SYSTEM VERILOG

CONSTRAINTS – Part 5

Write a System Verilog constraint to generate a number such that the number is always a multiple of 4 and is less than 20. This means that the possible values for rand number can be 0, 4, 8, 12, or 16.

```
class constraint_41;
	rand bit[4:0]a;
	constraint c1{a<20;}
	constraint c2{a%4 == 0;}
endclass

constraint_41 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("a=%d",c1.a);
				end
		end
endmodule
```

Write a constraint to generate a pattern 5, 4, 3, 2, 1 (Reverse Count Pattern).

```
class constraint_42;
	rand bit[3:0] da[];
	constraint c1{da.size == 5;}
	constraint c2{foreach(da[i])
				da[i] == 5 - i;}
endclass

constraint_42 c1;

module test();
	initial
		begin
			c1=new;
			assert(c1.randomize());
			$display("da: %p",c1.da);
end
endmodule
```

Write a constraint to fill an array with the first 5 square numbers: 1, 4, 9, 16, 25.

```
class constraint_43;
	rand bit[7:0] da[];
	constraint c1{da.size == 5;}
	constraint c2{foreach(da[i])
				da[i] == (i+1)**2;}
endclass

constraint_43 c1;

module test();
	initial
		begin
			c1=new;
			assert(c1.randomize());
			$display("da: %p",c1.da);
		end
endmodule
```

Write a System Verilog code to randomize an array of size 5 such that the number must be a power of 2, and the randomized numbers in the array must be greater than or equal to 8.

```
class constraint_44;
        rand int da[];
        constraint c1{da.size == 5;}
        constraint c3{foreach(da[i])
                                        da[i] == 2**(i+3);}
endclass

constraint_44 c1;

module test();
        initial
                begin
                        c1=new;
                        assert(c1.randomize());
                        $display("da: %p",c1.da);
                end
endmodule
```

Generate a 8-bit random number such that the number is divisible by 3. Additionally ensure that the 8-bit number must have an even number of 1's.

```
class constraint_45;
	rand bit[7:0]a;
	constraint c1{a%3==0;}
	constraint c2{$countones(a) == 0 || $countones(a) ==
2 || $countones(a) == 4 || $countones(a) == 4 ||
$countones(a) == 6 || $countones(a) == 8;}
endclass

constraint_45 c1;

module test();
	initial
		begin
			repeat(5)
				begin
				c1=new;
				assert(c1.randomize());
				$display("a: %b",c1.a);
			end
		end
endmodule
```

Write a System Verilog code to generate a random 5-bit number that is divisible by 4 but not divisible by 6. Ensure the number falls within the range 0 to 31.

```
class constraint_46;
	rand bit[4:0]a;
	constraint c1{a inside {[0:31]};}
	constraint c2{(a%4 == 0) && (a%6 != 0);}
endclass

constraint_46 c1;

module test();
	initial
		begin
			repeat(5)
				begin
					c1=new;
					assert(c1.randomize());
					$display("a: %d",c1.a);
				end
		end
endmodule
```

Generate a random number ‘a’ between 1 to 50, the number must not be divisible by 2, 3, and 5.

```
class constraint_47;
      rand int a;
      constraint c1{a inside {[1:50]};}
      constraint c2{a%2 !=0 && a%3!=0 && a%5!=0;}
endclass

constraint_47 c1;

module test;
      initial
            begin
                  repeat(10)
                        begin
                              c1=new;
                              assert(c1.randomize());
                              $display("a: %d",c1.a);
                        end
            end
endmodule
```

Write a System Verilog code to generate an array of size 6, each element should be in between 10 and 20, and the sum of the elements must be 100.

```
class constraint_48;
	rand int da[];
	constraint c1{da.size == 6;}
	constraint c2{foreach(da[i])
				da[i] inside {[10:20]};}
	constraint c3{da.sum() == 100;}
endclass

constraint_48 c1;

module test();
	initial
		begin
			repeat(3)
				begin
					c1=new;
					assert(c1.randomize());
					$display("da: %p",c1.da);
				end
		end
endmodule
```

Write a System Verilog code to generate an array of size 10, each element should be in between 0 and 20, the sum of the elements in array must be an even number.

```
class constraint_49;
        rand int da[];
        constraint c1{da.size == 10;}
        constraint c2{foreach(da[i])
                                da[i] inside {[0:20]};}
        constraint c3{da.sum()%2 == 0;}
endclass

constraint_49 c1;

module test();
        initial
                begin
                        repeat(5)
                                begin
                                        c1=new;
                                        assert(c1.randomize());
                                        $display("da: %p",c1.da);
                                end
                end
endmodule
```

Generate an array of size 10, where the first and last elements are both constrained to be even numbers and the other numbers can be any number between 1 and 10.

```
class constraint_50;
      rand bit[7:0] da[];
      constraint c1{da.size == 10;}
      constraint c2{foreach(da[i])
                              if(i==0 || i==da.size-1)
                                    da[i]%2 == 0;
                              else
                                    da[i] inside {[1:10]};}
endclass

constraint_50 c1;

module test();
      initial
            begin
                  repeat(5)
                        begin
                              c1=new;
                              assert(c1.randomize());
                              $display("da: %p",c1.da);
                        end
            end
endmodule
```