# Randomization

As designs grow larger, it becomes more difficult to create a complete set of stimulus needed to check their functionality. You can write a directed test-case to check a certain set of features, but you cannot write enough directed test-cases when the number of features keeps doubling on each project.

The solution is to create test cases automatically using constrained-random tests (CRT). A directed test finds the bugs you think are there, but a CRT finds bugs you never thought about, by using random stimulus. You restrict the test scenarios to those that are both valid and of interest by using constraints.

This is good for randomizing the variables alone, but it is hard to use in case of class object randomization. for easy randomization of class properties, System Verilog provides rand keyword and randomize() method.

## rand and randc keywords

To randomize a class object, the following keywords are used while declaring class variables.

**rand:** On randomizing an object, the rand keyword provides uniformly distributed random values.

```
Rand bit [4:0] variable;
```

**randc:**

**randc is random-cyclic. For the variables declared with the randc keyword, on randomization variable values don't repeat a random value until every possible value has been assigned.**

```
Randc bit [1:0]value;
```

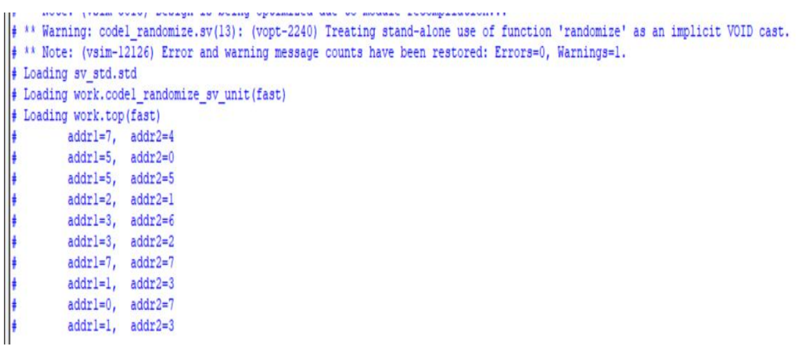
## CODE-1 Simple Class with Random Variables

Two variables addr1 and addr2 of same bit type are declared as rand and randc respectively, observe the randomized values of addr1 and addr2.

```
class sample_pkt;
rand bit [2:0]addr1;
randc bit [2:0]addr2;
endclass

module top;
Sample_pkt pkt;
initial begin
            pkt = new();
            repeat(10) begin
            pkt.randomize();
            $display("\t addr1=%0d,\taddr2=%0d",pkt.addr1,pkt.addr2);
   end
end
endmodule
```

## Simulator Output:



```
#    Note: (vsim-[illegible]) Design is being optimized due to module recompilation...
# ** Warning: codel_randomize.sv(13): (vopt-2240) Treating stand-alone use of function 'randomize' as an implicit VOID cast.
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=1.
# Loading sv_std.std
# Loading work.codel_randomize_sv_unit(fast)
# Loading work.top(fast)
#       addr1=7,  addr2=4
#       addr1=5,  addr2=0
#       addr1=5,  addr2=5
#       addr1=2,  addr2=1
#       addr1=3,  addr2=6
#       addr1=3,  addr2=2
#       addr1=7,  addr2=7
#       addr1=1,  addr2=3
#       addr1=0,  addr2=7
#       addr1=1,  addr2=3
```

## CODE-2:

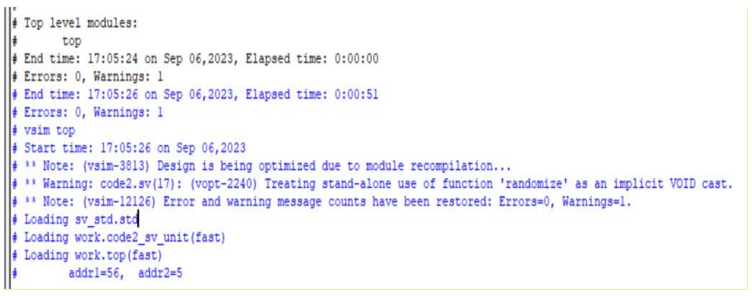
a packet class with random variables and constraints. plus testbench code that constructs and randomizes a packet.

```
class sample_pkt;
rand bit[7:0]addr1;
randc bit[2:0]addr2;

//limit the value of addr1 and addr2
constraint addr_c{
                    addr1>15;
                     addr2>4;
                                     }
endclass

module top;
sample_pkt s;
initial begin
                         s = new();
                         s.randomize();
                         $display("\t addr1=%0d,\t
addr2=%0d",s.addr1,s.addr2);

            end
endmodule
```



```
# Top level modules:
#       top
# End time: 17:05:24 on Sep 06,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 1
# End time: 17:05:26 on Sep 06,2023, Elapsed time: 0:00:51
# Errors: 0, Warnings: 1
# vsim top
# Start time: 17:05:26 on Sep 06,2023
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# ** Warning: code2.sv(17): (vopt-2240) Treating stand-alone use of function 'randomize' as an implicit VOID cast.
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=1.
# Loading sv_std.std
# Loading work.code2_sv_unit(fast)
# Loading work.top(fast)
#        addr1=56,  addr2=5
```

## Systemverilog Disable Randomization:

The rand_mode() method is used to disable the randomization of a variable declared with the rand/randc keyword.

- rand_mode(1) means randomization enabled
- rand_mode(0) means randomization disabled

**Syntax:**

**Object_name.constraint_name.Constraint_mode(0)**

## CODE-3: randomization enable and disable examples:

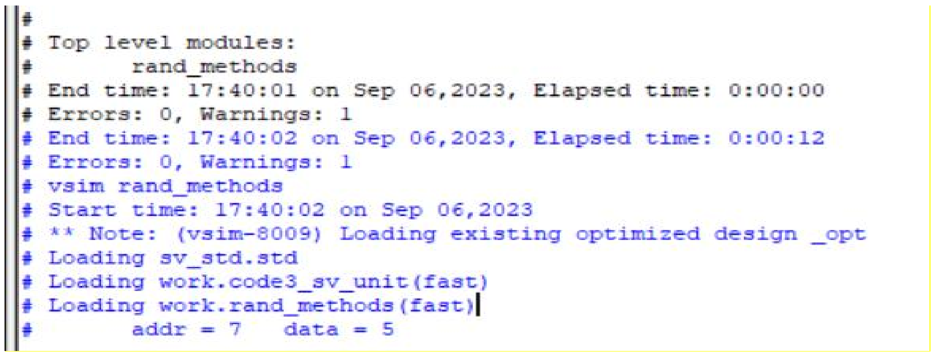
The class packet has random variables addr and data, on randomization, these variables will get random value.

### without randomization disable:

```
class sample_pkt;
  rand bit [2:0]addr;
  randc bit[2:0] data;
endclass

module rand_methods;
  initial begin
    sample_pkt pkt;
     pkt = new();

    //calling randomize method
      pkt.randomize();
    $display("\taddr = %0d \t data = %0d",pkt.addr,pkt.data);
  end
endmodule
```



```
#
# Top level modules:
#       rand_methods
# End time: 17:40:01 on Sep 06,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 1
# End time: 17:40:02 on Sep 06,2023, Elapsed time: 0:00:12
# Errors: 0, Warnings: 1
# vsim rand_methods
# Start time: 17:40:02 on Sep 06,2023
# ** Note: (vsim-8009) Loading existing optimized design _opt
# Loading sv_std.std
# Loading work.code3_sv_unit(fast)
# Loading work.rand_methods(fast)
#       addr = 7   data = 5
```

## randomization disable for a class

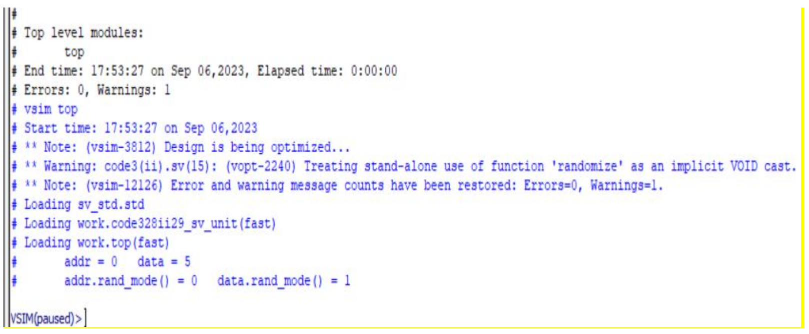
The class packet has random variables addr and data, randomization is disabled for a variable addr, on randomization only data will get random value. The addr will not get any random value.

```
class sample_pkt;
  rand bit [2:0]addr;
  rand bit [2:0] data;
endclass

module top;
  initial begin
    sample_pkt pkt;
    pkt = new();

    //disable rand_mode of addr variable of pkt
            pkt.addr.rand_mode(0);

    //calling randomize method
    pkt.randomize();
    $display("\taddr = %0d \t data = %0d",pkt.addr,pkt.data);
    $display("\taddr.rand_mode() = %0d \t data.rand_mode()
= %0d",pkt.addr.rand_mode(),pkt.data.rand_mode());
  end
endmodule
```



```
#
# Top level modules:
#       top
# End time: 17:53:27 on Sep 06,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 1
# vsim top
# Start time: 17:53:27 on Sep 06,2023
# ** Note: (vsim-3812) Design is being optimized...
# ** Warning: code3(ii).sv(15): (vopt-2240) Treating stand-alone use of function 'randomize' as an implicit VOID cast.
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=1.
# Loading sv_std.std
# Loading work.code328ii29_sv_unit(fast)
# Loading work.top(fast)
#       addr = 0   data = 5
#       addr.rand_mode() = 0   data.rand_mode() = 1

VSIM(paused)>
```

## Randomization Methods

System Verilog randomization provides a built-in methods : methods that comes implemented with the language.

randomize method comes with 2 callback methods:

- Pre_randomize
- Post_randomize

What is callback method?
When user calls pkt.randomize();

### Pre_randomize():

the pre_randomize function can be used to set pre-conditions before the object randomization.

randomization enable or disable by using rand_mode() method.
User can be override this method.

### Post_randomize():

the post_randomization function can be used to check and perform post-conditions after the object randomization.

User can be override this method.

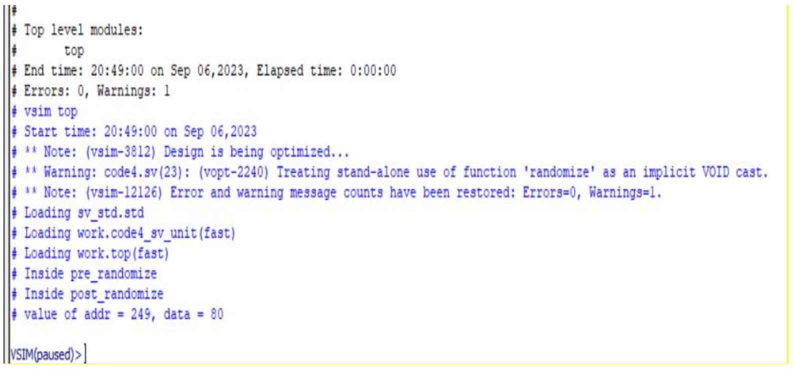
## CODE-4

Implementing pre randomize and post randomize methods in the class.

```
class sample_pkt;
  rand  bit [7:0] addr;
  randc bit [7:0] data;

  //post randomization method
  function void pre_randomize();
    $display("Inside pre_randomize");
  endfunction
  //post randomization method
  function void post_randomize();
    $display("Inside post_randomize");
    $display("value of addr = %0d, data = %0d",addr,data);
  endfunction
endclass

module top;
  initial begin
    sample_pkt pkt;
    pkt = new();
    pkt.randomize();
  end
endmodule
```



```
#
# Top level modules:
#       top
# End time: 20:49:00 on Sep 06,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 1
# vsim top
# Start time: 20:49:00 on Sep 06,2023
# ** Note: (vsim-3812) Design is being optimized...
# ** Warning: code4.sv(23): (vopt-2240) Treating stand-alone use of function 'randomize' as an implicit VOID cast.
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=1.
# Loading sv_std.std
# Loading work.code4_sv_unit(fast)
# Loading work.top(fast)
# Inside pre_randomize
# Inside post_randomize
# value of addr = 249, data = 80

VSIM(paused)>
```