1. What is the difference between Static and Dynamic testbench?

Static Testbench:

* A static testbench is predefined and fixed, with hard-coded stimulus, sequences, and expectations.
* It generally doesn’t change or adapt during the simulation.
* Limited flexibility since all inputs and checks are defined before the simulation starts.

Dynamic Testbench:

* A dynamic testbench is flexible and can generate random or constrained random stimulus.
* It can adjust behavior during simulation based on conditions, errors, or feedback.
* Typically uses SystemVerilog's random generation features (e.g., randc, constraint, randomize).

1. Why should we define default input and output skew?

In a digital design, skew refers to the difference in arrival times of signals, especially between clock signals or data inputs

Need for default skew:

* Helps ensure that signals are correctly aligned in time for setup and hold requirements.
* It ensures proper synchronization between the testbench components, such as between clocking and signal generation.
* By defining default skew, you help maintain predictable and correct simulation behavior.

1. How will you do error injection in SV?

Error injection is a method of introducing faults or errors into the design to verify the robustness of the system under fault conditions.

Techniques for Error Injection in SystemVerilog:

* Force and Release: You can use the force statement to override the value of a signal to simulate an error and then use release to return the signal to its original driving source
* Random Error Generation: You can introduce random errors by using the randomize function to modify a signal or variable’s value randomly.
* Fault Modeling: Use predefined fault models, such as stuck-at faults, in your testbench to simulate and check for error handling.
* Error flags: Inject errors by manipulating specific flags or signals that represent error conditions in the design.

1. What is the difference between local and global variables?

Local Variables:

* Declared inside a block, function, or task.
* Only accessible within that block, function, or task.
* Typically used for temporary storage and to avoid unintended interference with other parts of the code.

Global Variables:

* Declared at the module or class level, outside of specific tasks or functions.
* Accessible throughout the entire module or class.
* Used for shared state across different parts of the design or testbench.

1. What is the difference between static and dynamic casting?

Static casting:

* Performed at compile-time. It is used when the type relationship between two classes or types is known at compile time.
* It’s safer and faster since the compiler checks the type compatibility.
* Example

class A;

int data;

endclass

class B extends A;

int more\_data;

endclass

A a = new B();

B b = B'(a); // Static cast, works if 'a' is of type 'B'

Dynamic casting:

* Performed at runtime. It is used to check type relationships dynamically when the type is not known until execution.
* Requires an explicit check, such as if or is.
* Example

class A;

int data;

endclass

class B extends A;

int more\_data;

endclass

A a = new B();

if (a is B) begin

B b = a; // Dynamic cast, checked at runtime

end

1. How can you make a hierarchical connection with a scoreboard from a monitor?

To make a hierarchical connection between a monitor and a scoreboard:

* Monitor: Observes signals or data on the DUT and gathers the information required for checking.
* Scoreboard: Compares the observed behavior to expected values and keeps track of correct/incorrect data.
* Connection Mechanism
  + The monitor can send the captured data to the scoreboard using a queue, mailbox, or direct task/function calls.
  + For example, a monitor might push observed data into a queue, which the scoreboard can then pull from to perform checks:

// Monitor pushes data to the queue

monitor.my\_queue.push(observed\_data);

// Scoreboard pulls data from the queue for comparison

scoreboard.check\_data(monitor.my\_queue.pop());

1. What is the difference between Hard and Soft constraints?

Hard Constraints:

* Strict and cannot be violated during randomization.
* The randomization will fail if hard constraints cannot be satisfied.
* Used when certain conditions are absolutely critical (e.g., timing, signal ranges).

Soft Constraints:

* Flexible and can be violated if no valid random values satisfy them.
* Used when a constraint is desirable but not mandatory (e.g., desirability of a certain value range).

1. What is the difference between Pre randomize and Post randomize?

Pre randomize:

* Refers to actions that should happen before the randomization of variables.
* Useful for setting up or initializing variables or states prior to randomization.
* Example

class my\_class;

rand bit [3:0] value;

function void pre\_randomize();

value = 4'b1111; // Set initial value before randomization

endfunction

endclass

Post randomize:

* Refers to actions that should occur after the randomization.
* Used for additional processing or checking after random values have been assigned.
* Example:

class my\_class;

rand bit [3:0] value;

function void post\_randomize();

assert(value != 4'b0000); // Check after randomization

endfunction

endclass

1. Difference between for and foreach loop?

For loop:

* Used for iterating over a range or array with a known or fixed size.
* Example

for (int i = 0; i < 10; i++) begin

// Loop logic

end

Foreach loop:

* Specifically designed for iterating over arrays, whether dynamic or packed, and handles the size of the array automatically.
* More convenient for arrays of unknown or variable length.
* Example:

int arr[10];

foreach (arr[i]) begin

// Loop logic for each element in arr

end

1. Differences between Immediate and Deferred Immediate Assertions?

Immediate Assertions:

* Checked immediately when they are encountered in the simulation time.
* Useful for checking conditions right at the point of execution.
* Example:

assert (a == 1);

Deferred Assertions:

* Checked after the simulation time has passed and at a specific time event or trigger.
* These assertions are deferred to the next simulation cycle.
* Example:

assert property (@(posedge clk) a == 1);

1. How to establish communication between SV to any foreign language using DPI.

DPI (Direct Programming Interface) enables communication between SystemVerilog and foreign languages such as C or C++.

Steps:

* Export from SystemVerilog: Declare a foreign function in SystemVerilog that can be called from C.

export "DPI" function void c\_function();

* Import from C: Use import in SystemVerilog to access foreign language functions.

import "DPI" function void sv\_function();

* Create the C function: Define the foreign language function in the C code and ensure the linkage between the two.

#include "svdpi.h"

void c\_function() {

// C function body

}