DAY-75
#100DAYSRTL

"System Verilog:-Randomization"

"Introduction":-

Explore System Verilog's prowess in designing complex digital systems, focusing on its dynamic randomization feature. From syntax insights to practical examples, this article navigates the basics, revealing how controlled randomness enhances flexibility and aids in uncovering design flaws. Stay tuned for advanced constraint discussions.

"Randomization":-

- System Verilog employs randomization to assign unpredictable values to variables or objects, serving a dual purpose.
- It generates dynamic inputs for the DUT and introduces diversity into testbench elements such as monitors and checkers.
- This process enhances code coverage and unveils elusive corner cases, unraveling unexpected behaviors for comprehensive testing.
- Test benches demand extensive stimuli for the DUT, making directed scenarios impractical and prone to missing corner cases.
- Randomization offers a succinct solution, enabling the creation of unpredictable and realistic test scenarios without extensive code.
- To declare a variable as random we use **rand** or **randc** keyword

“rand”:-

- The **rand** keyword means that the variable or object can take any value within its range
- the same value may repeated until the possible random values have been used.

“randc”:-

- The **randc** keyword means that the variable or object can take any value within its range, but not the same value until the possible random values have been used.
- Using **randc** ensures that we get unique random values.

“randomize()”:-

- every class incorporates the built-in "randomize" method, facilitating variable randomization. If this method remains uninvoked, variables retain their default values.
- This design choice permits the utilization of random variables with specified values without randomization. Moreover, the "randomize" method doubles as an error indicator by returning the randomization status.
- This feedback proves invaluable, offering insights into unsuccessful randomization caused by conflicting constraints or solver-related issues, enabling effective error handling in the design and verification process.

“Code practising”:-

```
class randomization;
  rand bit [3:0] a;
  randc bit [3:0] b;
  function display();
    $display("The value a:-%b",a);
    $display("The value b:-%b",b);
  endfunction
endclass:randomization
module tb;
  randomization obj;
  initial begin
    $display("The randomly generated values are :-");
    obj=new();
    for(int i=0;i<10;i++) begin
      obj.randomize();
      obj.display();
    end
  end
endmodule
```

“Result”:-

```
The randomly generated values are :-
The value a:-0110
The value b:-1000
The value a:-0011
The value b:-0101
The value a:-0100
The value b:-0000
The value a:-1111
The value b:-1111
The value a:-1101
The value b:-1101
The value a:-1011
The value b:-0010
The value a:-1000
The value b:-1001
The value a:-0111
The value b:-1010
```

“Disable randomization”:-

- We can disable randomization for any of the random variables using the rand_mode() method. This ensures that we have finer granularity in randomizing variables and randomization for some variables can be enabled/disabled for specific scenarios.
- **<var>.rand_mode(0);** will disable the randomization
- **<var>.rand_mode(1);** will enable randomization for variable

“Code practising”:-

```
class randomization;
  rand bit [3:0] a;
  randc bit [3:0] b;
  function display();
    $display("The value a:-%b",a);
    $display("The value b:-%b",b);
  endfunction
endclass:randomization
module tb;
  randomization obj;
  initial begin
    $display("The randomly generated values are :-");
    obj=new();
    for(int i=0;i<2;i++) begin
      obj.randomize();
      obj.display();
    end
    obj.rand_mode(0);
    $display("After disabling the randomization");
    for(int i=0;i<2;i++) begin
      obj.randomize();
      obj.display();
    end
  end
endmodule
```

“Result”:-

```
The randomly generated values are :-
The value a:-0110
The value b:-1000
The value a:-0011
The value b:-0101
After disabling the randomization
The value a:-0011
The value b:-0101
The value a:-0011
The value b:-0101
Simulation has finished. There are no more test vectors to simulate.
```

“Static Randomization”:-

SV also provides a static randomization function which can be called from any class and can be used to randomize variables. It is different from the normal randomization function as in this we explicitly pass the variables which we want to randomize. Even variables which are declared as random can be used with this static function.

“Code Practising”:-

```
module test();
    bit [15:0] addr;
    initial begin
        repeat(5) begin
          std::randomize(addr) with { addr < 50; };
            $display("addr = %0d", addr);
        end
    end
endmodule
```

“Result”:-

```
addr = 16
addr = 36
addr = 32
addr = 11
addr = 19
Simulation has finished. There are no more test vectors to simulate.
```