An-Najah National University
Computer Engineering Department

VLSI Design Verification
Final assignment
Functional Coverage

Constraint-random testing (CRT) is very dominant in modern test bench and verification approaches. For CRT to be effective, it is usually combined with functional coverage to clearly quantify the how good is the CRT stimulus as a first step. Advanced functional coverage goes beyond identifying how some variables are driven by the TB or the DUT, but also identifying complex stimulus/output combinations and how well the TB drives these complex scenarios.

In this exercise, the goal is to experiment with functional coverage to find bugs.

Assignment weight: 5%

Instructions:

1. A black boxed DUT is provided which is similar to the design we used in the previous exercise, except the fact that it has an accumulate feature (either add or multiply). The **acc** bit enables the accumulate feature and the **func** bit decides whether it is an accumulate add write or accumulate multiply write:

```
module reg_ctrl
 # (
   parameter ADDR_WIDTH  = 8,
   parameter DATA_WIDTH  = 24,
   parameter DEPTH       = 256,
   parameter RESET_VAL   = 24'h123456      // Adjusted reset value
 )
 (
  input clk,
  input rstn,
  input [ADDR_WIDTH-1:0] addr,
  input sel,
  input wr,
  input acc,          // accumulate
  input func,         // function
  input [DATA_WIDTH-1:0] wdata,
  output reg [DATA_WIDTH-1:0] rdata,
  output reg ready
 );
```

2. Update the TB to include the following features:
    a. Make sure the TB has a transaction object (randomized) for driving the values to the DUT, Generator class connected to the Driver, Monitor and Checker classes.
    b. Coverage: a special component/class where the functional coverage covergroup are defined and instantiated and transactions are monitored for updating the coverage. Could be connected to the monitor using a mailbox.

3. Try to achieve 100% functional coverage only using pseudo random tests.

4. Verify that the DUT works as expected: try to avoid driving specific values to the DUT (always use randomized objects for driving).

5. Define

6. Cross coverage items.

Good Luck

Note1: It is possible to display the coverage using the get_coverage function to display all coverage for all covergroups at the end of the simulation.

Note2: Exercise is based on ASIC-World functional coverage pages: https://www.asic-world.com/systemverilog/coverage1.html. The associated SystemVerilog file can be used as a template: https://www.asic-world.com/code/systemverilog/coverage_class.sv

Note3: to extract a file: "tar -xvf $file", to compile a file "vlogan $file -sverilog", to build a tb "vcs top_module -sverilog"