80 System Verilog Interview Questions

Logical, Tricky, and Brainstorming Topics

Compiled by: Kittu Patel

May 5, 2025

Contents

1 Data Types and Casting

1. Explain the differences between ‘logic‘, ‘reg‘, ‘wire‘, ‘bit‘, and ‘byte‘ in SystemVerilog.
2. How does SystemVerilog handle signed versus unsigned arithmetic by default?
3. Describe the four states of a 4-state data type and how they differ from a 2-state type.
4. What are implicit casts and when might they lead to unexpected behavior?
5. Compare and contrast ‘$cast‘ and ‘cast<type>()‘. Provide use-case examples.
6. How would you define a packed ‘struct‘ and how does alignment affect it?
7. Explain the behavior of ‘union‘ in SystemVerilog, especially regarding overlapping memory.
8. What is the difference between static and automatic variables inside functions and tasks?
9. How does SystemVerilog manage overflow and underflow in arithmetic operations?
10. When and why would you use ‘typedef‘ for parameterized types?

2 Processes and Concurrency

1. Describe the differences between ‘always_ff‘, ‘always_comb‘, and ‘always_latch‘.
2. Explain blocking (‘=‘) versus non-blocking (‘<=‘) assignments and typical use-cases.
3. How do delta cycles and event scheduling work in SystemVerilog simulation?
4. What are race conditions, and how can you avoid them?
5. How do you generate an automatic sensitivity list for combinational logic?
6. Explain the semantics of ‘fork...join‘, ‘fork...join_any‘, and ‘fork...join_none‘.
7. How can you safely terminate or disable a running process?
8. Discuss the use of explicit ‘@‘ event control versus sensitivity lists.
9. What mechanisms exist for prioritizing processes or events?
10. How would you use an unresolved ‘wire‘ to model multiple drivers?

3 Interfaces and Modports

1. What is a SystemVerilog ‘interface‘ and what problems does it solve?
2. How do you connect an ‘interface‘ instance to a module port?
3. Explain the purpose of ‘modport‘ and how it restricts signal access.
4. Describe ‘clocking‘ blocks within an interface and their benefits.
5. When and how would you use a ‘virtual interface‘?
6. How does the ‘bind‘ statement work with interfaces?
7. What is a parameterized interface, and how do you declare one?
8. How do you package and reuse an interface across multiple modules?
9. Compare using an ‘interface‘ versus individual port lists.
10. How can you synchronize reset and clock using constructs in an interface?

4 Constraints and Randomization

1. Explain the difference between ‘rand‘ and ‘randc‘ variables.
2. How do you constrain a random variable to a specific range?
3. What happens when two constraints conflict? How do you debug it?
4. Describe soft constraints and their typical use-cases.
5. Outline the solver phases in SystemVerilog randomization.
6. How do you randomize dynamic arrays and queues?
7. Explain conditional constraints with an example.
8. What is constraint coverage and why is it useful?
9. How can you seed the randomization process for reproducibility?
10. Describe using ‘solve...before‘ to order constraint evaluation.

5 Classes and Object-Oriented Programming

1. How do you declare a class and create an object instance in SV?
2. What are static class members and when would you use them?
3. Explain inheritance and polymorphism in SystemVerilog.
4. What is a virtual method and how is it declared?
5. How do you call a parent class constructor using ‘super‘?

6. Describe type parameterization in classes with an example.
7. When would you declare a method as ‘pure virtual‘?
8. How is ‘clone()‘ used to duplicate class objects?
9. Discuss garbage collection and object lifetime management.
10. How can you implement a factory pattern in SV classes?

6 Functional Coverage

1. Define functional coverage and its role in verification.
2. How do you declare a ‘covergroup‘ and ‘coverpoint‘?
3. Explain bins and wildcard bins in coverage.
4. How do you cross two ‘coverpoints‘?
5. What is coverage sampling and how is it performed?
6. How do you control coverage options like ‘option.weight‘?
7. Describe using coverage directives to exclude illegal states.
8. How can you report coverage holes programmatically?
9. Discuss strategies to achieve coverage closure.
10. How do you integrate coverage data into a testbench flow?

7 Sequences and Drivers

1. Explain how you would generate a sequence of transactions using tasks and ‘fork...join‘.
2. How do you randomize the order of transactions in a sequence?
3. Describe using a clocking block to synchronize sequence items.
4. How would you implement timeouts within a sequence to avoid hangs?
5. Explain the use of ‘mailbox‘ for communication between sequence and driver.
6. How can you parameterize a driver task for multiple test scenarios?
7. Discuss handling back-to-back transactions in a pipelined driver.
8. How do you monitor and check transaction timing in a driver?
9. What techniques can you use to pause and resume a driver task on an event?
10. How do you implement a variable delay between transactions?

8 Scoreboarding and Analysis

1. What is a scoreboard and why is it critical in verification?
2. How do you implement a scoreboard using associative arrays?
3. Describe comparing DUT output to a reference model.
4. How can you handle out-of-order transactions in a scoreboard?
5. Explain techniques for tolerant comparison (e.g., bitmasks).
6. How do you reset and reinitialize scoreboard state between tests?
7. Discuss integrating functional coverage within a scoreboard.
8. How can you generate a summary report of mismatches?
9. What strategies exist for concurrent scoreboarding across agents?
10. How do you debug and trace scoreboard failures effectively?