

FPGAs CAN BE GREAT
AT VIDEO PROCESSING,
BUT VERIFYING FPGA-
BASED VIDEO SYSTEMS
REQUIRES CAREFUL
ATTENTION TO YOUR
APPROACH.

FPGA VERIFICATION

in BY HARSHAL CHHAYA AND TARAK PATEL • eINFOCHIPS
embedded
video-processing
systems



Participating in the consumer-electronics segment has many advantages. Despite that fact, design teams in this segment will encounter a drastically reduced time-to-market window. Consequently, FPGA (field-programmable-gate-array)-based designs have evolved as the first choice of many system architects. Meanwhile, the increasing requirement for multimedia in consumer products has made DSPs and streaming interfaces must-have components in many embedded products. Several FPGA vendors have developed FPGAs with DSP cores and streaming interfaces that are technologically sufficient and complex enough to handle these recent design requirements.

An FPGA interfacing with DSP cores and with high-speed-video data streaming through it is far from a simple system, however. This increased design complexity has added verification challenges and raised the specter of costly re-spins of the system board if you catch a critical error late in the design cycle. To put that ghost back to bed, you must carefully consider your approach to verification so that you can reduce the risk of costly re-spins.

The largest advantage of FPGA-based design verification is that, at the lowest level, the system has a predefined archi-

ecture, so you know the scope of necessary test scenarios at the beginning of the design. With this fact in mind, a verification team can build on the FPGA a verification environment that mimics the actual system architecture.

In addition to verifying your external peripherals, you must verify any design elements internal to the FPGA, such as the DCM (digital-clock manager) and block RAM, that you use in your designs. This requirement gives you a lot to verify, however. So the time it takes for test cases to complete greatly affects overall product-development time. Therefore, the verification environment must be time-efficient. The early design of a proper verification environment that you develop with an understanding of the FPGA-design elements and external surrounding devices leads to accurate test-case writing and board bring-up.

CORRECT USE OF PRIMITIVES

FPGA vendors provide well-verified FPGA primitives, such as DCM and block RAM. You must comply with certain guidelines to correctly use these primitives in any FPGA design, however. It is crucial to catch any incorrect usage before the design goes to silicon. For example, one such DCM constraint is the allowed clock jitter on the input clock. In a test case, the DCM has a constraint of ± 300 psec on cycle jitter when in low-frequency mode. As per design specifications, the input clock for the DCM can be 16.384, 22.5792,

AT A GLANCE

- ▶ FPGAs can fit the needs of today's video-processing systems.
- ▶ In video-processing applications, a lot of care has to go into test-bench design.
- ▶ Making the verification environment as similar as possible to the real world eases board-level integration and reduces the need for re-spins.

or 24.576 MHz. However, during design verification, the DCM unlocks when experimenters switch the input clock from one frequency to the other because switching the frequency inherently violates the input-clock-jitter constraint. So a modified design implements a mechanism to reset the DCM while changing the input-clock frequency. If you do not catch such bugs during the front-end verification, it could easily take a week or more just to identify the bug during board bring-up.

With the advances in technology, FPGAs now include block RAM, which can be either a single- or a dual-port memory. As a dual-port RAM, the block RAM allows both ports to simultaneously access the same memory cell. If the designers improperly implement the RAM controller, however, both memory ports may attempt to write different data to the same RAM location during the same valid write cycle. The verification team must have separate tests for such scenarios. Thus, it is essential

that not only a FPGA designer but also a FPGA-verification engineer be aware of the requirements or constraints of internal FPGA components.

INPUT-SIGNAL VARIATIONS

In the real world, the input signals to your FPGA have routing-path delays and quality degradations. Your FPGA-verification plan should take into account such variations in timing and signal integrity when generating the input-stimulus signals. It is a good practice, for instance, to know how much drift an input signal would have from an ideal condition so that you can verify that the FPGA design will function smoothly during drift. This requirement becomes crucial when the interface is synchronous and an external device is driving the clock. The data, control, and clock may all have different delays based on the routing-path delay, clock-to-output delay of the transmitting device, and input-setup time of the receiving device. In high-frequency operation, this constraint may leave a narrow sampling window for the FPGA to capture the input data. In such cases, you should consider such real-time delays when providing stimulus to the FPGA design.

In the real world, input clocks come with jitter and drift variations. Although you can use a DCM to deal with these variations, the DCM has its own limitations in its tolerance of input-clock variations. A verification engineer must know the possible clock variations that can occur in the real system and incorporate the same variations when generating input clocks in the verification environment. Adopting such practices can help to discover the limitations of an FPGA design and implement corrective actions during the early development phase.

PERIPHERAL INTEGRATION

The rapid growth in streaming media requires systems to work at higher speeds. With higher-frequency systems, you must take care when integrating an FPGA with its peripheral devices. Those peripheral devices have timing constraints in input setup-and-hold time. Verification engineers must know these

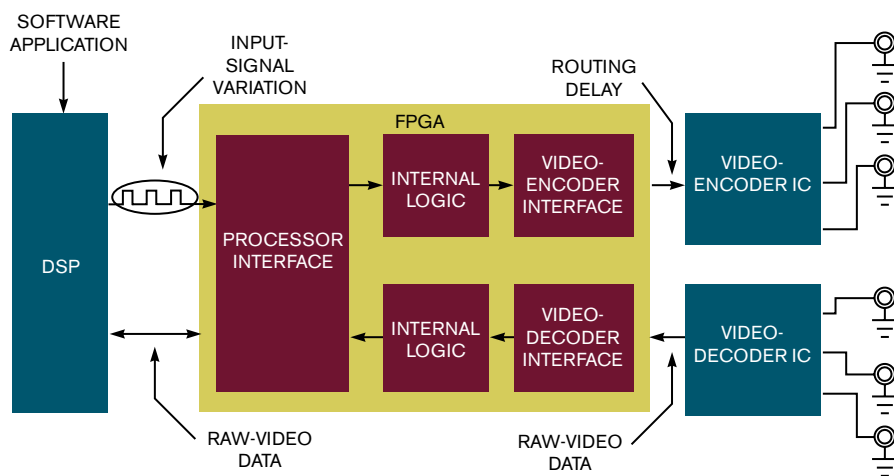


Figure 1 An FPGA can play a central role in a streaming-video application.

timing constraints for all the peripheral devices. Adding an oscilloscope for verifying FPGA designs with different timing constraints forces the designers to follow the proper design guidelines to make the FPGA design compatible with the system.

FPGAs may implement standard interfaces, such as UART (universal asynchronous receiver/transmitter), I²C (inter-integrated circuit), SPI (serial-peripheral interface), and GPIO (general-purpose input/output). The standards should dictate the verification strategy for these ports. When verifying such designs, you must also consider the timing constraints of peripheral devices using custom interfaces. For example, a GPIO interface of an FPGA may interface with an onboard multiplexer. The FPGA is responsible for driving selection inputs of the multiplexer and then capturing the multiplexer output. The multiplexer requires settling time on its outputs once the selection input has changed. Implementing that delay in generating response from a verification model ensures that the FPGA captures the multiplexer output only after the output has settled.

SYNCHRONIZING THE TEAMS

It is a good practice to give the FPGA design under test a feel for the real-time software application flow. FPGA designs that appear to work on the verification testbench may fail to survive when application software imposes some limitations of its own. Consider, for example, an FPGA design for high-definition video capture. The FPGA must capture the raw-video data and fill up an internal FIFO (first-in/first-out) buffer. A DSP, interfacing with the FPGA on its external memory interface, reads the FPGA FIFO buffer to capture the video data (Figure 1). Using timing information from the real application-software flow, the verification engineer can estimate the maximum time that the DSP can take between two consecutive FIFO-buffer reads. The engineer can then implement a test case with this DSP limitation in mind. The test records an error if the FIFO buffer is too shallow to buffer all the data that can arrive during the longest interval between two consecutive FIFO-buffer reads.

Problems in video-signal processing

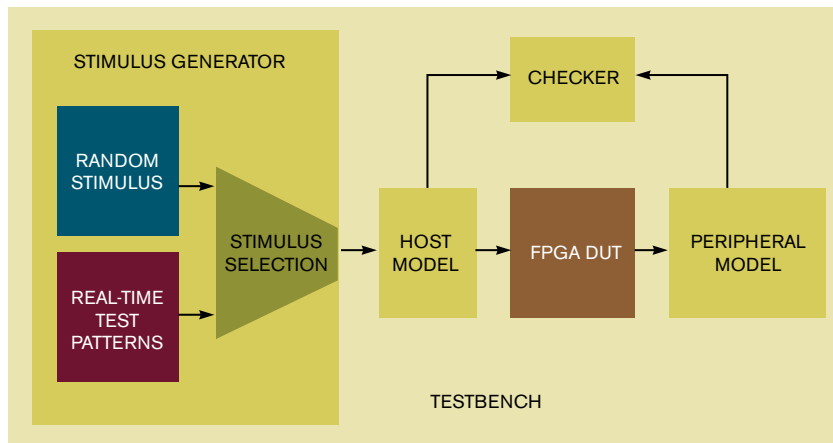


Figure 2 Using software or by capturing patterns directly from a system, verification engineers can catch design failures under real-time-test scenarios and fix them during verification.

can be data-dependent. So it can be important to use different types of video patterns during board bring-up. This approach helps ensure accurate video processing of any video-streaming application. With advances in the open-source community, verification engineers can readily find open-source software to generate such test patterns in raw-data format. Verification engineers can generate such video patterns as a raw-data file using software or by directly capturing patterns from a system (Figure 2). Adopting such methods, verification engineers can catch design failures under real-time-test scenarios and fix them during front-end verification itself, not when the customer happens to apply a pattern that breaks the system.

Verification and software teams should have a common set of test scenarios in the test plan at the beginning of the design cycle. The common set of test cases would ensure that there are no loopholes during board bring-up. Also, designers can target or rectify any of the implementation or integration errors early in the design cycle.

The complexity of FPGA designs requires designers to detect issues as early

as possible in the design cycle to avoid re-spins. The role of verification becomes important in reducing the number of faults arising during board bring-up. Careful consideration of system architecture along with working knowledge of the peripheral hardware will lead verification engineers to write test scenarios closer to the real-time application. These practices can lead to an effective verification effort that will ultimately ease board bring-up. **EDN**

AUTHORS' BIOGRAPHIES

Harshal Chhaya is an ASIC engineer at eInfochips, where he has worked for three years. He provides RTL (register-transfer-level) design and verification for FPGAs; SOC (system-on-chip) verification; and IP (intellectual-property) verification, development, and synthesis. Chhaya has a master's degree in electrical engineering from San Jose State University (San Jose, CA). His personal interests include Buddhist philosophy, the education process, and cricket. You can reach him at harshal.chhaya@einfochips.com.

Tarak Patel is an ASIC-verification engineer at eInfochips, where he has worked for three years. He defines and develops verification environments for ASIC and FPGA designs. Patel has a bachelor's degree in electronics and communications from Hemchandracharya North Gujarat University (Gujarat, India). His personal interests include listening to music. You can reach him at tarak.patel@einfochips.com.

Go to www.edn.com/090709df and click on Feedback Loop to post a comment on this article.

For more technical articles, go to www.edn.com/features.