1. What is a mailbox?

A mailbox in SystemVerilog is a synchronization and communication mechanism used for passing messages between processes or threads. It is essentially a FIFO (First In First Out) queue, which allows one thread to send messages to another. A mailbox ensures thread-safe access to the shared messages.

* Methods: put(), get(), try\_put(), try\_get(), etc.
* Usage: Mailboxes are commonly used in producer-consumer problems, where one thread produces messages and another consumes them.

1. How to count the number of elements in a mailbox?

mailbox m;

int num\_elements;

initial begin

m = new();

m.put(1); // Adding element to mailbox

m.put(2);

num\_elements = m.num(); // Get the number of elements

$display("Number of elements in mailbox: %0d", num\_elements);

end

1. What are semaphores & in which scenarios it is used?

A semaphore is a synchronization primitive used to control access to a shared resource by multiple threads. It acts as a signaling mechanism, allowing one or more threads to wait for or signal the availability of a resource.

Usage:

* Controlling access to a shared resource (e.g., memory or hardware).
* Critical Section Protection: Preventing race conditions.
* Thread Synchronization: Ensuring threads execute in a specific order.

1. What is the difference between a mailbox and a queue?

Mailbox

* A mailbox is a synchronization mechanism that allows communication between threads.
* It is a FIFO structure, but it is designed for thread-safe communication and typically supports only message passing.
* Methods: put(), get(), try\_put(), try\_get().

Queue

* A queue is a generic dynamic array that also follows a FIFO structure.
* It is not inherently thread-safe; synchronization needs to be handled manually if used across multiple threads.
* Methods: push\_back(), pop\_front(), size(), etc.

1. What is mutex?

A mutex (short for mutual exclusion) is a synchronization primitive used to protect critical sections in multi-threaded environments. It ensures that only one thread can access a shared resource at any given time. It is typically used when there is a need to prevent race conditions by ensuring exclusive access to a resource.

1. Explain how messages are handled.

Messages in SystemVerilog are handled using mailboxes and queues. A producer thread puts a message into a mailbox, and a consumer thread gets the message from the mailbox.

Example:

mailbox m;

int msg;

initial begin

m = new();

// Producer thread

fork

m.put(42); // Putting a message

join

// Consumer thread

fork

m.get(msg); // Getting the message

$display("Received message: %0d", msg);

join

end

1. Explain with examples how can we copy and compare arrays in System Verilog.

int arr1[5] = '{1, 2, 3, 4, 5};

int arr2[5];

initial begin

$copy(arr1, arr2); // Copy arr1 to arr2

$display("arr2[0] = %0d", arr2[0]);

end

if (arr1 == arr2) begin

$display("Arrays are equal");

end else begin

$display("Arrays are not equal");

end

1. Explain the linked list with an example.

A linked list is a dynamic data structure where each element (node) contains data and a reference (link) to the next element in the list. It is used when the size of the list is unknown, or when elements need to be inserted/deleted dynamically

Example:

class Node;

int data;

Node next;

endclass

module linked\_list\_example;

Node head, current;

initial begin

// Creating a linked list

head = new();

head.data = 1;

head.next = new();

head.next.data = 2;

head.next.next = new();

head.next.next.data = 3;

head.next.next.next = null;

// Traversing the list

current = head;

while (current != null) begin

$display("Node data: %0d", current.data);

current = current.next;

end

end

endmodule

1. What are fixed-size arrays in SV? Explain its applications.

A fixed-size array is an array with a predefined size, which cannot be changed during simulation. It is used when the size of the array is known in advance.

Applications

* Memory-mapped registers: Where the size and the structure of the array are predefined.
* Constant data: When working with known values.

1. How to find indexes associated with associative array items?

int assoc\_array[string];

string key;

initial begin

assoc\_array["apple"] = 10;

assoc\_array["banana"] = 20;

foreach (assoc\_array[key]) begin

$display("Key: %s, Value: %0d", key, assoc\_array[key]);

end

end

1. Why is reactive scheduler used?

The reactive scheduler in SystemVerilog ensures that processes (e.g., always blocks) are executed only when specific conditions or events trigger them. This scheduler reacts to changes in signals or variables, making the simulation more efficient by avoiding unnecessary evaluations. It is used in event-driven simulations, where the order of process execution is determined by the occurrence of events, rather than fixed times or cycles.

1. Explain the use of typedef with an example.

typedef is used to create type aliases. It allows you to define a new name for an existing data type or structure, making code easier to read and maintain.

Example:

typedef int my\_int; // Define my\_int as an alias for int

my\_int x; // Now x is of type int, but using the alias

x = 10;

$display("x = %0d", x);

1. What is a struct?

A struct in SystemVerilog is a user-defined composite data type that groups different data types into a single unit. It allows you to model complex data.

Example:

typedef struct {

int id;

string name;

bit [7:0] value;

} Sensor;

Sensor s1; // Declaring a variable of type Sensor

initial begin

s1.id = 1;

s1.name = "Temp Sensor";

s1.value = 100;

$display("Sensor ID: %0d, Name: %s, Value: %0d", s1.id, s1.name, s1.value);

end

1. Explain Union with an example.

A union is similar to a struct, but it allows storing multiple data types in the same memory location. Only one member of the union can hold a value at any time.

Example:

typedef union {

int i;

real r;

string s;

} Data;

Data d1;

initial begin

d1.i = 42; // Only one member can hold a value at a time

$display("Union value: %0d", d1.i);

d1.r = 3.14; // Overwrites the previous value

$display("Union value: %0f", d1.r);

end