ALU verification plan

Submitted to Eng. Hussein Galal

Omar Adel Abbas

11/2/2020

Si-vision hiring process

Basic scenarios for functional coverage

Scenario ID	Scenario	Description
Sc1	Perform basic operation on two random operands	Perform all operation specified by signals a_op, b_op, a_en and b_en on two random numbers and check the basic functionality for each operation • Perform operation on combinations of positive and negative numbers for arithmetic operations • Perform operation on large numbers to check the overflow behavioral for arithmetic operations
Sc2	Check the design behavioral when all operations enable signals are disabled	Set a_en and b_en signals to the '0 ' and verify the design behavioral (as the added requirement in list of assumptions)
Sc3	Check the NULL functionality in all sub_specs tables (negative test case)	Set a_op and b_op signals to the values correspond to NULL operation and verify the design behavioral (as the added requirement in list of assumptions)
Sc4	Toggle the reset signal	• Set the the rst_n signal to '0' and check the asynchronous property • Check the system behavioral after reset
Sc5	Toggle the enable signal	• disable the ALU_enable signal and read that the output value is zeros (as the added requirement in list of assumptions) • Enable the ALU_enable signal and verify that the output value is the operation specified by the operations selections (a_op, b_op, a_en, b_en)

Basic scenarios for code coverage

- Condition, decision and statement coverage are guaranteed during the functional coverage since the design is very simple
- No other coverage techniques are required (FSM coverage for example).

Testbench architecture

Testbench is a class based design follows the standard components as shown in figure 1 with random stimulus in the generator, self-checking mechanism in the scoreboard and coverage group to assure that the test is completed.

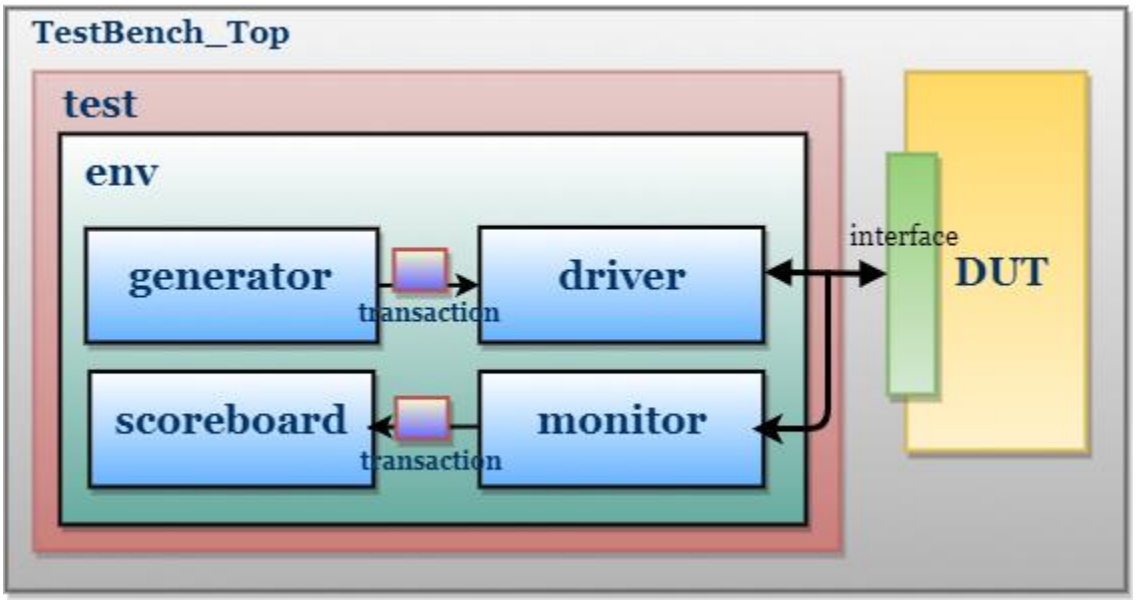

Figure 1: test bench architecture

Testbench components

Component	Functionality	notes
generator	Generating the stimulus by randomizing the transaction class Transaction class can be used as a placeholder for the inputs	Generates constrained random inputs Uses mailbox technique to pass the inputs to the driver The signals responsible for choosing the operation (a_op and b_op) will be generated as a cyclic random to aviod duplicated process
Driver	Receive the stimulus generated from the generator and drive to DUT by assigning transaction class values to interface signals.	Set a_en and b_en signals to the '0 ' and verify the design behavioral (as the added requirement in list of assumptions)
monitor	Send the sampled transaction to Scoreboard	Uses mailbox technique to communicate to the scoreboard
scoreboard	Receive samples from the monitor and	

	check for its correctness	
Environment	Environment is container class contains Mailboxes, Generator and Driver and all other components	

Evaluation metrics

The evaluation process and definition of done (DOD) will be controlled by a coverage group defined in coverage class which covers the following inputs:

- Positive and negative numbers for arithmetic operation
- All operations in all subspecs tables specified in the requirements
- Large values for overflow behavioral
- Transition between ‘0’ and ‘1’ for all control signal