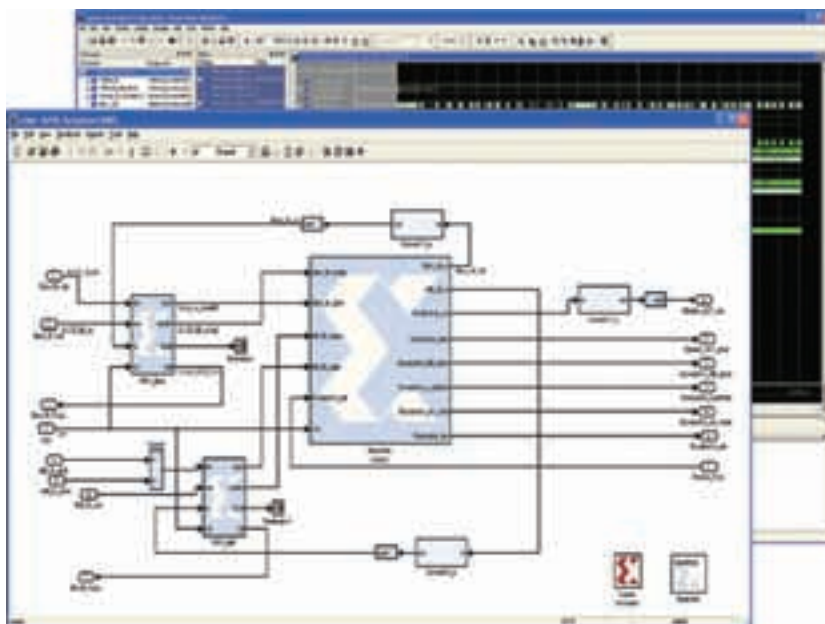


Figure 1 – Black box of CABAC HDL and ModelSim co-simulation

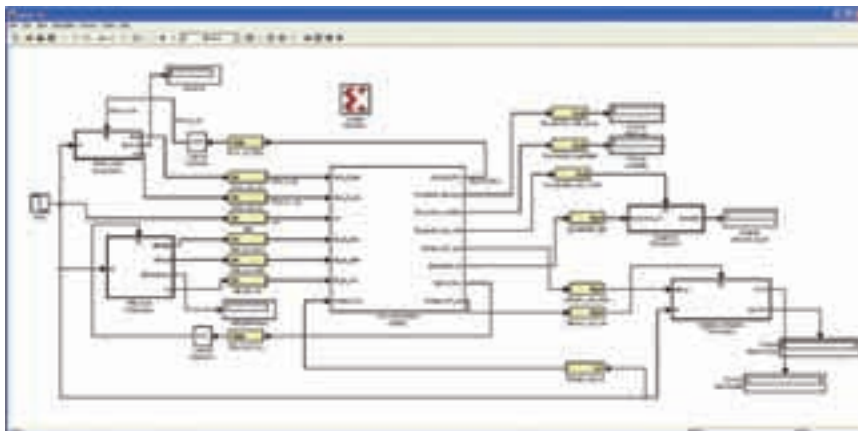


Figure 2 – Test bench setup for HDL simulation of the CABAC design

In Figure 2, the “Slice_input subsystem” block and “MB_input subsystem” block use special interface code to read from files containing stimuli created by the JM source code. The “Output_compare” subsystem is a special block that compares the results of the simulation with the original test vector results from the JM source code. This simulation is done on a single-step basis.

Functional Simulation with Hardware Using the ChipScope Analyzer

The next step is to use a similar environment where the complete HDL module is mapped onto hardware, in this case using an ML506 board with Ethernet and JTAG connections. The Ethernet connection is used to provide and read the stimuli, while the JTAG port is used to connect the ChipScope analyzer. This provides the same user experience, but now the HDL is completely implemented in hardware. The system setup is shown in Figure 3.

The advantage of this setup is that you can abstract completely from the details of the specific interfaces. You do not need to learn how the Ethernet connection is used to provide the input stimuli or read the output from the CABAC block. The special gateway blocks shown in Figure 3 abstract completely from these details.

Real-Time Hardware Verification

Previous simulations and executions provide a good environment for the detailed execution of the block at the cycle level. In a complete design, you often want to use large test sets that are representative of real-world test cases. The single cycle or step interface is not suited for this style of verification.

Using the same hardware setup with an ML506 board, verification now occurs with large data sets provided through a novel MATLAB interface called M-HWcosim. M-HWcosim is an API that transfers data to hardware from a MATLAB script M-file. Now the MATLAB scripting environment is used to provide all of the data to the actual CABAC block running in the hardware.

The implementation of FIFOs with flow control allows asynchronous communication between the computer running

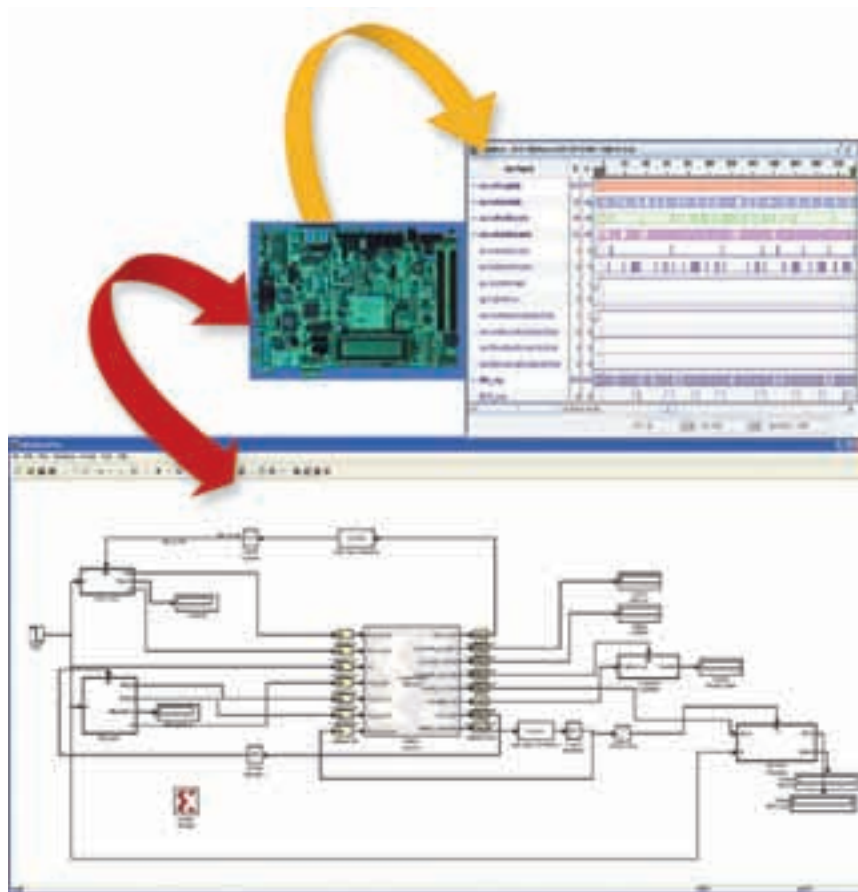


Figure 3 – Execution of the CABAC design in hardware, with Ethernet interface for stimuli and JTAG for the ChipScope analyzer.

MATLAB and the hardware running the CABAC block at full speed. This environment abstracts the details of this interface and has been critical in verifying large data sets of the CABAC module. For details about this environment, see the white paper, “Using System Generator for Systematic HDL Design, Verification, and Validation,” on www.xilinx.com.

Conclusion

In a complete system design, the verification is often as much work as the actual

design. The design of the CABAC block in H.264 leverages the JM source code model for generating test vectors from a high-level language. Here the HDL design verification is integrated with System Generator for DSP and MATLAB. Furthermore, the integration with complete boards that can run the CABAC block at speed is a major improvement over ad-hoc environments. This substantially reduces the time needed to build a verification environment, and therefore allows you to focus on the actual block at hand. ●●●

TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

- Familiarize yourself with System Generator for DSP by taking the free online recorded lecture, “System Generator, Getting Started Training.”
- Download the System Generator for DSP User Guide.
- Order the Xilinx H.264 CABAC core.