DAY-76
#100DAYSRTL

"System Verilog:-Constraints(Phase-1)"

"<u>Introduction</u>":-

Constraints are a pivotal facet of randomization. Constraints serve as the avenue through which we articulate specific conditions or limitations on the randomness imbued in our values. Given the expansive nature of this subject, we've opted for a phased approach, unraveling the intricacies of constraints across multiple articles.

"<u>Constraints</u>":-

During randomization, it might require to randomize the variable within a range of values or with an inset of values or other than a range of values. this can be achieved by using constraints inside the operator. With the System Verilog inside operator, random variables will get values specified within the inside block.

- values within the inside block can be variable, constant, or range
- the inside block is written with an inside keyword followed by curly braces {}

"<u>Constraint Blocks</u>":-

- Constraint blocks are class members like tasks, functions, and variables
- Constraint blocks will have a unique name within a class
- Constraint blocks consist of conditions or expressions to limit or control the values for a random variable
- Constraint blocks are enclosed within curly braces { }.

- Constraint blocks can be defined inside the class or outside the class like extern methods, a constraint block defined outside the class is called an extern constraint block

“Code Practising”:-

```
class packet;
  rand bit [3:0] addr;
  constraint addr_range {addr > 5; }
endclass
module constr_blocks;
initial begin
 packet pkt;
 pkt = new();
  repeat(10) begin
   pkt.randomize();
    $display("\taddr = %0d",pkt.addr);
  end
 end
endmodule
```

“Result”:-

```
      addr = 8
      addr = 11
      addr = 11
      addr = 9
      addr = 12
      addr = 14
      addr = 15
      addr = 9
      addr = 6
      addr = 13
Simulation has finished. There are no more test vectors to simulate.
```

“Code Practising”:-

```
class packet;
  rand bit [3:0] addr;
  constraint addr_range ;
endclass
constraint packet::addr_range {addr>5;}; //constraint block outside the class
module constr_blocks;
initial begin
 packet pkt;
 pkt = new();
  repeat(10) begin
   pkt.randomize();
    $display("\taddr = %0d",pkt.addr);
  end
 end
endmodule
```

“Result”:-

```
      addr = 8
      addr = 11
      addr = 11
      addr = 9
      addr = 12
      addr = 14
      addr = 15
      addr = 9
      addr = 6
      addr = 13
Simulation has finished. There are no more test vectors to simulate.
```

“Constraint Inheritance”:-

- Like class members, constraints also will be inherited from parent class to child class.
- Constraint blocks can be overridden by writing constraint blocks with the same name as in the parent class

“Code Practising”:-

```
class packet1;
  rand bit [3:0] addr;
  constraint addr_range {addr > 5; }
endclass
class packet2 extends packet1;
  constraint addr_range {addr < 5; }//overriding constraint of parent class
endclass
module const_inhe;
initial begin
 packet1 pkt1;
 packet2 pkt2;
 pkt1 =new();
 pkt2 =new();
repeat(5) begin
pkt1.randomize();
  $display("\tpkt1:: addr = %0d",pkt1.addr);
end
  $display("----------------------------------------");
repeat(5) begin
pkt2.randomize();
  $display("\tpkt2:: addr = %0d",pkt2.addr);
end
end
endmodule
```

“Result”:-

```
        pkt1:: addr = 7
        pkt1:: addr = 14
        pkt1:: addr = 6
        pkt1:: addr = 8
        pkt1:: addr = 12
----------------------------------------
        pkt2:: addr = 2
        pkt2:: addr = 0
        pkt2:: addr = 1
        pkt2:: addr = 3
        pkt2:: addr = 2
Simulation has finished. There are no more test vectors to simulate.
```

“Constraints inside”:-

- values within the inside block can be variable, constant or range
- the inside block is written with an inside keyword followed by curly braces {}

```
constraint addr_range { addr inside { ... }; }
```

- the range is specified by [ ]

```
constraint addr_range { addr inside { [5:10]}; }
```

- set of values are specified by 'comma',

```
constraint addr_range { addr inside { 1,3,5,7,9}; }
```

- it is allowed to mix range and set of values

```
constraint addr_range { addr inside {1,3,[5:10],12,[13:15]};}
```

"Code Practising":-

```
class packet;
  rand bit [3:0] addr;
  rand bit [3:0] start_addr;
  rand bit [3:0] end_addr;
  constraint addr_1_range { addr inside {[start_addr:end_addr]}; }
endclass
module constr_inside;
  initial begin
    packet pkt;
    pkt = new();
    $display("************************************************");
    repeat(3) begin
      pkt.randomize();
      $display("\tstart_addr = %0d,end_addr = %0d",pkt.start_addr,pkt.end_addr);
      $display("\taddr = %0d",pkt.addr);
      $display("************************************************");
    end
  end
endmodule
```

"Result":-

```
************************************************
      start_addr = 2,end_addr = 11
      addr = 4
************************************************
      start_addr = 1,end_addr = 12
      addr = 6
************************************************
      start_addr = 3,end_addr = 11
      addr = 4
************************************************
Simulation has finished. There are no more test vectors to simulate.
```