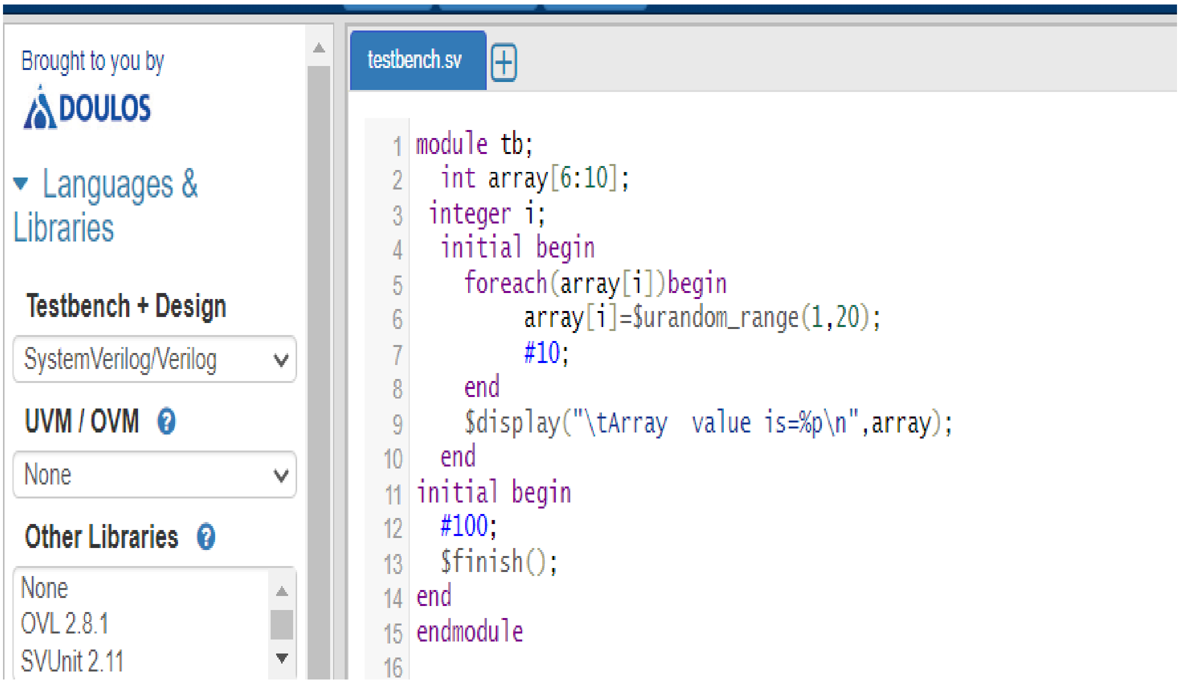
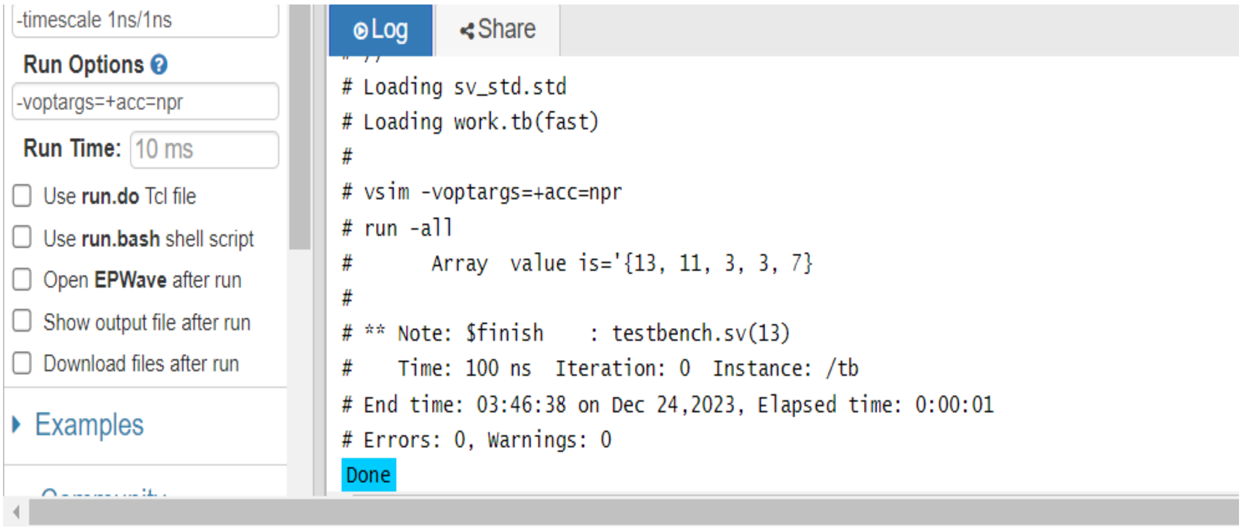
//You have an array, int array[6:10] Randomize array between 1 to 20 without using randc.



Testbench + Design: SystemVerilog/Verilog

UVM / OVM: None

```
module tb;
  int array[6:10];
 integer i;
  initial begin
    foreach(array[i])begin
         array[i]=$urandom_range(1,20);
         #10;
    end
    $display("\tArray  value is=%p\n",array);
  end
initial begin
  #100;
  $finish();
end
endmodule
```



-timescale 1ns/1ns

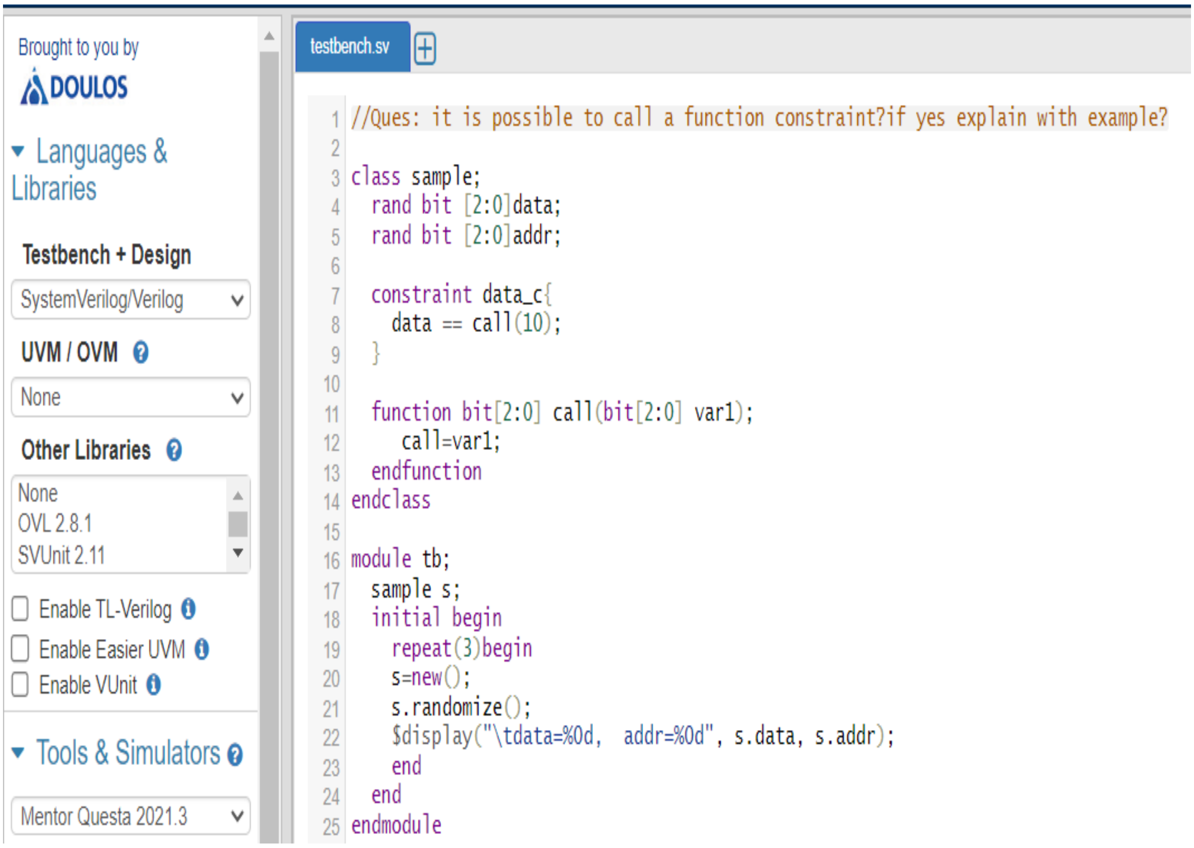
Run Options: -voptargs=+acc=npr

Run Time: 10 ms

☐ Use run.do Tcl file
☐ Use run.bash shell script
☐ Open EPWave after run
☐ Show output file after run
☐ Download files after run

```
# Loading sv_std.std
# Loading work.tb(fast)
#
# vsim -voptargs=+acc=npr
# run -all
#       Array  value is='{13, 11, 3, 3, 7}
#
# ** Note: $finish    : testbench.sv(13)
#    Time: 100 ns  Iteration: 0  Instance: /tb
# End time: 03:46:38 on Dec 24,2023, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
Done
```

//Ques:3 it is possible to call a function constraint?if yes explain with example?



Testbench + Design: SystemVerilog/Verilog

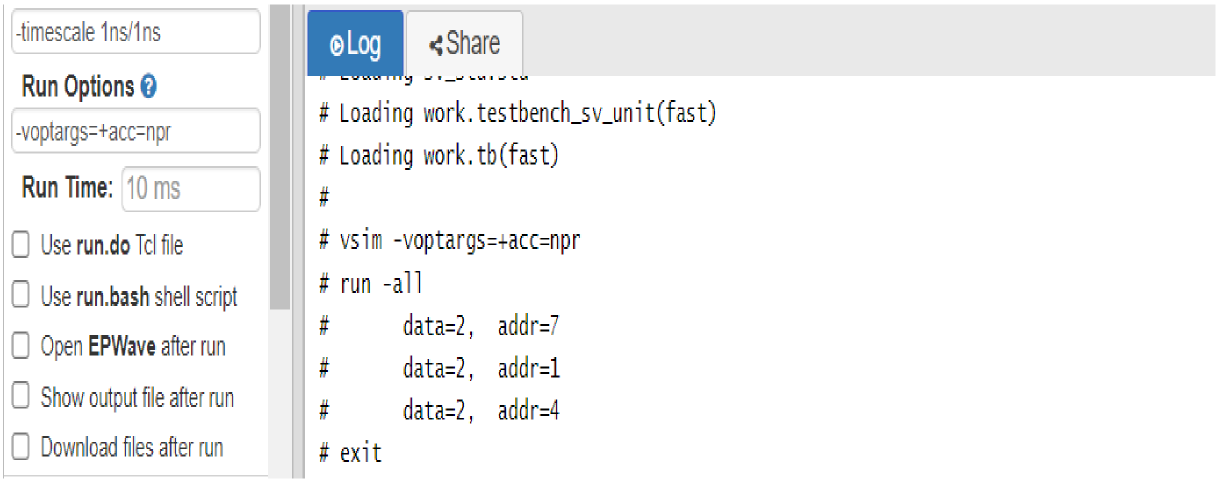
UVM / OVM: None

Other Libraries: None, OVL 2.8.1, SVUnit 2.11

□ Enable TL-Verilog
□ Enable Easier UVM
□ Enable VUnit

Tools & Simulators: Mentor Questa 2021.3

```
//Ques: it is possible to call a function constraint?if yes explain with example?

class sample;
  rand bit [2:0]data;
  rand bit [2:0]addr;

  constraint data_c{
    data == call(10);
  }

  function bit[2:0] call(bit[2:0] var1);
     call=var1;
  endfunction
endclass

module tb;
  sample s;
  initial begin
    repeat(3)begin
    s=new();
    s.randomize();
    $display("\tdata=%0d,  addr=%0d", s.data, s.addr);
    end
  end
endmodule
```



-timescale 1ns/1ns

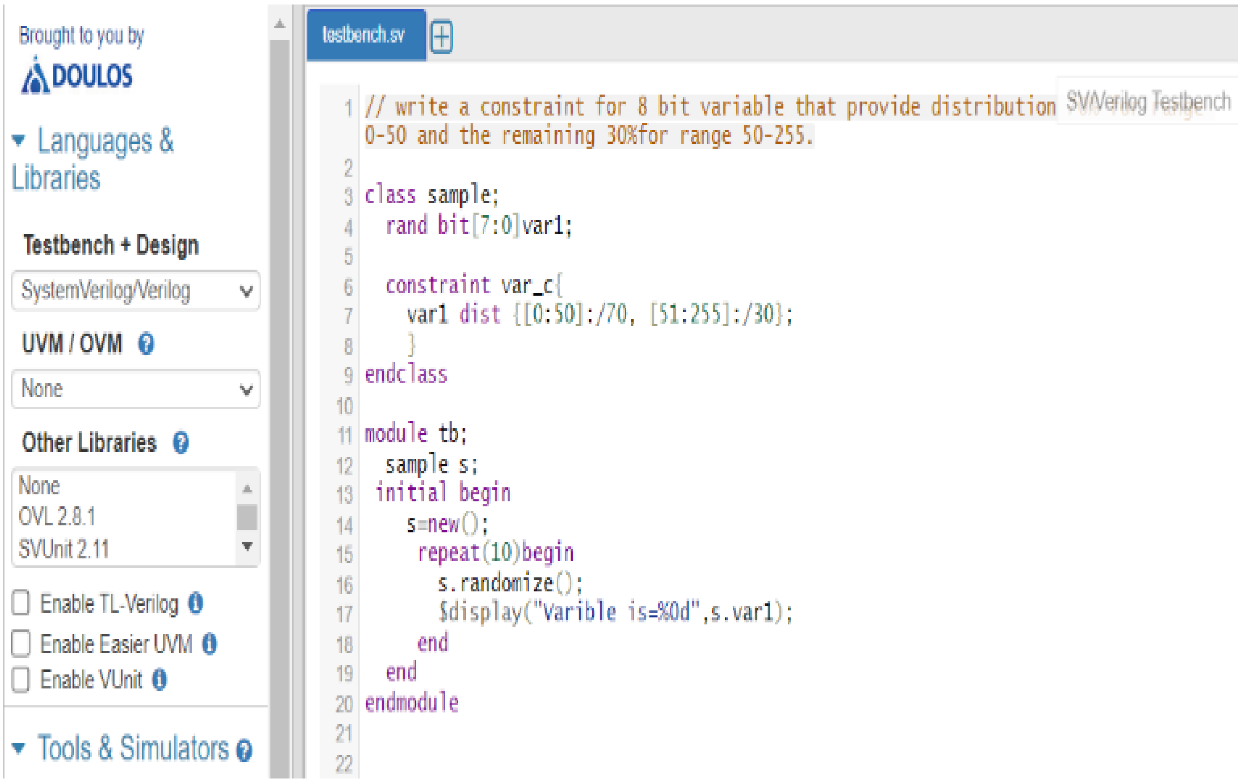
Run Options: -voptargs=+acc=npr

Run Time: 10 ms

□ Use run.do Tcl file
□ Use run.bash shell script
□ Open EPWave after run
□ Show output file after run
□ Download files after run

```
# Loading work.testbench_sv_unit(fast)
# Loading work.tb(fast)
#
# vsim -voptargs=+acc=npr
# run -all
#       data=2,  addr=7
#       data=2,  addr=1
#       data=2,  addr=4
# exit
```

//Ques:4 write a constraint for 8-bit variable that provide distribution 70% for range 0-50 and the remaining 30%for range 50-255.



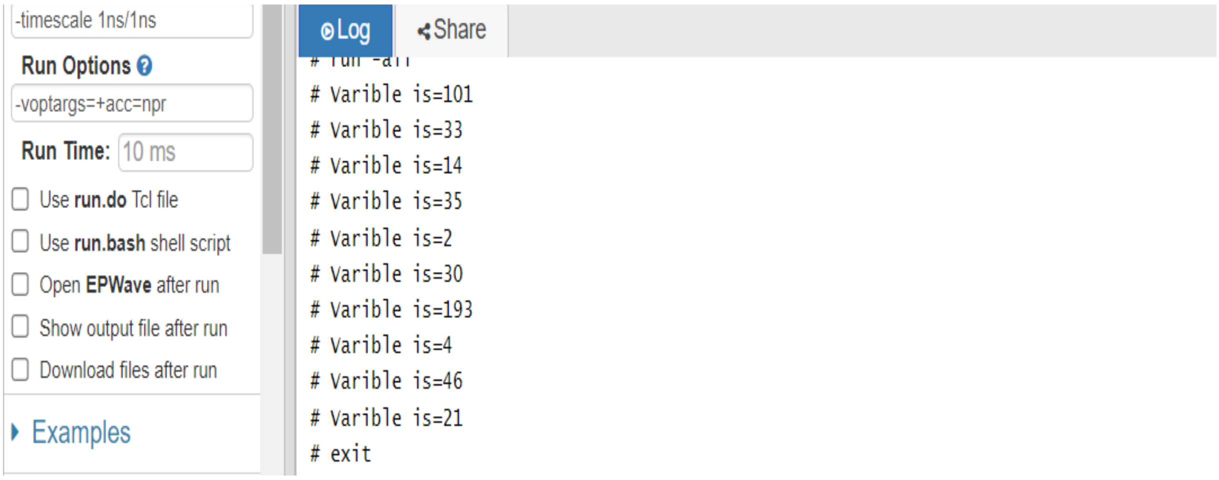
Testbench + Design
SystemVerilog/Verilog

UVM / OVM
None

Other Libraries
None
OVL 2.8.1
SVUnit 2.11

□ Enable TL-Verilog
□ Enable Easier UVM
□ Enable VUnit

```
// write a constraint for 8 bit variable that provide distribution
0-50 and the remaining 30%for range 50-255.

class sample;
  rand bit[7:0]var1;

  constraint var_c{
    var1 dist {[0:50]:/70, [51:255]:/30};
    }
endclass

module tb;
  sample s;
 initial begin
    s=new();
     repeat(10)begin
       s.randomize();
       $display("Varible is=%0d",s.var1);
     end
   end
endmodule
```



-timescale 1ns/1ns

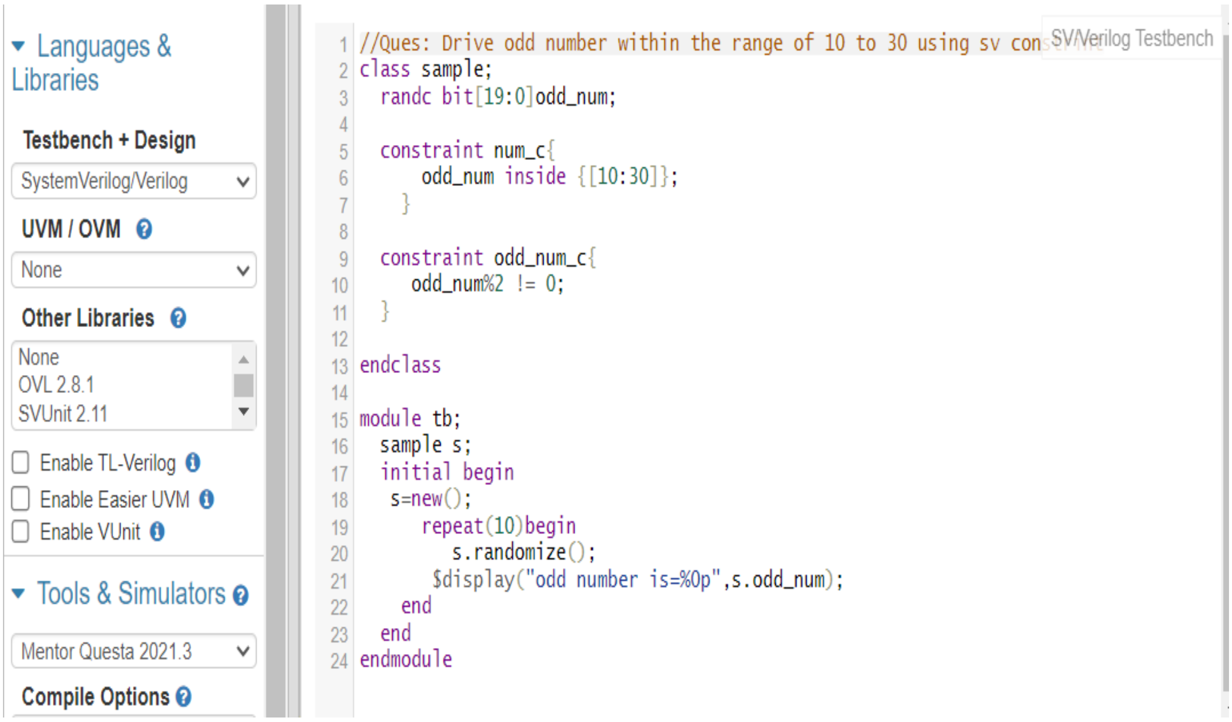
Run Options
-voptargs=+acc=npr

Run Time: 10 ms

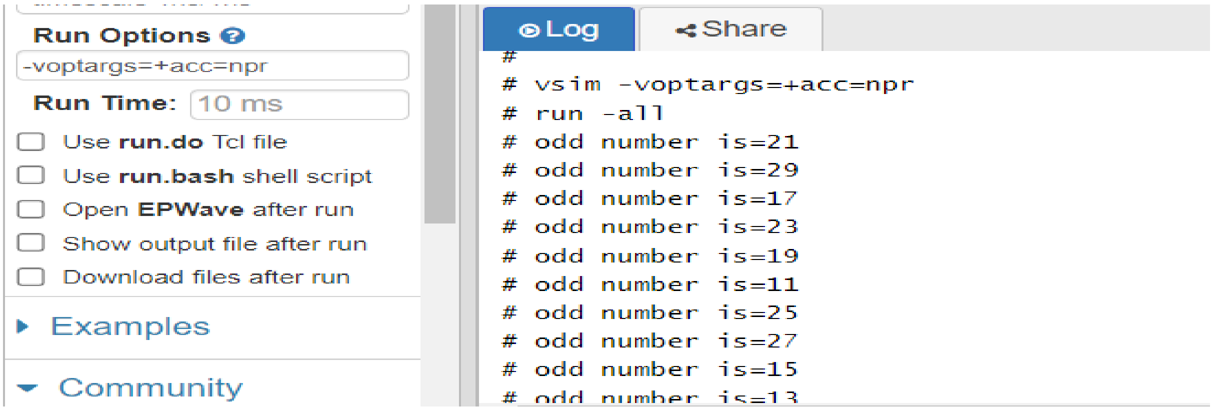
□ Use run.do Tcl file
□ Use run.bash shell script
□ Open EPWave after run
□ Show output file after run
□ Download files after run

```
# run -all
# Varible is=101
# Varible is=33
# Varible is=14
# Varible is=35
# Varible is=2
# Varible is=30
# Varible is=193
# Varible is=4
# Varible is=46
# Varible is=21
# exit
```

//Ques:5 Drive odd number within the range of 10 to 30 using sv constraint.

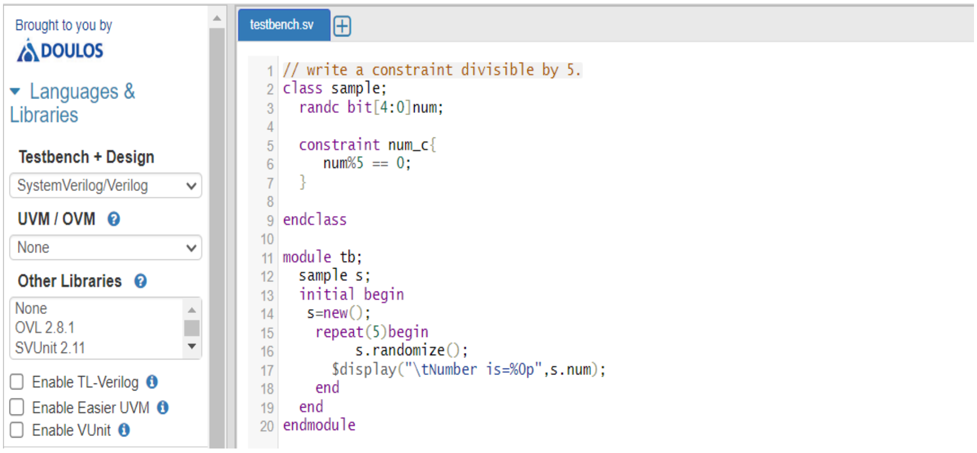


```
//Ques: Drive odd number within the range of 10 to 30 using sv con
class sample;
  randc bit[19:0]odd_num;

  constraint num_c{
      odd_num inside {[10:30]};
    }

  constraint odd_num_c{
     odd_num%2 != 0;
  }

endclass

module tb;
  sample s;
  initial begin
   s=new();
      repeat(10)begin
         s.randomize();
       $display("odd number is=%0p",s.odd_num);
    end
  end
endmodule
```



```
#
# vsim -voptargs=+acc=npr
# run -all
# odd number is=21
# odd number is=29
# odd number is=17
# odd number is=23
# odd number is=19
# odd number is=11
# odd number is=25
# odd number is=27
# odd number is=15
# odd number is=13
```

//Ques:6 write a constraint divisible by 5.



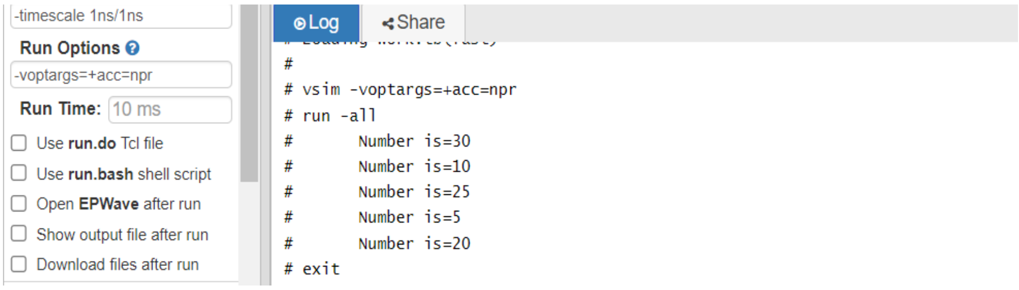
Testbench + Design: SystemVerilog/Verilog

UVM / OVM: None

Other Libraries: None, OVL 2.8.1, SVUnit 2.11

□ Enable TL-Verilog
□ Enable Easier UVM
□ Enable VUnit

```
// write a constraint divisible by 5.
class sample;
  randc bit[4:0]num;

  constraint num_c{
     num%5 == 0;
  }

endclass

module tb;
  sample s;
  initial begin
   s=new();
    repeat(5)begin
         s.randomize();
      $display("\tNumber is=%0p",s.num);
    end
  end
endmodule
```



-timescale 1ns/1ns

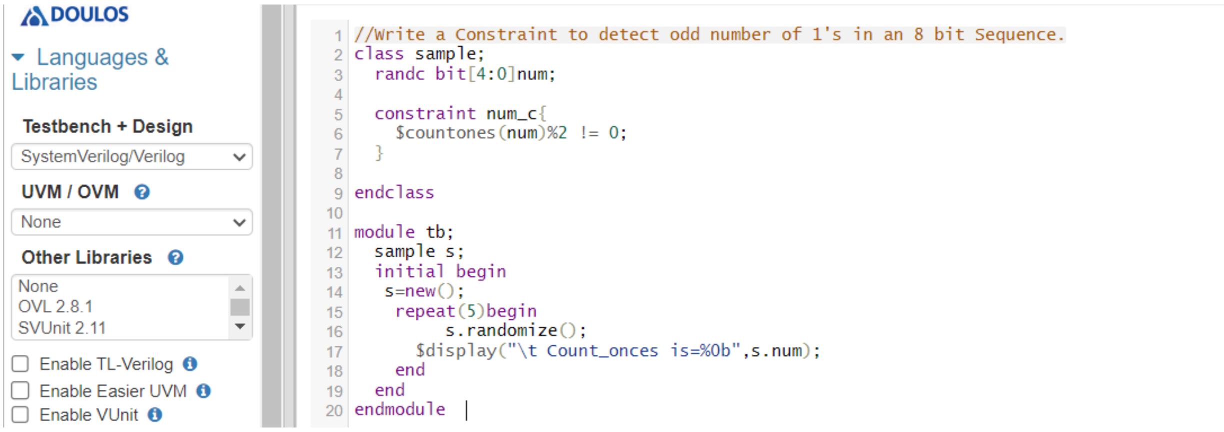
Run Options: -voptargs=+acc=npr

Run Time: 10 ms

□ Use run.do Tcl file
□ Use run.bash shell script
□ Open EPWave after run
□ Show output file after run
□ Download files after run

```
#
# vsim -voptargs=+acc=npr
# run -all
#       Number is=30
#       Number is=10
#       Number is=25
#       Number is=5
#       Number is=20
# exit
```

//Ques:7 Write a Constraint to detect odd number of 1's in an 8 bit Sequence.



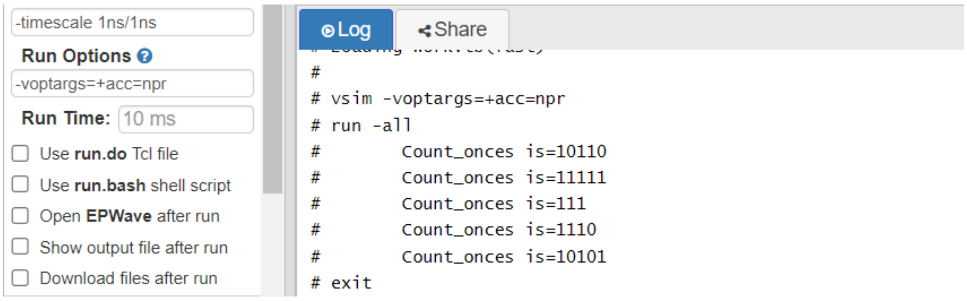
Testbench + Design: SystemVerilog/Verilog

UVM / OVM: None

Other Libraries: None, OVL 2.8.1, SVUnit 2.11

□ Enable TL-Verilog
□ Enable Easier UVM
□ Enable VUnit

```
//Write a Constraint to detect odd number of 1's in an 8 bit Sequence.
class sample;
  randc bit[4:0]num;

  constraint num_c{
    $countones(num)%2 != 0;
  }

endclass

module tb;
  sample s;
  initial begin
   s=new();
    repeat(5)begin
         s.randomize();
      $display("\t Count_onces is=%0b",s.num);
    end
  end
endmodule
```



-timescale 1ns/1ns

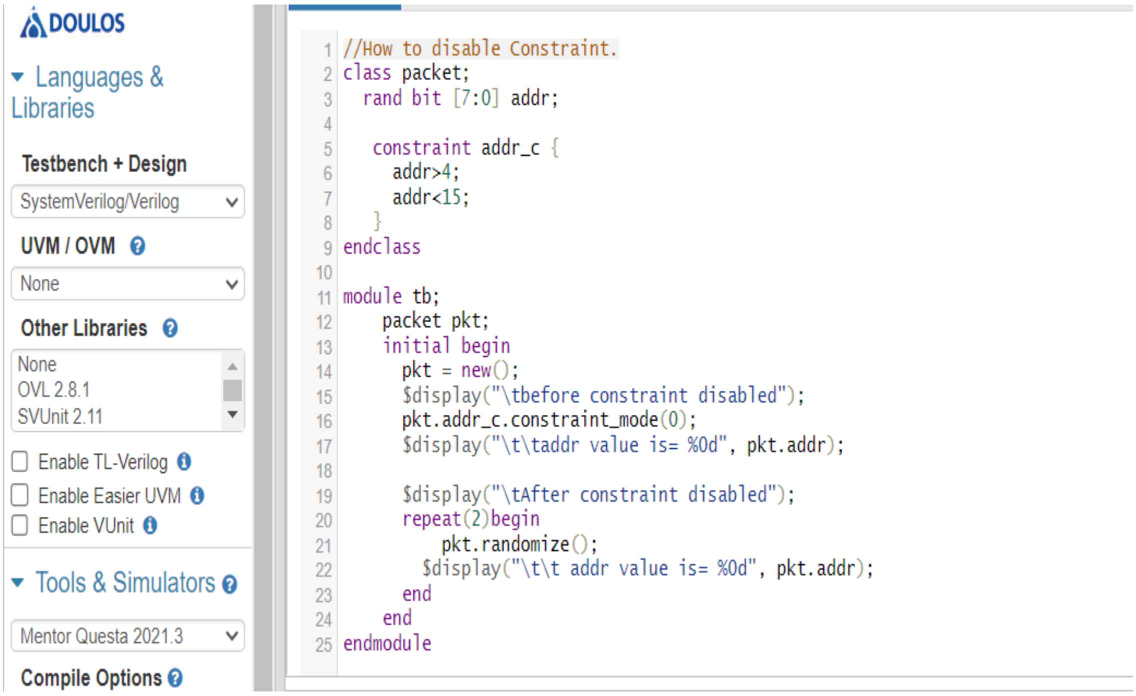
Run Options: -voptargs=+acc=npr

Run Time: 10 ms

□ Use run.do Tcl file
□ Use run.bash shell script
□ Open EPWave after run
□ Show output file after run
□ Download files after run

```
#
# vsim -voptargs=+acc=npr
# run -all
#       Count_onces is=10110
#       Count_onces is=11111
#       Count_onces is=111
#       Count_onces is=1110
#       Count_onces is=10101
# exit
```

//Ques:8 How to disable Constraint.



Testbench + Design: SystemVerilog/Verilog

UVM / OVM: None

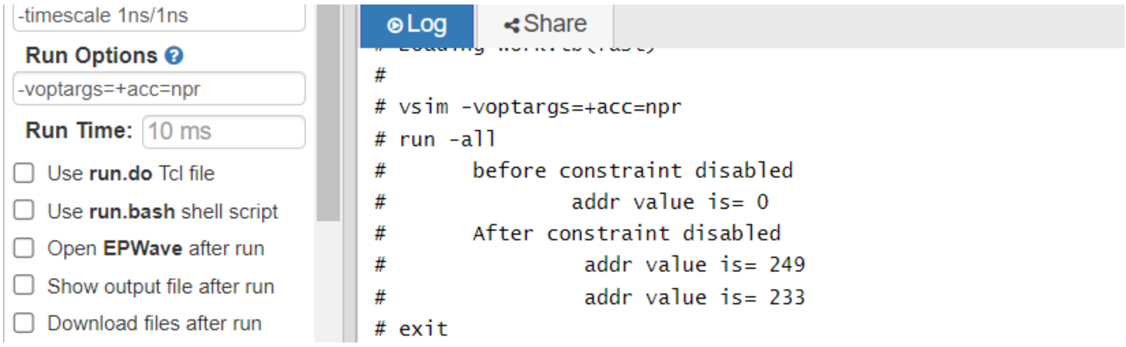
Other Libraries: None / OVL 2.8.1 / SVUnit 2.11

☐ Enable TL-Verilog
☐ Enable Easier UVM
☐ Enable VUnit

Mentor Questa 2021.3

Compile Options

```
//How to disable Constraint.
class packet;
  rand bit [7:0] addr;

   constraint addr_c {
     addr>4;
     addr<15;
   }
endclass

module tb;
    packet pkt;
    initial begin
      pkt = new();
      $display("\tbefore constraint disabled");
      pkt.addr_c.constraint_mode(0);
      $display("\t\taddr value is= %0d", pkt.addr);

      $display("\tAfter constraint disabled");
      repeat(2)begin
          pkt.randomize();
        $display("\t\t addr value is= %0d", pkt.addr);
      end
    end
endmodule
```



-timescale 1ns/1ns

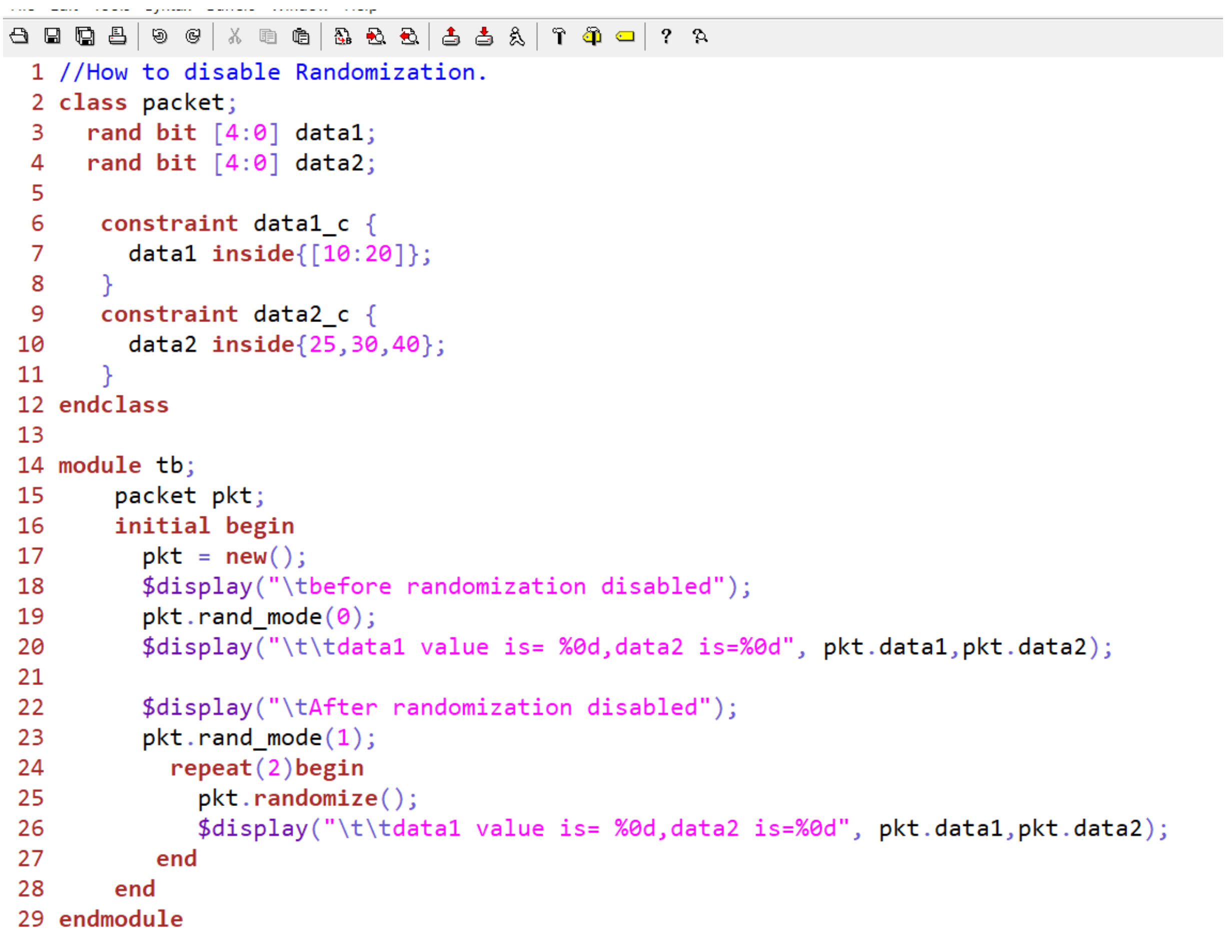
Run Options: -voptargs=+acc=npr

Run Time: 10 ms

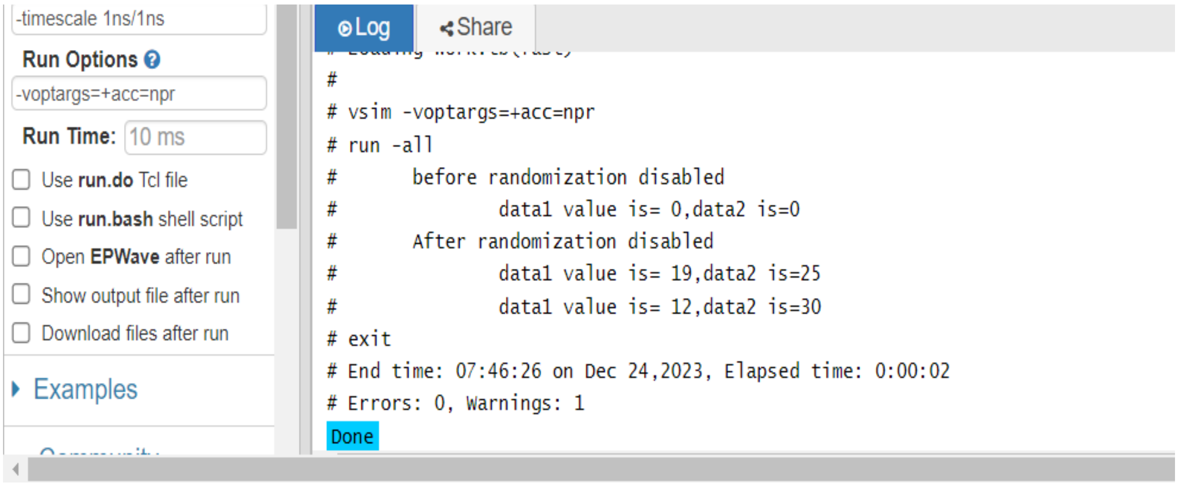
☐ Use run.do Tcl file
☐ Use run.bash shell script
☐ Open EPWave after run
☐ Show output file after run
☐ Download files after run

```
# Loading work.tb(fast)
#
# vsim -voptargs=+acc=npr
# run -all
#       before constraint disabled
#               addr value is= 0
#       After constraint disabled
#                addr value is= 249
#                addr value is= 233
# exit
```

//Ques:9 How to disable Randomization.



```
//How to disable Randomization.
class packet;
  rand bit [4:0] data1;
  rand bit [4:0] data2;

   constraint data1_c {
     data1 inside{[10:20]};
   }
   constraint data2_c {
     data2 inside{25,30,40};
   }
endclass

module tb;
    packet pkt;
    initial begin
      pkt = new();
      $display("\tbefore randomization disabled");
      pkt.rand_mode(0);
      $display("\t\tdata1 value is= %0d,data2 is=%0d", pkt.data1,pkt.data2);

      $display("\tAfter randomization disabled");
      pkt.rand_mode(1);
        repeat(2)begin
          pkt.randomize();
          $display("\t\tdata1 value is= %0d,data2 is=%0d", pkt.data1,pkt.data2);
        end
    end
endmodule
```



-timescale 1ns/1ns

Run Options

-voptargs=+acc=npr

Run Time: 10 ms

☐ Use **run.do** Tcl file

☐ Use **run.bash** shell script

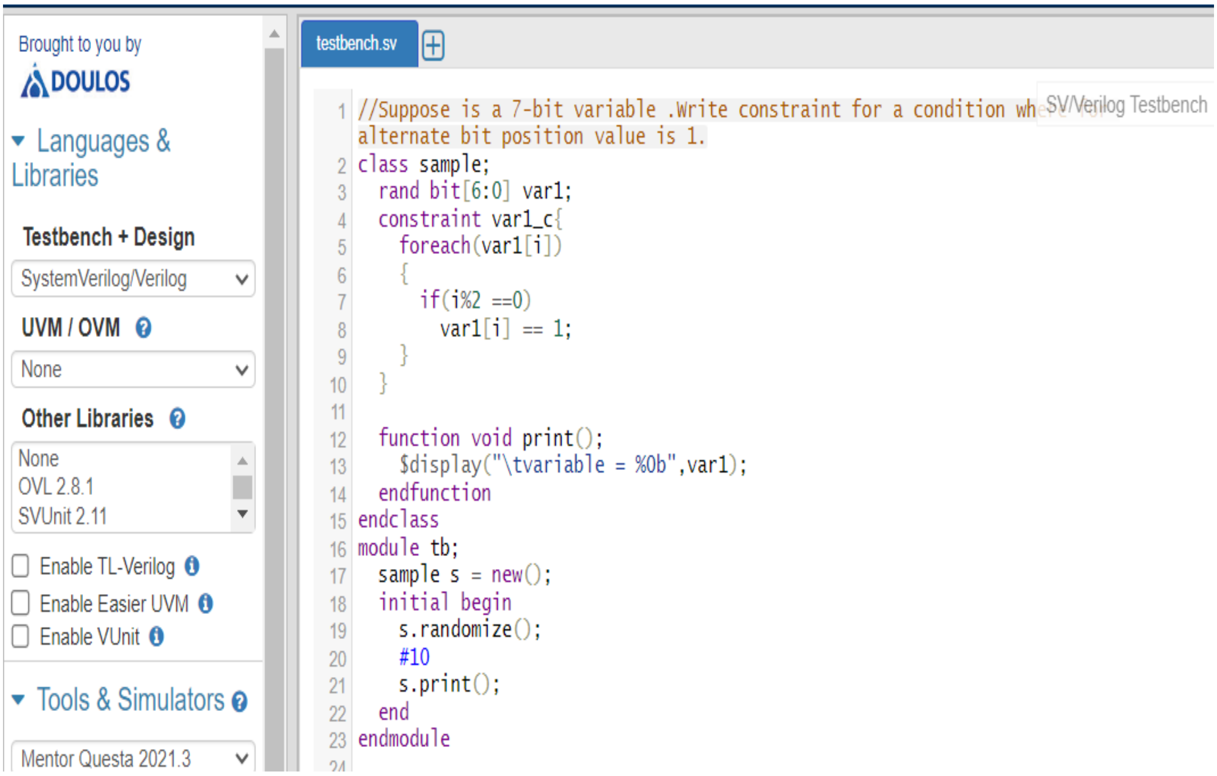
☐ Open **EPWave** after run

☐ Show output file after run

☐ Download files after run

```
#
# vsim -voptargs=+acc=npr
# run -all
#       before randomization disabled
#               data1 value is= 0,data2 is=0
#       After randomization disabled
#               data1 value is= 19,data2 is=25
#               data1 value is= 12,data2 is=30
# exit
# End time: 07:46:26 on Dec 24,2023, Elapsed time: 0:00:02
# Errors: 0, Warnings: 1
Done
```

//Ques:14 Suppose is a 7-bit variable. Write constraint for a condition where for alternate bit position value is 1.



Testbench + Design: SystemVerilog/Verilog

UVM / OVM: None

Other Libraries: None, OVL 2.8.1, SVUnit 2.11

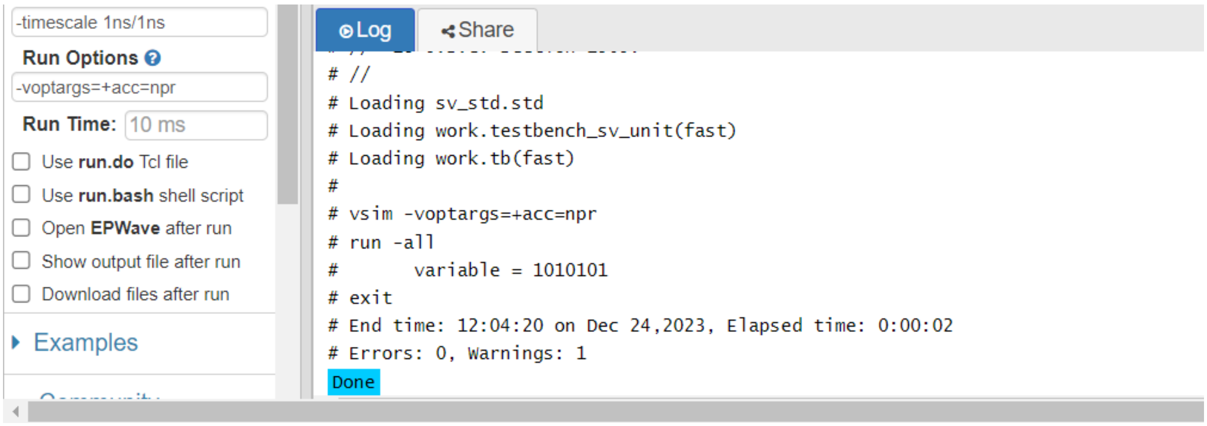
□ Enable TL-Verilog

□ Enable Easier UVM

□ Enable VUnit

Mentor Questa 2021.3

```
//Suppose is a 7-bit variable .Write constraint for a condition wh
alternate bit position value is 1.
class sample;
  rand bit[6:0] var1;
  constraint var1_c{
    foreach(var1[i])
    {
      if(i%2 ==0)
        var1[i] == 1;
    }
  }

  function void print();
    $display("\tvariable = %0b",var1);
  endfunction
endclass
module tb;
  sample s = new();
  initial begin
    s.randomize();
    #10
    s.print();
  end
endmodule
```



-timescale 1ns/1ns

Run Options: -voptargs=+acc=npr

Run Time: 10 ms

□ Use run.do Tcl file

□ Use run.bash shell script

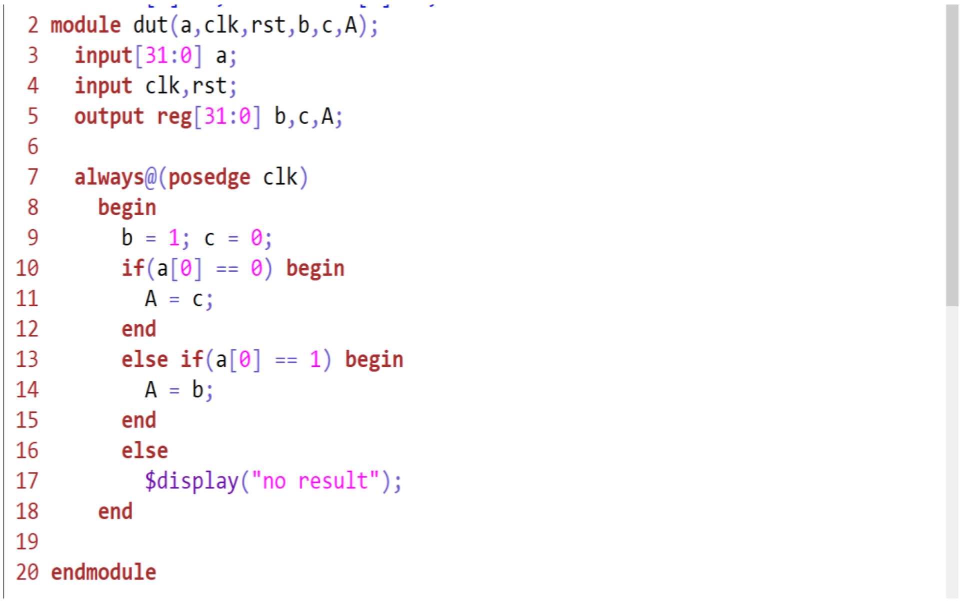
□ Open EPWave after run

□ Show output file after run

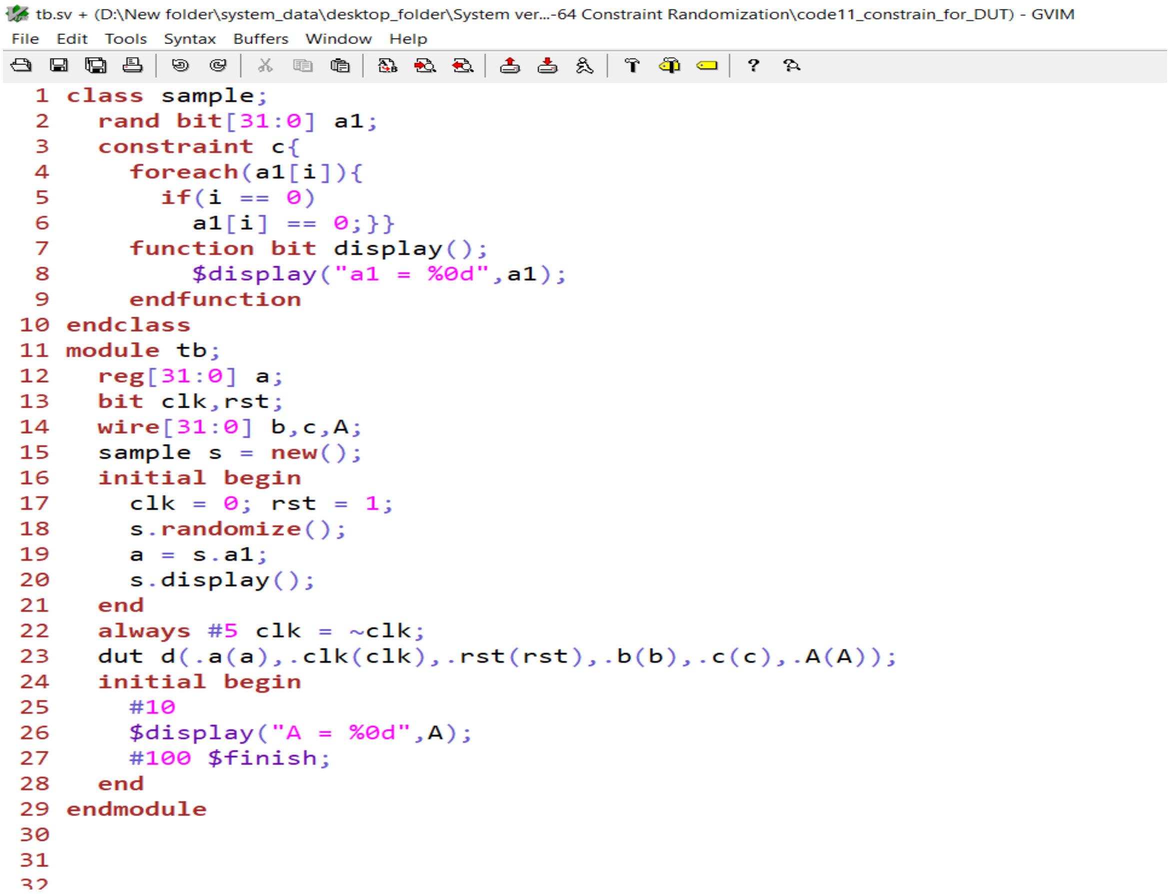
□ Download files after run

```
# //
# Loading sv_std.std
# Loading work.testbench_sv_unit(fast)
# Loading work.tb(fast)
#
# vsim -voptargs=+acc=npr
# run -all
#        variable = 1010101
# exit
# End time: 12:04:20 on Dec 24,2023, Elapsed time: 0:00:02
# Errors: 0, Warnings: 1
Done
```

//Ques:15 a is 32-bit input to Dut with clk and rst Output are 32 bits b and c. when a[0]==1,a=b. when a[0]==0,a=c.write the constraints for this Dut.



```
module dut(a,clk,rst,b,c,A);
  input[31:0] a;
  input clk,rst;
  output reg[31:0] b,c,A;

  always@(posedge clk)
    begin
      b = 1; c = 0;
      if(a[0] == 0) begin
        A = c;
      end
      else if(a[0] == 1) begin
        A = b;
      end
      else
        $display("no result");
    end

endmodule
```



tb.sv + (D:\New folder\system_data\desktop_folder\System ver...-64 Constraint Randomization\code11_constrain_for_DUT) - GVIM
File Edit Tools Syntax Buffers Window Help

```
class sample;
  rand bit[31:0] a1;
  constraint c{
    foreach(a1[i]){
      if(i == 0)
        a1[i] == 0;}}
    function bit display();
        $display("a1 = %0d",a1);
    endfunction
endclass
module tb;
  reg[31:0] a;
  bit clk,rst;
  wire[31:0] b,c,A;
  sample s = new();
  initial begin
    clk = 0; rst = 1;
    s.randomize();
    a = s.a1;
    s.display();
  end
  always #5 clk = ~clk;
  dut d(.a(a),.clk(clk),.rst(rst),.b(b),.c(c),.A(A));
  initial begin
    #10
    $display("A = %0d",A);
    #100 $finish;
  end
endmodule
```