**CONSTRAINTS**

**Basic Questions**

1. **What are constraints in SystemVerilog? Why are they used?**
2. **What is the difference between a soft constraint and a hard constraint?**
3. **Explain the usage of the randc keyword. How is it different from rand?**
4. **What is a default constraint, and how is it declared?**
5. **What is constraint inheritance? Provide an example.**
6. **How do you apply randomization to class objects in SystemVerilog?**
7. **What is the use of the inside operator in constraints?**
8. **What are inline constraints? Provide an example.**

**Intermediate Questions**

1. **How can you solve over-constrained scenarios in SystemVerilog?**
2. **What are the methods to ensure randomization meets specific functional coverage?**
3. **Explain the purpose of the dist operator. How is it used in constraints?**
4. **What happens if constraints are conflicting or unsatisfiable? How can you debug such cases?**
5. **What is random stability, and how can it be ensured in SystemVerilog?/**

ability to produce the same random number in multiple simulations. It can be possible through random seed($urandom(seed)), and randomization with inline constraints,

1. **How can you constrain the size of a dynamic array during randomization?**

Data[];

Constraint c1 {Data.size();}

1. **How do you exclude specific values from being randomized? Provide an example.**

**Rand int var1;**

**Int exclude\_value = 25;**

**Constraint c1 {var1 != 25;}**

1. **What is the pre\_randomize and post\_randomize function? Where is it used?**
2. **Explain constraint block ordering. How does SystemVerilog resolve multiple constraints applied to the same variable?**

No sequential order to solve the multiple constraints on the same variable but all constraints treated as conjunction (&&). Solver takes all the constraint together and work as a unified system

Constraint c1 {var1>5;}

Constraint c2 {var2 <10;} ---------🡪 constraint c1 {var1 >5 && var2<10;}

**Advanced Questions**

1. **How can you apply constraints conditionally in SystemVerilog?**
2. **What is the difference between static and dynamic constraints? How are they used in UVM?**
3. **What are weighted constraints? Provide an example using the dist operator.**
4. **Explain constraint solve-before relationships. Provide an example of when to use it.**
5. **How do you handle situations where randomization fails due to highly complex constraints?**
6. **How do random sequence generation constraints differ from standard randomization constraints?**
7. **How can you randomize associative arrays and queues?**
8. **Explain the importance of constraint coverage. How do you measure it?**
9. **What are user-defined constraints? How do they improve randomization flexibility?**
10. **How can you dynamically modify constraints during simulation?**
11. **What is the difference between inline constraints and constraint blocks in SystemVerilog?**
12. **How can you randomize class members with interdependencies? Provide an example.**

**Scenario-Based and Practical Questions**

1. **Write a constraint to randomize a 4-bit value that excludes the values 0x3 and 0xA.**
2. **Given two integers, a and b, write constraints to ensure:**
   * **a is always greater than b.**
   * **The difference between a and b is always an odd number.**
3. **Write a constraint to randomize an array of integers such that:**
   * **The array size is between 5 and 10.**
   * **All elements are unique.**
4. **How would you constrain a sequence of numbers to form a Fibonacci series?**
5. **Write constraints to randomize an IP address structure containing address, subnet mask, and gateway. Ensure the address is always in the same subnet as the gateway.**
6. **Explain how you would debug failing constraints in a UVM testbench.**
7. **How can you apply constraints selectively to specific objects in a large-scale testbench?**
8. **Write a constraint to ensure random values are divisible by 3.**

**Data Structure and Arrays**

1. **Randomize a dynamic array such that:**
   * The array contains between 5 and 10 elements.
   * The elements are even numbers.
   * The sum of all elements is less than 100.
2. **Write constraints to randomize a 2D array where:**
   * The size of the array is m x n.
   * Each row contains unique values.
   * The elements in each column are in ascending order.
3. **Create constraints for a queue of integers such that:**
   * The queue size is always less than 10.
   * All elements are prime numbers.
4. **Randomize a structure that represents a packet. Ensure:**
   * The header size is between 4 and 8 bytes. Header inside {[4:8]}
   * The payload size is between 64 and 128 bytes. Payload\_size inside {[64:128]}
   * The total packet size (header + payload) is a multiple of 16.(header+payload)%16 ==0
5. **Randomize an associative array where the keys are strings, and the values are integers. Ensure:**
   * Each key has a unique value.
   * All values are between 10 and 50.

**Protocol Verification**

1. **Randomize a transaction object for a memory read/write operation such that:**
   * For a read operation, the address is always aligned to 4 bytes.
   * For a write operation, the data size is always a multiple of 8 bytes.
2. **Randomize an AMBA AXI transaction object such that:**
   * The burst size is between 1 and 16.
   * The starting address is aligned to the burst size.
   * The burst type is fixed, incrementing, or wrapping (randomly chosen).
3. **Randomize a packet for an Ethernet frame:**
   * The source MAC address and destination MAC address are different.
   * The payload size is between 64 and 1500 bytes.
   * The checksum value is valid for the given payload.
4. **Randomize a USB transaction object ensuring:**
   * The packet type can be control, bulk, interrupt, or isochronous.
   * The transfer size is aligned to 64 bytes for bulk transfers.

**Dependency and Relationships**

1. **Write constraints to randomize two integers, a and b, such that:**
   * a + b is always an odd number.
   * a is a multiple of b.
2. **Randomize two variables, start\_time and end\_time, ensuring:**
   * start\_time is always less than end\_time.
   * The duration (end\_time - start\_time) is a random value between 5 and 20 units.
3. **Randomize three signals x, y, and z such that:**
   * x + y = z.
   * x, y, and z are non-zero values.
4. **Create constraints to randomize a lottery system where:**
   * There are 5 winning numbers.
   * All numbers are unique and between 1 and 50.
   * The sum of the winning numbers is greater than 100.

**Real-World Applications**

1. **Randomize a calendar date object such that:**
   * The month has 30 or 31 days (no February).
   * The day is valid for the given month.
   * The date is not on a weekend.
2. **Randomize an employee record containing:**
   * Employee ID (unique integer).
   * Department (chosen randomly from a list of departments).
   * Salary (random value between $50,000 and $100,000, depending on the department).
3. **Randomize an IP address object ensuring:**
   * The address is in the range 192.168.x.x.
   * The first three octets are fixed, and the last octet is random.
4. **Create constraints for an airline ticket system where:**
   * The seat number is unique within the flight.
   * The class is economy, business, or first class (randomly selected).
   * The price depends on the class chosen.

**Complex Scenarios**

1. **Randomize a Sudoku puzzle generator:**
   * Ensure the grid is a valid 9x9 Sudoku board.
   * At least 17 numbers are pre-filled.
2. **Randomize a chessboard configuration such that:**
   * The placement of all pieces is valid for the start of a chess game.
   * The king and rooks are always in their initial positions.
3. **Randomize a configuration for a robotic arm with 3 joints such that:**
   * Each joint angle is between 0° and 180°.
   * The combined reach of the arm is less than 2 meters.
4. **Randomize a sequence of events for a traffic signal controller:**
   * The green signal duration is between 20 and 60 seconds.
   * The yellow signal duration is fixed at 5 seconds.
   * The red signal duration is calculated based on the remaining time in the cycle.

**Debugging and Optimization**

1. **How would you constrain random values for a test case where previous randomization failed due to over-constraining?**
2. **Write constraints to generate test inputs for a Finite State Machine (FSM):**
   * Ensure valid state transitions.
   * Randomize inputs that trigger transitions between states.
3. **Create constraints for randomizing test cases in a scoreboard:**
   * Ensure the input and output transactions match in timing and order.
   * The output depends on random but valid inputs.

**Open-Ended Challenges**

1. **Design constraints for a packet generator that mimics real-world traffic:**
   * Randomly choose between TCP, UDP, and ICMP packets.
   * For TCP packets, ensure the sequence numbers increment correctly.
   * For UDP packets, ensure valid checksum generation.
2. **Randomize a test case for a vending machine:**
   * The machine accepts coins of 1, 5, 10, and 25 cents.
   * The total inserted amount should match the price of the item selected.
3. **Randomize a testbench scenario for power management:**
   * Ensure voltage levels are randomly selected within valid ranges.
   * Ensure the power-up sequence is completed before power-down.