Assignment 3

Based on the ALU design in Assignment 1 and Assignment 2 we will add Interface, code coverage and functional coverage.

Requirements:

1. Create the interface and top module.
2. Create a **do file** and use it to run the simulation, adding coverage options.
3. Add functional coverage coverpoints in a class called **ALU_coverage** to check:

- **Serial_in**
- **Direction**, and define bins for right and left.
- **Opcode**, make bin for every opcode operation (Addition, Subtraction, ..etc.)
- **a and b**,
    - Bin for maximum value
    - Bin for minimum value
    - Bin for *checkerboard pattern* (e.g., 101)
    - Bin for all other values
- Create a **cross** between **a** and **opcode**.
    - Define ignore or illegal bins for unneeded opcodes (hint: use binsof).
- Create a **cross** between **b** and **opcode**.
    - Define ignore or illegal bins for unneeded opcodes (hint: use binsof).
- Create a **cross** between **serial_in** and **direction**.
- Create a **cross** between **serial_in** and **opcode**.
    - Hint1: Define bins for **serial_in** and opcode **shifting**, using binsof and &&.
    - Hint2: Use ignore bins, use binsof and intersect.
        - *Note*: You can use either of the two hints.

    - If you used **Hint1**: After completing the assignment, perform the following step:
        - Use option.cross_auto_bin_max = 0 and observe what happens **before and after** in the coverage report.
- Create a **cross** between **direction** and **opcode**.
    - Define bins for both *direction* and opcode shifting operations (**Hint**: Use binsof and &&).
    - Also define bins for **opcode rotation**.
    - After completing the assignment, perform the following step:
        - Use option.cross_auto_bin_max = 0 and observe what happens **before and after** in the coverage report.

4. Generate the **code and functional coverage report**, and make sure to reach **100% code coverage** using either randomization or a directed testbench.

Deliver one pdf containing:

- Alsu_coverage class and any edits you make in others files of your project.
- Do file
- Code and functional coverage Report.

Hints for Coverage class:

1. Make the class inside a package has the Covergroup.
2. Make a Function inside the class called sample_cvg.
3. Inside sample_cvg Function, Call the sample method for the covergroup you created.
4. Import the Package, Instance and handle class in testbench, and call the Sample_cvg function using dot operator (e.g., class_inst.sample_cvg) on the **negedge of clk**, similar to what you did in the scoreboard.