Verification Mind Games

Jeffrey Montesano, Verilab
Montreal, Canada
Jeffrey.Montesano@verilab.com

Mark Litterick, Verilab
Munich, Germany
Mark.Litterick@verilab.com

Is it the verification environment’s duty to accurately replicate the real world?

Where is the line between inter-operating with the design and stressing it?

Gotta get all the bugs out by the Ape-out date

Is it worthwhile to target corner cases that designers consider invalid?

To what extent must the verification component follow the design protocol?

The protocol has bi-directional signals with the potential for multiple masters. Should I split each signal into two at the VC interface level?... Yes.

The protocol says that when an error condition is detected, the design must send a NACK packet. Should my VC automatically send NACK too?... No.

Can I snoop the design’s internal clock to synchronize my VC to?... No.

The design indicates FIFO fullness with signals “full” and “empty”. Can my VC or testcase make use of them to prevent overflow/underflow?... Yes, but only if checks are made on them.

Should I ask myself “what is it I’m trying to accomplish here?” when embarking on a new verification task... Yes.

The DUT is sending data without an accompanying clock - should my VC do Clock-Data-Recovery?... No.

A design has a register bit that defaults to ‘0’, and causes the DUT to enter low-power mode when written with ‘1’. Is writing ‘0’ when it is already at ‘0’ important to test?... Yes.

Is it sufficient to limit error injection to the scenarios for which the DUT has detection capabilities?... No.

A design draws circles of radius given by an input parameter. Is testing a radius of zero important?... Yes.

Is it my responsibility to ensure that the design architect and design engineer are on the same page?... Yes.

Is coverage closure more important than testcase passing rate?... Yes.

Should I be relying mainly on waveforms to debug my VC?... No.

Is it necessary to allow a simulation to continue running after it has encountered an error?... No.

Should I implement coverage on individual register values?... No.

Should I regularly step back from low-level implementation and take a high-level view of the verification effort as a whole?... Yes.

Is it worthwhile to target corner cases that designers consider invalid?

Gotta get all the bugs out by the Ape-out date

The protocol has bi-directional signals with the potential for multiple masters. Should I split each signal into two at the VC interface level?... Yes.

The protocol says that when an error condition is detected, the design must send a NACK packet. Should my VC automatically send NACK too?... No.

Can I snoop the design’s internal clock to synchronize my VC to?... No.

The design indicates FIFO fullness with signals “full” and “empty”. Can my VC or testcase make use of them to prevent overflow/underflow?... Yes, but only if checks are made on them.

Should I ask myself “what is it I’m trying to accomplish here?” when embarking on a new verification task... Yes.

The DUT is sending data without an accompanying clock - should my VC do Clock-Data-Recovery?... No.

A design has a register bit that defaults to ‘0’, and causes the DUT to enter low-power mode when written with ‘1’. Is writing ‘0’ when it is already at ‘0’ important to test?... Yes.

Is it sufficient to limit error injection to the scenarios for which the DUT has detection capabilities?... No.

A design draws circles of radius given by an input parameter. Is testing a radius of zero important?... Yes.

Is it my responsibility to ensure that the design architect and design engineer are on the same page?... Yes.

Is coverage closure more important than testcase passing rate?... Yes.

Should I be relying mainly on waveforms to debug my VC?... No.

Is it necessary to allow a simulation to continue running after it has encountered an error?... No.

Should I implement coverage on individual register values?... No.

Should I regularly step back from low-level implementation and take a high-level view of the verification effort as a whole?... Yes.

interface protocol_if (inputout data, inoutput clk);
logic clk_o = 1;
logic clk_i;
logic data_o = 1;
logic data_i;
assign (highz1,strong0) data = data_o;
assign (highz1,strong0) clk = clk_o;
assign data_i = data;
assign clk_i = clk;
endinterface: protocol_if