1. What is an objection in uvm?

In UVM (Universal Verification Methodology), an objection is a mechanism used to control the simulation flow, ensuring that certain processes (like starting and stopping a simulation) occur only when specific conditions are met. Objections allow a component to "hold" the simulation until it has completed its tasks, ensuring that the simulation doesn't prematurely finish while some critical tasks are still running. This is especially useful in complex verification environments where various components may need to complete certain activities before the simulation can end.

1. Explain the following methods:
   1. raise\_objection

The raise\_objection method is used to raise an objection in UVM, signalling that a component requires more time to complete its tasks and that the simulation should not finish until the objection is dropped.

Example:

// Raise an objection to prevent the simulation from ending

uvm\_objection.raise\_objection("Task is still in progress");

* 1. drop\_objection

The drop\_objection method is used to drop the objection that was previously raised by calling raise\_objection. Dropping an objection signals that the component has completed its task and that the simulation can continue or terminate if no other objections remain.

Example:

// Drop the objection when the task is complete

uvm\_objection.drop\_objection("Task is complete, simulation can proceed.");

* 1. set\_drain\_time

The set\_drain\_time method allows you to set a time (in simulation time units) that UVM will wait before forcibly ending the simulation, even if some objections remain. This is useful for preventing the simulation from waiting indefinitely in case there are unforeseen issues.

Example:

// Set a drain time of 100ns before forcing the simulation to end

uvm\_objection.set\_drain\_time(100ns);

1. Explain the following methods to check objection status:
   1. get\_objection\_count

The get\_objection\_count method returns the current number of objections that have been raised for a particular component. It represents how many components (or tasks) are still blocking the simulation from ending by keeping objections raised.

Example:

int count;

count = uvm\_objection.get\_objection\_count();

$display("Current objection count: %0d", count);

* 1. get\_objection\_total

The get\_objection\_total method returns the total number of objections (from all components) raised in the entire simulation environment. This can help determine if any objections are still pending from other components outside the current one.

Example:

int total;

total = uvm\_objection.get\_objection\_total();

$display("Total objections raised across the environment: %0d", total);

* 1. get\_drain\_time

The get\_drain\_time method returns the amount of simulation time that UVM will wait before forcibly terminating the simulation (if there are still objections). This helps you understand how much time is left before UVM will ignore objections and finish the simulation.

Example:

time drain\_time;

drain\_time = uvm\_objection.get\_drain\_time();

$display("Remaining drain time before forced simulation termination: %0t", drain\_time);

* 1. display\_objections

The display\_objections method provides a way to display detailed information about all active objections in the simulation environment. It can be used to debug or monitor the status of objections raised by different components.

Example:

// Display the state of all objections in the environment

uvm\_objection.display\_objections();

1. Explain the following callback hooks:
   1. raised()

The raised() callback method is a UVM callback hook that is called when an objection is raised for the component. This method can be overridden in a derived class to perform specific actions when an objection is raised.

Example:

// Overriding the raised() callback to perform custom actions

class my\_agent extends uvm\_agent;

// Callback method to handle when an objection is raised

function void raised(string id);

$display("Objection raised: %s", id);

endfunction

endclass

* 1. dropped()

The dropped() callback method is called when an objection is dropped for a component. This callback can be overridden to perform specific actions when a component finishes its task and no longer requires time.

Example:

// Overriding the dropped() callback to perform custom actions

class my\_agent extends uvm\_agent;

// Callback method to handle when an objection is dropped

function void dropped(string id);

$display("Objection dropped: %s", id);

endfunction

endclass

* 1. all\_dropped()

The all\_dropped() callback method is called when all objections in the simulation have been dropped, indicating that no more tasks are pending and that the simulation can now proceed to completion.

Example:

// Overriding the all\_dropped() callback to perform final actions

class my\_test extends uvm\_test;

// Callback method to handle when all objections have been dropped

function void all\_dropped();

$display("All objections dropped, simulation can now end.");

endfunction

endclass