Project Design Verification Environment

To create a layered testbench architecture design in System Verilog for AHB-Lite Slave Protocol

3- JULY - 2023

Submitted By : Ammar Sarwar

10x Engineers (Pvt.) Ltd.

Transaction

1. Can I run predefined sequences? (e.g. reset sequence, random write sequence, random read/write bursts, a directed test). Can I debug easily if my test fails? Do I need one or multiple transactions for bursts?

Yes! I can run the predefined sequences by using a program block in a separate file you can define predefined sequences.

Yes! If my test fails I can debug it easily. I have added a method in each class named as **print_trans() ,print_drv(),** and **print_mon().**

For multiple transactions we actually require more than one transaction but for a single burst we actually need only one transaction.

Generator

2. How to control how many transactions get generated? Sometimes random transactions are not needed. How do I generate non-random transactions when required?

In my generator class I have added a variable named r**epeat_count,** that actually controls how many transactions to be generated by the generator,as shown in the **figure below.** We can actually set this variable to any number to create as many transactions that we want.

If we want to generate non-random transactions we can actually do this in the test file, in the **pre_randomize** function by first disabling the randomization for either a specific

property within the class or for all properties. Afterward, we can provide our desired values within the pre_randomize function of the program block.

```
class generator;
    transaction trans, tr;
    mailbox     gen2driv;
    int         repeat_count;
    event       gen_ended;

  function new( mailbox gen2driv, event gen_ended );
    this.gen2driv     = gen2driv;
    this.gen_ended    = gen_ended;
    trans             = new();
  endfunction

  task main();
    repeat (repeat_count) begin
      if ( !trans.randomize() )  $fatal("[Generator]: Trans randomization failed");
      tr = trans.do_copy();
      gen2driv.put(tr);
      $display("[Generator]: All packets sent to driver");
      end
    -> gen_ended;
  endtask

endclass
```

Driver

3. Are the interface signals driven according to the spec? Does the transaction have proper address/data Phases? Do I need to sample inputs to decide whether to drive outputs or not on the next clocking event?

Yes, all the interface signals are driven according to the specifications. **Yes,** each transaction has a proper address and data phases which can be observed by the waveform as well.
Yes, we need to sample inputs to decide whether to drive outputs in the next clocking event,because we will drive the signals if the HREADY signal is high.

Monitor

4. Are the interface signals sampled according to the spec? Does the transaction have proper address/data Phases?

Yes! The Interface signals were sampled according to the specifications, and each transaction has a separate address and data phases which can be observed through waveform also.

Scoreboard

5. Does the scoreboard implement proper endianness? How to change endianness if required? How to not compare reset values and to compare only those memory locations which have already been written? Should scoreboard memory be static or dynamic?

Yes! The scoreboard implements proper endianness, you can use a variable named as **big_endian** in the scoreboard to turn on and off the endianness. By default that variable is set to 0 which means that it will implement only the little endianness but if you want it to be big endianness you can change the value of that variable to 1,which can be changed in a test program as done in many tests.

Verification Plan

Test ID	Test Name	Test Status	Test Description	Stimulus Generation Procedure	Test Passes When	Test Fails When	Checking Procedure	Comments
1	test_hready_high	PASS	To insert wait states to check if require more time to provide or sample the data	Firstly the HReady signal randomization was off and then the HREADy signal was set to 1	Since the HREADY signal was set to high state all and slave select signal was set 0 the slave is not selected but the master is ready to recieve	When the HREADY signal goes to low state	Checks if the Hready signal is high the HTRANS must be constant	HREADY high signal shows the readiness of the target AHB component to accept data or complete a transfer.
2	test_slave_select	PASS	To check that slave is connected or not	Firstly the slave select signal randomization was turned off and then the HSEL signal was turned high means slave is connected	When the HSEL is high	when the HSEL is low	Checks if the HSEL is high means slave is connected or not, In our case slave is only 1 so this will always pass	HSEL indicates that the current transfer is intended for the selected slave
3	test_hprot	PASS	To checks that Protection control for Data Access only	For HPROT data access only the HPROT[0] should be 1 so it's randomization was turned off and the value was set manually	When the HPROT[0] is 1	When the HPROT is signal is other than 1	Checks if HPROT[0] is 1 if it is test passes else test fails	HPROT primarily intended for use by any module that wants to implement some level of protection.
4	test_htrans_idle	PASS	To Indicate that no data transfer is required	HTRANS state randomization was turned off and then HTRANS was set to idle state	when the HTRANS = `IDLE	If HTRANS is other than the IDLE state	Checks if the HTRANS state is of IDLE state if it is test passes else fails	HTRANS set to IDLE that a master uses when it does not want to perform a data transfer.
5	test_htrans_busy	PASS	Test to enable master to insert idle cycles in the middle of a burst.	HTRANS state randomization was turned off and then HTRANS was set to busy state	when the HTRANS = `BUSY	If HTRANS is other than the BUSY state	Checks if the HTRANS state is of BUSY state if it is test passes else fails	HTRANS set to BUSY indicates that the master is continuing with a burst but the next transfer cannot take place immediately .
6	test_non_seq_write_word	PASS	Checks the non sequence wirte word transfers	HTRANS state was set to NON_SEQ and HSIZE was set Word and HWRITE signal was set High, Using randomization 32 bit random data was stored at arbitrary address in the memory	When the data stored in the dut mem mathces the data stored in the local memory in the scoreboard	When the data stored in the mem does not matches with the data stored at the local mem	Checks that data stored in the local memory matches with data stored in the actual mem	This test checks that word stored in the form of little Endian in the actual mem matches with the data stored at local mem
7	test_non_seq_read_word	PASS	Checks that the data read from the actual memory matches with the data read from the local mem	HTRANS state was set to NON_SEQ and HSIZE was set Word and HWRITE signal was set low, data was read from the actual memory at random 32 bit address	When the data read in the local mem matches with the data read from the actual memory	When the data stored in the mem does not matches with the data stored at the local mem	Checks the data read between actual and local memory if matches test passes else fails	This test checks that word stored in the form of little Endian in the actual mem matches with the data stored at local mem
8	test_non_seq_write_halfword	PASS	Checks the non sequence wirte halfword in the actual and local memory	HTRANS state was set to NON_SEQ and HSIZE was set HALFWord and HWRITE signal was set High, and then at random address the halfwords was stored in the actual memory	When the halfword stored in the dut mem mathces the data stored in the local memory in the scoreboard	When the data stored in the mem does not matches with the data stored at the local mem	Checks that halfword stored in actual memory mathces the halfword stored in the local mem in the scoreboard	This test checks that halfword stored in the form of little Endian in the actual mem matches with the data stored at local mem
9	test_non_seq_read_halfword	PASS	Checks that the data read from the actual memory matches with the data read from the local mem	HTRANS state was set to NON_SEQ and HSIZE was set HALF Word and HWRITE signal was set low, data was read from the actual memory at random 32 bit address	When the data read in the local mem matches with the data read from the actual memory	When the data stored in the mem does not matches with the data stored at the local mem	Checks the data read between actual and local memory if matches test passes else fails	This test checks that half word stored in the form of little Endian in the actual mem matches with the data stored at local mem
10	test_non_seq_write_byte	PASS	Checks the non sequence wirte byte transfers	HTRANS state was set to NON_SEQ and HSIZE was set byte and HWRITE signal was set high, data was written from the actual memory at random 32 bit address	When the byte stored in the local mem matches the data stored in the actual memory	When the byte stored in the local mem does not matches the data stored in the actual memory	Checks the data read between actual and local memory if matches test passes else fails	This test checks that byte stored in the form of little Endian in the actual mem matches with the data stored at local mem
11	test_non_seq_read_byte	PASS	Checks that the data read from the actual memory matches with the data read from the local mem	HTRANS state was set to NON_SEQ and HSIZE was set byte and HWRITE signal was set low, data was read from the actual memory at random 32 bit address	When the byte read in the local mem matches the data stored in the actual memory	When the byte stored in the local mem does not matches the data stored in the actual memory	Checks the data read between actual and local memory if matches test passes else fails	This test checks that byte stored in the form of little Endian in the actual mem matches with the data stored at local mem
12	test_non_seq_write_big_endian_byte	PASS	Checks the non sequence wirte big endian byte transfers	HTRANS state was set to NON_SEQ and HSIZE was set byte and HWRITE signal was set low, data was read from the actual memory at random 32 bit address in the form of big Endian	When the data stored in the form of big endian in local mem mathces the data stored in the actual mem	When the data stored in the form of big endian in local mem does not mathces the data stored in the actual mem	By comparing the data stored in the actual and local mem	This test checks that byte stored in the form of BIG ENDIAN in the actual mem matches with the data stored at local mem
13	test_non_seq_write_big_endian_halfword	FAIL	Checks the non sequence wirte big endian HALFWORD transfers	Randomly generated Halfwords are stored at random address in the actual mem	When the data stored in the form of big endian in local mem mathces the data stored in the actual mem	Incorrect halfword is write/read to/from memory	By comparing the data stored in the actual and local mem	This test checks that halfword stored in the form of BIG ENDIAN in the actual mem matches with the data stored at local mem
14	test_non_seq_write_big_endian_word	FAIL	Checks the non sequence wirte big endian WORD transfers	Randomly generated words are stored at random address in the actual mem	When the data stored in the form of big endian in local mem mathces the data stored in the actual mem	Incorrect word is write/read to/from memory	By comparing the data stored in the actual and local mem	This test checks that word stored in the form of BIG ENDIAN in the actual mem matches with the data stored at local mem
15	test_non_seq_write_random	PASS	Stores the randomly byte,halfword or words at random address in the memory	Randomly generated byte,halfword or words are stored at random address in the actual mem	When the data written in actual and local mem matches	When the data written in actual and local mem does not matches	By comparing the data stored in the actual and local mem	Stores randomly byte,halfword or words at randomly generated address
16	test_non_seq_read_random	PASS	Read the byte,halfword or words in the memory	HTRANS is of non seq HSIZE can be of byte, halfword or words	When the data read in actual and local mem matches	When the data written in actual and local mem does not matches	By comparing the data stored in the actual and local mem	Read byte,halfword or words that were stored previously in the memory
17	test_seq_write_random	PASS	Stores the randomly byte,halfword or words at sequence of address in the memory	Randomly generated byte,halfword or words are stored at sequence of address in the actual mem	When the data written in actual and local mem matches	When the data written in actual and local mem does not matches	By comparing the data stored in the actual and local mem	This test checks that word stored in the form of Little ENDIAN in the actual mem matches with the data stored at local mem
18	test_seq_read_random	PASS	Read the byte,halfword or words in the memory in a sequence of address	Sequence of address was generated in the test case where randomly byte, halfword and words were stored	When the data read from the local mem matches the data in the actual mem	When the data read from the local mem does not matches the data in the actual mem	By comparing the data stored in the actual and local mem	It checks the data read from the sequence of address
19	test_burst_single_write_random	PASS	Only one transfer write takes place	The address change it self randomly after each transfer is written	if the write data matches in local and and actual memory	if the write data does not matches in local and and actual memory	Checks the data written between actual and local memory if matches test passes else fails	This test checks that word stored in the form of Little ENDIAN in the actual mem matches with the data stored at local mem
20	test_burst_single_read_random	PASS	Only one transfer reads takes place	The address change it self randomly after each transfer is read	if the write data read matches in local and actual memory	Write transfer occur during wait states	Checks the data read between actual and local memory if matches test passes else fails	This test checks that word read in the form of Little ENDIAN in the actual mem matches with the data stored at local mem
21	test_burst_wrap4_write	PASS	Address repeats itself after every 4 iteration HWDATA is written in the memory	HWRITE signal was turned on and randomly data is stored such that address repeats itself after every four iterations	Data written in actual and local mem matches	When data written in actual and local mem does not matches	By comparing the data stored in the actual and local mem	The data is stored sequentially and is wrapped around after four transfer
22	test_burst_wrap4_read	PASS	Address repeats itself after every 4 iteration and data is read form memory	HWRITE signal was turned on and randomly data is stored such that address repeats itself after every four iterations	Data read in actual and local mem matches	When data read in actual and local mem does not matches	By comparing the data stored in the actual and local mem	The data is read sequentially and is wrapped around after four transfer

23	test_burst_4incr_write	PASS	Address increments itself after each Iteration	Randomly generated data is stored at address in the memory which is incremented by four after each iteration	if the write data read matches in local and actual memory	if the write data read does not matches in local and actual memory	By comparing the data stored in the actual and local mem	The data is read sequentially and is incremented after each iteration
24	test_burst_4incr_read	PASS	Address increments itself after each Iteration	Randomly generated data is stored at address in the memory which is incremented by four after each iteration	if the read data read matches in local and actual memory	if the read data read does not matches in local and actual memory	By comparing the data stored in the actual and local mem	The data is read sequentially and is incremented after each iteration
25	test_waited_transfer	PASS	HREADY is turned of and on at differnt time intervals	HREADY signal was turned off when number of transaction is even else turned on	When the HREADY signal is high	HREADY == Low	Passes only when the HREADY is high	Checks that either the transfer takes only one cycle or it require more than one cycle
26	test_bonus_read_write	PASS	Data is read and written at the same address	when the data written in one cycles matches the data read in other cycle	when the data written in one cycles does not matches the data read in other cycle	when the data written in one cycle does not not matches the data read in other cycle	By comparing the data stored in the actual and local mem	Checks when data written previosuly matches data read in next cycle