1. What is the use of always\_ff?

The always\_ff block is a special form of the always block introduced in SystemVerilog, used to describe sequential logic. It ensures that the block is triggered only on edge-sensitive events (such as rising or falling edges of a clock signal), and prevents the inclusion of non-edge-sensitive assignments (like combinational logic).

* It ensures that the block only responds to clock edges or other edge-sensitive events.
* It is specifically used for flip-flops or latches and ensures proper synthesis for sequential elements.
* Helps avoid common errors like accidental latch inference.

1. What is a callback?

A callback is a mechanism in SystemVerilog that allows the dynamic invocation of a function or method at a later time. Callbacks are commonly used for extending functionality, such as in testbenches, or when you need to dynamically modify behavior during simulation. Callbacks are often used in classes or randomization to dynamically modify the behavior of testbenches or to hook additional functions to a method.

Example:

class MyClass;

function void do\_something();

// Code here

endfunction

// Callback function

function void callback\_function();

$display("Callback called");

endfunction

endclass

// Hooking a callback

MyClass obj;

obj = new();

obj.set\_callback(callback\_function); // Register callback

1. What is a clocking block?

A clocking block in SystemVerilog is used to define the synchronization of signals with respect to a clock. It defines timing constraints for signals in a testbench and simplifies the modeling of timing behavior in a clocked domain. It is used to specify the timing relationships for signals in a module, ensuring proper synchronization with the clock signal (e.g., setup/hold time).

clocking cb @(posedge clk);

input reset, data\_in;

output data\_out;

endclocking

1. What is a program block?

A program block in SystemVerilog is used for writing testbench code. It contains simulation code such as stimulus generation, assertions, and other verification components, and is separated from design code (like modules and interfaces).

* Contains verification logic such as assertions, covergroups, and stimulus generation.
* Enforced synchronization with the clock and other verification components.
* Can be compiled separately from the design code.

1. What is the difference between program block and module?

Program

* Describes the testbench (verification logic).
* Used for simulation and verification.
* Cannot instantiate other program blocks or modules.
* Synchronized to the simulation time.

Module

* Describes the design (hardware logic).
* Can instantiate other modules and interfaces.
* Has ports like input, output, and inout.

1. What is a final block?

A final block is used for finalization tasks after the simulation has completed. It is typically used to display simulation results, perform clean-up tasks, or check the state of the design at the end of simulation.

1. What are input and output skews in clocking block?

In a clocking block, the input and output skews define the timing relationship between the clock signal and the signals defined within the clocking block. Skew ensures that signals meet the setup and hold times correctly.

* Input Skew: The amount of time after the clock edge that an input signal is valid.
* Output Skew: The amount of time before the clock edge that an output signal is driven.

1. How to take an asynchronous signal from clocking block?

In SystemVerilog, to model an asynchronous signal inside a clocking block, you typically set the clocking block's edge to an arbitrary clock (or posedge of any signal), and then sample or drive the asynchronous signal within the block.

1. Difference betwee final and initial blocks?

Initial block

* Executes once at the start of simulation.
* Typically used to initialize variables or start the simulation stimulus.
* Runs before any other code is executed.

Final block

* Executes after the simulation has completed.
* Typically used for final checks, coverage reporting, and clean-up tasks.
* Runs after all other simulation events have finished.

1. Why is an always block not allowed inside a program block?

An always block is not allowed inside a program block because program blocks are meant for simulation logic and synchronization, while always blocks are part of the design description for hardware. This separation maintains a clear distinction between simulation and design logic in SystemVerilog.

1. How to implement clock in program block ?

In a program block, you can implement a clock by either using a generated clock or by explicitly defining a clock signal within the initial block.

Example:

program test;

logic clk;

// Clock generation

initial begin

clk = 0;

forever #5 clk = ~clk; // Toggle clock every 5ns

end

// Stimulus or test logic here

endprogram

1. Write SV code to wait for a random delay in the range of 100 to 500ns.

initial begin

int delay;

delay = $random % 401 + 100; // Random delay between 100 and 500 ns

$display("Waiting for %0d ns", delay);

#delay; // Wait for the generated delay

$display("Delay complete!");

end