SystemVerilog
Quick Reference Guide

Author: Shanthi V A

Email: techsupport_vm@maven-silicon.com

www.maven-silicon.com

Maven Silicon Confidential

All the presentations, books, documents [hard copies and soft copies] labs and projects [source code] that you are using and developing as part of the training course are the proprietary work of Maven Silicon and it is fully protected under copyright and trade secret laws. You may not view, use, disclose, copy, or distribute the materials or any information except pursuant to a valid written license from Maven Silicon

Table of Contents

1 DATA TYPES

1.1 Logic Data Type

```
module logic_test();

    parameters CYCLE = 20;
    logic q,q_b,d,clk,rst_n;

    initial
       begin
            clk = 0;                //procedural assignment
            rst_n = 0;

            forever
                #(CYCLE/2) clk = ~clk;
            @(negedge clk);
            rst_n = 'b1;
        end

    assign q_b = ~q;                //continuous assignment

    diff DUT(q,d,clk,rst_n);        //DUT Instantiation
                                    //DUT-No inout ports
endmodule : logic_test
```

1.2 Two State Data Type

- **bit** is the basic 2-state scalar type
- **byte** is an 8-bit signed integer 2-state type
- **shortint** is a 16-bit signed integer 2-state type
- **int** is a 32-bit signed integer 2-state type
- **longint** is a 64-bit signed integer 2-state type
- When a value is assigned to an object of any of these types, x or z values are converted to 0.

```
bit clk;            //2-state, bit values: 0 or 1
bit [31:0] vbus;    //2-state, 32 bits unsigned integer
int cnt;            //2-state, 32 bits signed integer
int unsigned cnt;   //2-state, 32 bits unsigned integer
byte crc_byte;      //2-state, 8 bits signed integer
shortint sint;      //2-state, 16 bits signed integer
longint lint;       //2-state, 64 bits signed integer

//clock generation - bit data type improves simulation performance
bit clk;            //default clk = 'b0;
always
    #5 clk = ~clk;
```

1.3 Struct Data Type

```
struct {bit [7:0]r,g,b; int color;} pixel;

   pixel.r = 8'd25;
   pixel.g = 8'd55;
   pixel.b = 8'd11;

   pixel.color = 32'd894;
```

1.4 Userdefined Data Type

```
typedef bit[31:0] word_t;
   word_t word1,word2;

struct {bit [7:0] r,g,b; int color;} pixel;
   pixel.r = 8'd25;
   pixel.g = 8'd55;
   pixel.b = 8'd11;
   pixel.color = 32'd894;

//user defined structures
typedef struct {bit [7:0] r,g,b; int color;} pixel_st;
   pixel_st samsung_pixel;
   pixel_st sony_pixel;

   samsung_pixel.r = 8'd25;
   Samsung_pixel.color = 32'd85;
   sony_pixel = '{8'd38, 8'd98, 8'd69, 32'd894};
```

1.5 Enumerated Data Type

```
enum {init,read,write,brd,bwr} fsm_state;
    fsm_state = init;
    fsm_state = read;
```

```
typedef enum {init,read,write,brd,bwr} fsm_state_e;

    fsm_state_e pre_state, nxt_state;

    pre_state = read;                  //Direct assignment
    fsm_state_e state = state.first;  //Method

initial
    forever
       begin
           $display(" %s : %d", state.name,state);
           if(state == state.last())
               break;
           state = state.next();
       end
```

In built methods available are

1. first ( ) //returns first element
2. last ( ) //returns last element
3. next ( ) //returns next element
4. prev ( ) //returns previous element
5. next (N) //returns Nth next element
6. prev (N) //returns Nth previous element

1.6 String Data Type

```
string str;

initial
    begin
        str = "Maven Silicon";
        str = str.tolower();                                    //maven_silicon
        $display("character in 5th position is %s", str.getc(5));
        str.putc(5, "-");                                       //"_" is replaced with"-"
        $display("%s", str.substr(6, str.len-1));               //"silicon" is printed
        str = {str, ".com"};                                    //maven-silicon.com
        str = {{3{"w"}},".", str};                              //www.maven-silicon.com
        disp($sformatf("https://%s", str));
    end

task disp(string s);
        $display("at time t = %0t, %s", $time, s);
                                //at time t = 0, https://www.maven-silicon.com
endtask : disp
```

1.7 Package

```
package pkg;
    int no_of_trans;

    function void display(string s);
        $display($time, "%s, n = %0d",s,no_of_trans);
    endfunction

endpackage
```

```
module mem;
    import pkg::*;

    initial
        begin
            #1;
            no_of_trans = 10;

            #5;
            display("FROM MODULE MEM");
        end
endmodule : mem
```

```
module cpu;
    import pkg::*;

    initial
        begin
            #2;
            display("FROM MODULE CPU");

            #3;
            no_of_trans = 20;
        end
endmodule : cpu
```

Result :
#2, FROM MODULE CPU, n = 10
#6, FROM MODULE MEM, n = 20

2 AGGREGATE DATA TYPES

2.1 Packed Array

Syntax

```
<array_type> <Packed_dimension> array_name;
```

```
bit[7:0][3:0]packed_array;          //8 nibbles packed into 32 bits
bit[3:0][7:0]packed_byte;           //4 bytes packed into 32 bits

packed_array = 32'hABCD_1234;

$display( packed_array,             //show all 32 bits
          packed_array[0],          //least significant nibble
          packed_array [7][3]);     //most significant bit
```

2.2 Multi Dimensional Array

Syntax :

```
<array_type> <Packed_dimension> array_name <unpacked_dimension>;
```

```
bit [3:0] [7:0] byte_ma[0:2];           //packed: 3*32-bit
bit [7:0] [3:0] nibbles;                //packed array of nibbles

byte_ma[0] = 32'd255;
byte_ma[0][3] = 8'd255;
byte_ma[0][1][6] = 1'b1;

nibbles  = byte_ma[0];                  //copy packed values
```

2.3 Array copy and Compare

```
bit [31:0] packet1[8] = {'h21,'h22,'h31,'h32,'h41,'h42,'h51,'h52};
bit [31:0] packet2[8] = {'h22,'h21,'h32,'h31,'h42,'h41,'h51,'h52};

initial
   begin
      if(packet1 == packet2)      //compares all values from both the packets
         $display("packet1 is equal to packet2");
      else
         $display("packet1 is not equal to packet2");

      packet1 = packet2;          //copy packet2 to packet1

      if(packet1[0:1] == packet2[0:1])
                                  //compares last 2 values from both the packets
         $display("packet1 is equal to packet2");
      else
         $display("packet1 is not equal to packet2");
   end
```

2.4 Dynamic Array

Syntax :

```
<array_type> array_name [ ];
```

```
int da1[],da2[];

initial
    begin
        da1 = new[5];          //Allocate 5 elements
        foreach(da1[j])        //Initializing
            begin
                da1[j] = j;
                $display("%d",da1[j]);
            end
        da2 = da1;             //copying
        da1 = new[20](da1);    //Allocate 20 new elements & copy existing values
        $display("%p", da1);
        da1 = new[100];        //Allocate 100 new elements - old values are lost
        $display("%d",$size(da1));
        da1.delete();          //Delete all elements
    end
```

2.5 Queues

Syntax :

```
<array_type> array_name [$];
```

```
int q1[$] = '{1,3,4,5,6};
int q2[$] = '{2,3};
int k = 2;

q1.insert(1,k);         //{1,2,3,4,5,6}   Insert k @1
q1.delete(1);           //{1,3,4,5,6}     Delete the element @1
q1.push_front(7);       //{7,1,3,4,5,6}   insert at front
k = q1.pop_back();      //{7,1,3,4,5}     k = 6
q2.push_back(4);        //{2,3,4}         insert at back
k = q2.pop_front();     //{3,4}           k = 2

foreach(q1[i])
    $display(q1[i]);    //display the elements of the entire queue

q2.delete();            //delete entire queue
```

2.6 Associative Array

Syntax :

```
<array_type> array_name [*];
```

*** represents the index which can be any data type**

```
module test();
    int mem[int];
    int idx;

    initial
        begin
            mem[3]=5;
            mem[2]=1;
            mem[35]=60;
            mem[2000]=89;
            if(mem.exists(35))                  //exists method
                $display("entry exists in mem, whose value is %d", mem[35]);
            else
                $display("no entry");
            if(mem.first(idx))                  //first method
                $display("first entry %d is made in address %d",mem[idx],idx);
            else
                $display("no entry");
            $display("number of entries in array is %0d",mem.num);
        end
endmodule : test

//entry exists in mem, whose value is 60
//first entry 1 is made in address 2
//number of entries in array is 4
```

2.7 Array Methods

2.7.1 Array Sorting Methods

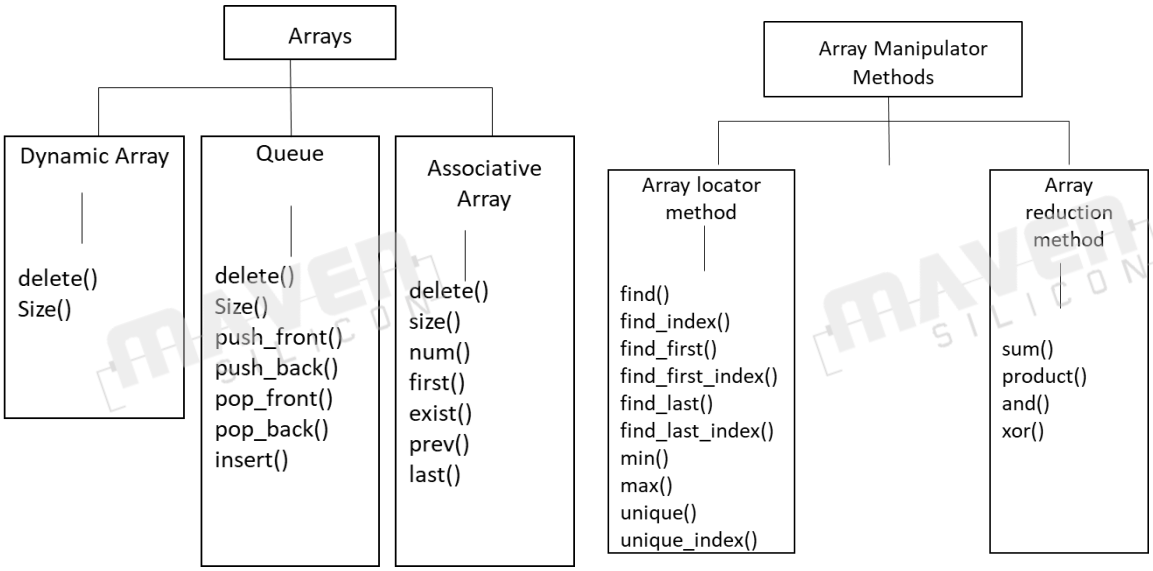
```
//Array sorting and ordering methods
int packet[10] = {1,3,4,6,2,7,9,5,10,8};

packet.reverse();                   //{8, 10, 5, 9, 7, 2, 6, 4, 3, 1}
packet.sort();                      //{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
packet.rsort();                     //{10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
packet.shuffle();                   //{2, 3, 10, 5, 7, 1, 8, 9, 6, 4}
```

2.7.2 Array Reduction Methods

```
//Array reduction methods
int packet[5] = {1,2,3,4,5};
int result;

result = packet.sum();                    //15 - sum of all elements
result = packet.product();                //120 - product of all elements
result = packet.and();                    //0 - bitwise AND of all elements
result = packet.xor();                    //1 -  bitwise XOR of all elements

int packet[10] = {1,2,3,4,5,6,7,8,9,10};
int result;

result = packet.sum with(int'(item%2 ==0));//5 - count of even numbers
result = packet.sum with ((item %2 == 0 )* item);//30 - sum of even numbers
```

2.7.3 Array Locator Methods

```
//Array locator methods: min, max, unique
int f[10] = {1,6,5,2,6,7,8,3,4,7};
int tq[$];

tq = f.min();              //{1}
tq = f.max();              //{8}
tq = f.unique();           //{1,2,3,4,5,6,7,8}

tq = f.find_first with(item%2 == 1);                  //{1}
tq = f.find_first_index with(item %2 == 1);           //{0}
tq = f.find_last with(item%2 == 1);                   //{7}
tq = f.find_last_index with(item%2 == 1);             //{9}
```



3 Tasks & Functions

3.1 Task

Task Syntax :

```
task <life_time> <task_identifier>(<formal argumets>);
        <task body>
endtask : <task_identifier>
```

```
task static double(int a , string s);
    #5;
    a = a*2;
    $display($time,"%s : %d",s,a);
endtask : double

initial
    fork
        begin
            double(5,"FROM THREAD 1 : ");
        end

        begin
            #2;
            double(4,"FROM THREAD 2 : ");
        end
    join
```

Result :

```
#5 FROM THREAD 2 : 8
#7 FROM THREAD 2 : 16
```

```
task automatic double(int a , string s);
    #5;
    a = a*2;
    $display($time,"%s : %d",s,a);
endtask : double

initial
    fork
        begin
            double(5,"FROM THREAD 1 : ");
        end

        begin
            #2;
            double(4,"FROM THREAD 2 : ");
        end
    join
```

Result :

```
#5 FROM THREAD 1 : 10
#7 FROM THREAD 2 : 8
```

3.2 Function

Function Syntax :

```
function <life_time> <return_datatype> <function_identifier>(<formal_arguments>);
            <function implementation>
endfunction : <function_identifier>
```

```
module test();
    int ma  = 20;
    function void pass_by_value(int fa); // In module life time of a method is static by default
       fa = fa*2;
       $display(" Inside the function pass_by_value fa = %d", fa);
    endfunction : pass_by_value

    function automatic void pass_by_ref(ref int fa);
       fa = fa*2;
       $display(" Inside the function pass_by_ref fa = %d", fa);
    endfunction : pass_by_ref

    initial
       begin
          pass_by_value(ma);
          $display("after calling the function pass_by_value ma = %d",ma);
          pass_by_ref(ma);
          $display("after calling the function pass_by_ref ma = %d",ma);
       end
endmodule : test

//Inside the function pass_by_value fa = 40
//after calling the function pass_by_value ma = 20
//Inside the function pass_by_ref fa = 40
//after calling the function pass_by_ref ma = 40
```

4 Interface

Syntax:

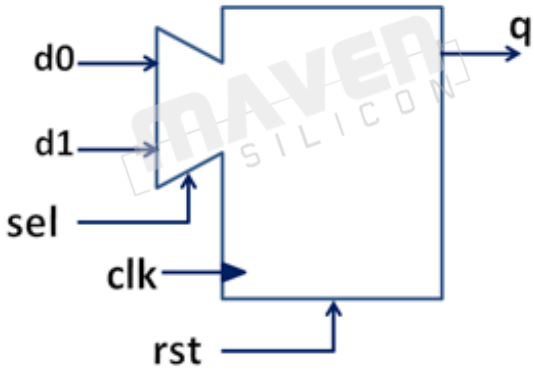
```
interface <interface_identifier>(<port_list>);
          <interface definition>
endinterface : <interface_identifier>
```

```
interface dff_if(input clk);

   logic d0,d1,sel,rst,q;

   parameter  thold  = 2,   //output_skew
              tsetup = 4;   //input_skew

   clocking cb @(posedge clk);
      default input #(tsetup) output #(thold);
      output d0;
      output d1;
      output rst;
      output sel;
      input q;
   endclocking : cb

   modport DUV (input d0,d1,rst,sel,clk, output q);

endinterface : dff_if
```

```
module dff(dff_if.DUV duv_if);

 //RTL code

  ...

endmodule : dff
```

```
module testcase(dff_if.TEST test_if);

  //TB Code

  ...

endmodule : testcase
```

```
`timescale 10ns/1ns

module top();

   bit clk;

   always
      #10 clk = ~clk;

   dff_if IF(clk);

   dff RTL(IF);

   testcase TB(IF);

endmodule : top
```



5 OOP

5.1 Class

Syntax:

```
class <class_identifier>;
  <class_properties>;
  <class_methods>;
endclass : <class_identifier>
```

5.2 Class Declaration & Creation of Object

```
class account_c; //class datatype
    int balance;

    function int summary;
        return balance;
    endfunction : summary

endclass : account_c

account_c acnt_h;
initial
        acnt_h = new;
```

```
class transaction;
    bit[31:0] src,dst;

    function new();
        src = 5;
        dst = 3;
    endfunction : new

endclass : transaction

transaction trans_h;

module test;
    initial
        trans_h = new();

endmodule : test
```

```
class transaction;
    bit[31:0] src,dst;

    function new(int x,int y=3);
            src = x;
            dst = y;
    endfunction : new

endclass : transaction

transaction trans_h;

module test;
    initial
        trans_h = new(5); //y uses default

endmodule : test
```

5.3 this Keyword

```
class account_c;
    int amount;

    function new(int amount);
        this.amount = amount;
    endfunction : new

endclass : account_c
```

5.4 Accesing Class Properties & Methods

```
class account_c;
    int balance;                    //class property

    function int summary;           //class method
        return balance;
    endfunction : summary

    task deposit(input int pay);    //class method
        balance = balance+pay;
    endtask : deposit

endclass : account_c
```

```
account_c acnt_h = new();

acnt_h.deposit(2000);                          //accessing class method
$display("balance is %d",acnt_h.summary());    //accessing class method
$display("balance is %d",acnt_h.balance);      //accessing class member
```

5.5 Handle Assignment & Copying the object

Handle Assignement:

```
class transaction;
    int data;
endclass : transaction

transaction trans_h1;     //handle1
transaction trans_h2;     //handle2

module test();
    initial
       begin
           trans_h1 = new();
           trans_h1.data = 10;
           trans_h2 = new();
           trans_h2.data = 20;
           trans_h2 = trans_h1;
           trans_h2.data = 50;
           $display(trans_h1.data);
           //display "50"
       end
endmodule  : test
```

Copying the object:

```
class transaction;
    bit [3:0] data;
    bit [15:0] addr;
    bit [15:0] mem;
endclass : transaction

transaction trans_h1;
transaction trans_h2;

module test();
    initial
       begin
           trans_h1 = new;
           trans_h2 = new trans_h1;
       end
endmodule : test
```

5.6 Deep Copy

```
class sub;
   int obj;

   function sub copy();
      copy = new();
      copy.obj = this.obj;
   endfunction : copy

endclass : sub
```

```
class trans;
   int data;
   sub sub_h = new();
   function trans copy();
      copy = new();
      copy.data = this.data;
      copy.sub_h = this.sub_h.copy;
   endfunction : copy
endclass : trans
```

```
trans t_h1, t_h2;
module test();
   initial
      begin
         t_h1 = new();
         t_h1.data = 4;
         t_h1.sub_h.obj = 5;
         t_h2 = t_h1.copy;
         t_h2.sub_h.obj = 10;
      end
endmodule : test
```

6 Advanced OOP

6.1 Inheritance

```
class transaction;
   bit[7:0] header,payload[],parity;
endclass : transaction
```

```
class error_trans extends transaction;
   bit error_parity;
endclass : error_trans
```

```
error_trans error_h;
error_h = new();
error_h.header = 42;
error_h.error_parity = 1;
```

6.2 Super Keyword

```
class account_c;
   int balance = 0;     //overridden by extended class property

   function int summary;
      return balance;
   endfunction : summary
   ...
endclass : account_c
```

```
class linked_account extends account_c;
   int balance;                 //overrides base class
                                //property balance
   function int summary;
      return balance + super.balance;
   endfunction : summary

endclass : linked_account
```

```
class account_c;
    int balance;

    function new(int pay);
          balance = pay;
    endfunction : new

endclass : account_c

class linked_account extends account_c;

    function new(int value);
       super.new(value);
    endfunction : new

endclass : linked_account

linked_account lin_h;
```

```
module top;

  initial
      begin
          lin_h = new(500);
      end

endmodule : top
```

6.3 Static Properties

```
class packet;
    static int ID = 0;
    int aid = 0;
    ...
endclass : packet

class generator;
    packet pkt_h = new();

    task start();
        pkt_h.ID++;
        pkt_h.aid++;
    endtask : start

endclass : generator
```

```
generator gen[4];

foreach (gen[i])
   gen[i] = new;

gen[0].start();      //ID=1,aid=1
gen[1].start();      //ID=2,aid=1
repeat(10)
   gen[2].start();   //ID=12,aid=10
gen[3].start();      //ID=13,aid=1
```

6.4 Chaining Constructor

```
class transaction;
   int a;

   function new(int a);
       this.a = a;
   endfunction : new

endclass : transaction

class extnd_trans extends transaction(7);
   int b;
endclass : extnd_trans

extnd_trans ext_h;

module test();
   initial
       begin
           ext_h = new();
       end

endmodule : test
```

6.5 Static Methods

```
class transaction;
    static int i;
    int j;

    static function void stat_fun();
        int b;
        i++;
        b++;
        $display("stat_fun,i = %0d, b = %0d",i,b);
    endfunction : stat_fun

    function static void fun_stat();
        int a;
        j++;
        a++;
        $display("fun_stat,j = %0d, a = %0d",j,a);
    endfunction : fun_stat

endclass : transaction
```

```
transaction t[];

    initial
        begin
            t = new[5];
            foreach(t[i])
                begin
                    t[i] = new();
                    t[i].stat_fun();
                end
            foreach(t[i])
                begin
                    t[i].fun_stat();
                        end
        end
```

```
stat_fun,i = 1,b = 1
stat_fun,i = 2,b = 1
stat_fun,i = 3,b = 1
stat_fun,i = 4,b = 1
stat_fun,i = 5,b = 1

fun_stat,j = 1, a = 1
fun_stat,j = 1, a = 2
fun_stat,j = 1, a = 3
fun_stat,j = 1, a = 4
fun_stat,j = 1, a = 5
```

6.6 Polymorphism

```
class parent_trans;
    virtual task send();
        //drives good parity
    endtask : send

endclass : parent_transaction

class error_trans extends parent_trans;
    virtual task send();
        //drives bad parity
    endtask : send

endclass : error_trans
```

Test1:

```
parent_trans trans_h;
trans_h = new();
trans_h.send();
```

Test2:

```
parent_trans trans_h;
error_trans err_h;
err_h = new();
trans_h = err_h;
trans_h.send();
```

6.7 $cast

```
class parent_trans;
    bit[31:0] header, parity;
    virtual function void calc_par();
        .........
    endfunction : calc_par
endclass : parent_trans

class error_trans extends parent_trans;
    bit bad_crc;
    virtual function void calc_par();
        .........
    endfunction : calc_par
endclass : error_trans
```

```
parent_trans par_h;
error_trans e1_h, e2_h;

e1_h = new(); //Allocate extended object
par_h = e1_h; //Assign to base handle
par_h.calc_par(); //calculate CRC
e2_h = par_h;     //Error! Not allowed

$cast(e2_h,par_h);//Allowed assignment
if ($cast(e2_h,par_h)) //Check if legal
e2_h.calc_par();
```

6.8 Typed Constructor

```
class transaction;
    int a;
endclass : transaction

class extnd_trans extends transaction;
   int b;
endclass : extnd_trans

transaction trans_h;

module test();

    initial
       begin
          trans_h = extnd_trans::new();
       end

endmodule : test
```

6.9 Virtual Class & Pure Virtual Methods

```
virtual class packet;
    bit [31:0] arr[0:3];
   pure virtual function void divider_pkt_one();
endclass : packet
```

```
class IPV4 extends packet;
    bit[7:0] arr_4 [7:0];

    virtual function void divider_pkt_one();
        arr_4[0] = super.arr[0] [7:0];
        arr_4[1] = super.arr[0] [15:8];
        arr_4[2] = super.arr[0] [23:16];
        arr_4[3] = super.arr[0] [31:24];
   endfunction : divider_pkt_one

endclass : IPV4
```

```
class IPV6 extends packet;
    bit[15:0] arr_2[0:1];

    virtual function void divider_pkt_one();
        arr_2[0] = super.arr[0] [15:0];
        arr_2[1] = super.arr[0] [31:16];
    endfunction : divider_pkt_one

endclass : IPV6
```

6.10 Parametrized Class

```
//Parameterized class
class vector #(int size = 1);
   bit [size-1:0] a;
endclass : vector

//Instances of this class can then be instantiated like modules
or interfaces
vector #(10) vten;              //object with vector of size 10
vector #(.size(2)) vtwo;        //object with vector of size 2
typedef vector#(4) vfour;       //class with vector of size 4
```

7 Constraints & Randomization

7.1 Randomize & Post_randomize Methods

```
class transaction;
    rand bit[7:0] data;
    rand bit[1:0] ch;

    function void post_randomize;
        $display("data is %d",data);
        $display("channel is %d",ch);
    endfunction : post_randomize

endclass : transaction

module test();

    initial
        begin
            int success;
            transaction trans_h = new();
            success = trans_h.randomize();
        end
endmodule : test
```

7.2 Constraints

Syntax:

```
constraint constraint_identifier
                        {constraint expression;}
```

```
class transaction;
    rand bit[15:0] pktlength;
    constraint undersize {pktlength <= 16'd64;}
endclass : transaction
```

```
class transaction;
    rand bit[3:0]addr;
    constraint addr_range {addr inside{3,7,[11:15]};}
endclass : transaction
```

7.3 Distribution Constraint

```
class transaction;
    rand int Vlantag;
    constraint tag{Vlantag dist{7:=5,[11:20]:=3,[26:30]:/5};}
endclass : transaction
```

7.4 Conditional Constraint

7.4.1 Using Implication

Syntax:

```
constraint constraint_identifier
                        {[expression] -> {constraint set};}
```

```
class transaction;
    rand int Vlantag;
    bit mode;
    constraint tag{mode == 1'b1 -> Vlantag < 96;
                   mode == 1'b0 -> Vlantag < 1248;}
endclass : transaction
```

7.4.2 Using if else

```
class transaction;
    rand int Vlantag;
    bit mode;
    constraint tag{if (mode == 1'b1)
                       Vlantag < 96;
                   else if (mode == 1'b0)
                       Vlantag < 1248;}
endclass : transaction
```

7.5 rand_mode() & constraint_mode()

```
class transaction;
   rand bit[15:0] pktlength;
   constraint undersize {pktlength <= 16'd64;}
endclass : transaction

transaction trans_h = new;
module test();
   initial
      begin
         int success;
         if(trans_h.pktlength.rand_mode())
            begin
               for(int i=0; i<16;i++)
               success = trans_h.randomize();
            end
         trans_h.undersize.constraint_mode(0);
         success = trans_h.randomize();
         trans_h.undersize.constraint_mode(1);
         success = trans_h.randomize();
      end
endmodule : test
```

7.6 Constraint & Inheritance

```
class transaction;
    rand bit[15:0] pktlength;
    constraint oversize {pktlength >= 16'hff00;}
endclass : transaction

class trans_ext extends transaction;
    constraint undersize {pktlength <= 16'hff50;}
endclass : trans_ext

trans_ext tex_h = new;

module test();
    initial
        begin
            assert(tex_h.randomize();
            $display("pktlength is %d",tex_h.pktlength);
        end
endmodule : test
```

randomizes **pktlength** using the constraints **undersize** and **oversize**

7.7 Constraint Overriding

```
class transaction;
    rand bit[15:0] pktlength;
    constraint oversize {pktlength >= 16'hff00;}
endclass : transaction

class trans_ext extends transaction;
    constraint oversize {pktlength <= 16'hff50;}
endclass: trans_ext

trans_ext tex_h = new;

module test();
    initial
        begin
            int success;
            success = tex_h.randomize();
            $display("pktlength is %d",tex_h.pktlength);
        end
endmodule : test
```

randomizes **pktlength** by overriding base class constraint **oversize** with extended class constraint **oversize** **pktlength <= 16'hff50**

7.8 Inline Constraint

```
class inline;
    rand bit[7:0]x,y,z;
    constraint c {z == x+y;}
endclass : inline

inline in_h;

module test;
    initial
        begin
            in_h = new;
            if(!(in_h.randomize() with {x<y;}))
               $display("randomization failed");
        end
endmodule : test
```

7.9 Static Constraint

```
class packet;
   rand  bit [3:0] addr;
   static constraint addr_range {addr>5;}
endclass : packet

packet pkt_h1;
packet pkt_h2;

module test;
    initial
       begin
          pkt_h1 = new();
          pkt_h2 = new();

          assert(pkt_h1.randomize()); // addr > 5

          pkt_h2.addr_range.constraint_mode(0);

          assert(pkt_h1.randomize()); // addr inside {[0:15]}
          assert(pkt_h2.randomize()); // addr inside {[0:15]}
       end
endmodule : test
```

7.10 Solve Before Construct

```
class transaction;

    rand bit a;
    rand bit[2:0] b;
    constraint valid{(a == 0) -> (b==0);
                     solve a before b;}
endclass : transaction

transaction tr_h;

module test();

    initial
       begin
          tr_h = new();
          assert(tr_h.randomize());
       end

endmodule : test
```

a	b	Probability
0	0	1/2
0	1	0
0	2	0
0	3	0
0	4	0
0	5	0
0	6	0
0	7	0
1	0	1/16
1	1	1/16
1	2	1/16
1	3	1/16
1	4	1/16
1	5	1/16
1	6	1/16
1	7	1/16

8 Threads, Mailbox, Events & Semaphore

8.1 Threads

8.1.1 fork – join

```
module test();

    initial
        begin
            #10;
            fork
                begin
                    #20; // Thread_1
                end
                begin
                    #40; // Thread_2
                end
                begin
                    #30; // Thread_3
                end
            join
            $display("exit is at time t = %t", $time);
        end

endmodule : test
```

```
exit is at time t = 50
```

8.1.2 fork – join_any

```
module test();

    initial
        begin
            #10;
            fork
                begin
                    #20; // Thread_1
                end
                begin
                    #40; // Thread_2
                end
                begin
                    #30; // Thread_3
                end
            join_any
            $display("exit is at time t = %t", $time);
        end

endmodule : test
```

```
exit is at time t = 30
```

8.1.3 fork – join_none

```
module test();

    initial
        begin
            #10;
            fork
                begin
                    #20; // Thread_1
                end
                begin
                    #40; // Thread_2
                end
                begin
                    #30; // Thread_3
                end
            join_none
            $display("exit is at time t = %t", $time);
        end

endmodule : test
```

exit is at time t = 10

8.2 Mailbox

```
mailbox #(transaction)mbx;          //Declare a mailbox
mbx = new();                        //allocate mailbox-object
mbx.put(trans_h);                   //put transaction into mailbox
mbx.get(trans_h);                   //get transaction and remove
success = mbx.try_get(trans_h);     //Non-blocking version
mbx.peek(trans_h);                  //get trans_h but don't remove-blocking
success = mbx.try_peek(trans_h);    //Non-blocking version
count = mbx.num();                  //Number of transaction in mailbox
```

```
class generator;
   trans trans_h;
   mailbox #(trans) gen2drv2;
   function new(mailbox #(trans)gen2drv);
      this.gen2drv2 = gen2drv;
   endfunction : new

   task start;
      fork
         repeat(10) begin
            trans_h = new();
            assert(trans_h.randomize());
            gen2drv2.put(trans_h);
         end
      join_none
   endtask : start
endclass : generator
```

```
class driver;
   trans trans_h;
   mailbox #(trans) gen2drv3;
   function new(mailbox #(trans) gen2drv);
      this.gen2drv3=gen2drv;
   endfunction : new

   task start;
      fork
         repeat(10) begin
            gen2drv3.get(trans_h);
            @(posedge duvif.cb.en);
            //Drive the stimulus
         end
      join_none
   endtask : start
endclass : driver
```

```
class env;
   mailbox #(trans)gen2drv1 = new();
   generator gen = new(gen2drv1);
   driver dr = new(gen2drv1);
   task start;
      gen.start;
      dr.start;
   endtask:start
endclass:env
```

8.3 Events

```
event ev;                  // Declare an event
-> ev;                     // Tigger an event
@(ev);                     // Block Process, wait for future event
wait(ev.triggered);        // Block Process, wait for event,
                           // including this timeslot
                           // Reduces race conditions
driver = new(ev);          // pass event into methods.
```

```
module test();
   event ev;

   initial
      begin
         $display("about to drive ctrl after 10 time units");
         #10;
         $display("control is driven at time, t = %t", $time);
         -> ev;
      end

   initial
      begin
         $display("about to drive data");
         #10;
         @(ev);
         $display("data is driven at time, t = %t", $time);
      end
endmodule : test

//about to drive ctrl after 10 time units
//about to drive data
//control is driven at time, t = 10
```

```
module test();
    event ev;

    initial
       begin
          $display("about to drive ctrl after 10 time units");
          #10;
          $display("control is driven at time, t = %t", $time);
          -> ev;
       end
    initial
       begin
          $display("about to drive data");
          #10;
          wait(ev.triggered);
          $display("data is driven at time, t = %t", $time);
       end

endmodule : test
//about to drive ctrl after 10 time units
//about to drive data
//control is driven at time, t = 10
//data is driven at time, t = 10
```

8.4 Semaphores

```
semaphore sem;
sem = new(optional_initial_keycount = 0);
sem.get(optional_num_keys = 1);
sem.put(optional_num_keys = 1);
```

```
module automatic test;

   driver dr[2];
   semaphore sem;

   initial
      begin
         foreach (dr[i])
            dr[i]= new();
         sem  = new(1);
         fork
            dr[0].send();
            dr[1].send();
         join
      end

endmodule : test
```

Allocate a semaphore with 1 Key

```
class driver

   task send();
      sem.get(1);
      @bus.cb;
      bus.cb.addr <= t.addr;
      bus.cb.kind <= t.kind;
      bus.cb.data <= t.data;
      sem.put(1);
   endtask : send

endclass : driver
```

Wait for bus to be available

Allow others to drive the bus

9 Functinal Coverage

9.1 Covergoup Definition & Creating the instance

Synatx:

```
covergroup cg_group(<optional arguments>);
                < definition of covergroup >
endgroup: cg_group
```

```
 cg_group = new();      //() is optional
```

9.2 Coverpoints

Syntax:

```
[ Coverpoint_id :] coverpoint variable_id ;
```

```
[ Coverpoint_id :] coverpoint variable_id {bins_def}
```

```
crossAB : cross a,b;
```

```
class coverage_model;

   logic  [7:0] address;
   logic  [2:0] opcode;
   logic valid;

   covergroup cg_group;
       c1 : coverpoint opcode;
       c2 : coverpoint address;
       c1_X_VAL : cross c1,valid;
   endgroup : cg_group

   function new();
       cg_group = new;
   endfunction : new

endclass : coverage_model
```

```
class coverage_model;
   bit [2:0] var;

   covergroup cg;
      coverpoint var {
                bins sca   = {0,3,4,6,7};   //creates single bin
                bins vec[] = {1,2,5};       //3 bins v[1],v[2] and v[5]
                   }
   endgroup : cg

   function new();
         cg = new();
   endfunction : new

endclass  : coverage_model
```

```
class coverage_model();
    bit [3:0] opcode;
    bit [3:0] bus;
    bit rst;

    covergroup cg;
        c1 : coverpoint opcode;
        c2 : coverpoint bus;
        c1_x_val  : cross c1,c2;
        c1_x_rst  : cross c1,rst; // cross with rst
    endgroup : cg

endclass : coverage_model
```

```
bit [7:0] v_a, v_b;
    covergroup cg;
       a:coverpoint v_a   {
                  bins a1 = {[0:63]};
                  bins a2 = {[64:127]};
                  bins a3 = {[128:191]};
                  bins a4 = {[192:255]}; }

       b:coverpoint v_b   {
                  bins b1 = {[0]};
                  bins b2 = {[1:84]};
                  bins b3 = {[85:169]};
                  bins b4 = {[170:255]}; }

       c:cross  a,b {
                  bins c1 != binsof(a) intersect {[100:200]};
                                                          //4 cross products
                  bins c2 = binsof(a.a2) || binsof(b.b2);
                                                          //7 cross products
                  bins c3 = binsof(a.a1) && binsof(b.b4); }
                                                          //1 cross product
    endgroup : cg
```

```
    bit[2:0]y;
    bit z;
    bit clock;

    covergroup cg (int i) @(posedge clock);
        option.per_instance = 1;//enables to view the coverage report for each instance
        option.at_least = i;    //specifies minimum number of hits for coverpoint

        CP_Y : coverpoint y {
                        bins y1 = {0};
                        bins y2 = {1};
                        bins y3 = {2};
                        bins y4 = {3};
                        bins y5 = {4};
                        bins y6 = {5};
                        bins y7 = {6};
                        bins y8 = {7};
                              }

         CP_Z : coverpoint z;
    endgroup : cg
```

9.3 Default , Illegal & Ignore Bins

9.3.1 Default Bin

```
class coverage_model();
    reg [1:0] a;
    reg [3:0] b;
    reg c;
    covergroup cg;
        c1: coverpoint b {
                        bins b1     = {[9:12]};   // 1 bin b1
                        bins b2[]   = {[13:15]};  // 3 bins  b2[13], b2[14], b2[15]
                        bins rest_b = default;    // 1 bin {0,...,8} not in cross
                          }
        c2: coverpoint c;                         // 2 bins
        A_x_B_x_C: cross a,c1,c2;                 // 32 bins = a(4) x B(4) x C(2)
    endgroup : cg

endclass : coverage_model
```

9.3.2 Ignore Bin

```
bit [3:0] bcd_value;

    covergroup cg;
        coverpoint bcd_values{
                    ignore_bins IGB = {[10,12,14]};}
                    //These bins will be excluded from coverage
    endgroup : cg
```

9.3.3 Illegal Bin

```
bit [3:0] bcd_value;

    covergroup cg;
       coverpoint bcd_values{
                  illegal_bins ILLB = {[10,11,12,13,14,15]};}
                  //Gives error if seen
       endgroup : cg
```

9.4 Conditional Expression iff

```
class coverage_model();
   reg [1:0] a;
   reg [3:0] b;
   reg c, empty, undersize, oversize, reset;
   logic [63:0] queue;
   logic [3:0] priority;
   logic [15:0] port_id;

   covergroup cg;
      c1 : coverpoint queue iff(!empty);
      c2 : coverpoint priority iff(undersize|oversize);
      c3 : coverpoint port_id iff(!reset);
   endgroup : cg

endclass : coverage_model
```

9.5 Transition Bins

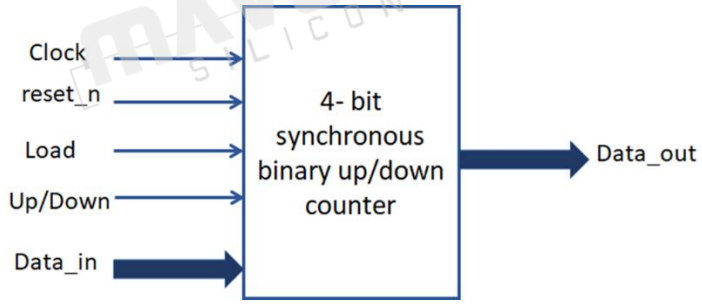
```
   bit [1:0]a;
   bit clk;

   covergroup cg @(posedge clk);
      coverpoint a {
                bins t_a = (1 => 3);
                bins t_b = (3 => 2);
                   }
   endgroup : cg
```

9.6 Sampling the Covergroup

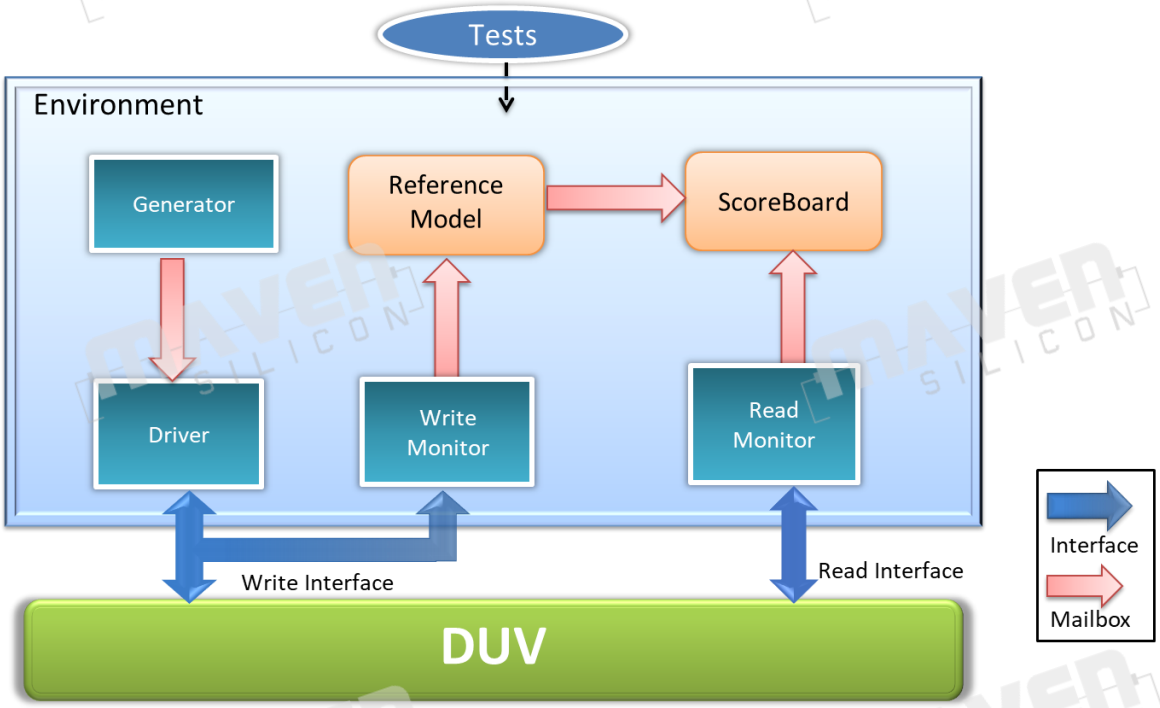
```
class cvg_ctrl;
    bit[3:0] g;

    covergroup cg_group;
        cp : coverpoint g;
    endgroup : cg_group

    function new();
        cg_group = new;
    endfunction : new

endclass : cvg_ctrl

cvg_ctrl cov_h;

module test();
    initial
        begin
            cov_h = new();
            while(cov_h.cg_group.get_coverage() < 100)
                begin
                    cov_h.g = $random;
                    cov_h.cg_group.sample();
                end
            $display("coverage now at 100");
        end
endmodule : test
```

10 Verification of Modulo12 Loadable up-down Counter

10.1 Block Diagram



10.2 TB Environment



10.3 Interface

```
interface count_if(input bit clock);

    //Interface Signals
    logic [3:0]din;
    logic [3:0]count;
    logic load;
    logic up_down;
    logic resetn;

    // Driver Clocking Block
    clocking dr_cb@(posedge clock);
        default input #1 output #1;
        output din;
        output load;
        output up_down;
    endclocking

    // write monitor clocking block
    clocking wr_cb@(posedge clock);
        default input #1 output #1;
        input din;
        input load;
        input up_down;
    endclocking
```

```
// Read monitor clocking block
    clocking rd_cb@(posedge clock);
        default input #1 output #1;
        input count;
    endclocking

    // Driver
    modport DRV(clocking dr_cb);

    // Write Monitor
    modport WR_MON(clocking wr_cb);

    // Read Monitor
    modport RD_MON(clocking rd_cb);

endinterface
```

10.4 Transaction Class

```
class count_trans;

    //Interface Signals
    rand logic [3:0]data_in;
    rand logic load;
    rand logic up_down;
    logic resetn;
    logic [3:0]count;

    //Constraints
    constraint C1 {data_in inside{[2:8]};}
    constraint C3 {load dist {1 := 30, 0 := 70};}
    constraint C4 {up_down dist {0 :=50,1:= 50};}

    //Display method
    virtual function void display(input string s);
        begin
            $display("..........................%s..........................",s);
            $display("Up_down = %d",up_down);
            $display("load = %d",load);
            $display("data_in = %d",data_in);
            $display("count = %d",count);
            $display("resetn = %d",resetn);
            $display("..........................................................");
        end
    endfunction
endclass
```

10.5 Transactors

10.5.1 Generator

```
class count_gen;
    count_trans trans_h;
    count_trans data2send;

    mailbox #(count_trans)gen2dr;

    function new(mailbox #(count_trans)gen2dr);
        this.gen2dr = gen2dr;
        this.trans_h = new;
    endfunction

    virtual task start();
        fork
            begin
                for(int i=0; i<no_of_transactions; i++)
                    begin
                        assert(trans_h.randomize());
                        data2send = new trans_h;
                        gen2dr.put(data2send);
                    end
            end
        join_none
    endtask
endclass
```

10.5.2 Write Driver

```
class count_driver;

    virtual count_if.DRV dr_if;

    count_trans data2duv;

    mailbox #(count_trans)gen2dr;

    function new(virtual count_if.DRV dr_if,
                 mailbox #(count_trans)gen2dr);
        begin
            this.dr_if  = dr_if;
            this.gen2dr = gen2dr;
        end
    endfunction
```

```
    virtual task drive();
        begin
            @(dr_if.dr_cb);
                dr_if.dr_cb.load <= data2duv.load;
                dr_if.dr_cb.din <= data2duv.data;
                dr_if.dr_cb.up_down <= data2duv.up_down;
        end
    endtask

    virtual task start();
        fork
            forever
                begin
                    gen2dr.get(data2duv);
                    drive();
                end
        join_none
     endtask

endclass
```

10.5.3 Write Monitor

```
class count_wr_mon;

    virtual count_if.WR_MON wrmon_if;

    count_trans data2rm,wr_data;

    mailbox #(count_trans)mon2rm;

    function new(virtual count_if.WR_MON wrmon_if,
                 mailbox #(count_trans)mon2rm);
        begin
            this.wrmon_if = wrmon_if;
            this.mon2rm = mon2rm;
            this.wr_data = new();
        end
    endfunction
```

```
    virtual task monitor();
        begin
            @(wrmon_if.wr_cb);
            begin
                wr_data.up_down = wrmon_if.wr_cb.up_down;
                wr_data.load = wrmon_if.wr_cb.load;
                wr_data.data = wrmon_if.wr_cb.din;
                wr_data.display("From Write Monitor");
            end
        end
    endtask

    virtual task start();
        fork
            forever
                begin
                    monitor();
                    data2rm = new wr_data;
                    mon2rm.put(data2rm);
                end
        join_none
    endtask

endclass
```

10.5.4 Read Monitor

```
class count_rd_mon;

    virtual count_if.RD_MON rdmon_if;

    count_trans data2sb, rd_data;

    mailbox #(count_trans)mon2sb;

    function new(virtual count_if.RD_MON rdmon_if,
                 mailbox #(count_trans)mon2sb);
        begin
            this.rdmon_if = rdmon_if;
            this.mon2sb   = mon2sb;
            this.rd_data  = new();
        end
    endfunction
```

```
    virtual task monitor();
        begin
            @(rdmon_if.rd_cb);
            begin
                rd_data.count = rdmon_if.rd_cb.count;
                rd_data.display("From Read Monitor");
            end
        end
    endtask

    virtual task start();
        fork
            forever
                begin
                    monitor();
                    data2sb = new rd_data;
                    mon2sb.put(data2sb);
                end
        join_none
    endtask

endclass
```

10.5.5 Reference Model

```
class count_model;

    count_trans w_data;

    static logic[3:0]ref_count = 0;

    mailbox #(count_trans) wrmon2rm;
    mailbox #(count_trans) rm2sb;

    function new(mailbox #(count_trans) wrmon2rm,
                 mailbox #(count_trans) rm2sb);
        this.wrmon2rm = wrmon2rm;
        this.rm2sb    = rm2sb;
    endfunction
```

```
virtual task count_mod(count_trans model_counter);
    begin
        if(model_counter.load)
            ref_count <= model_counter.w_data;
            wait(model_counter.load == 0)
                begin
                    if(model_counter.up_down == 0)
                        begin
                            if(ref_count > 12)
                                ref_count <= 4'd0;
                            else
                                ref_count <= ref_count + 1'b1;
                        end
                    else if(model_counter.up_down == 1)
                        begin
                            if((ref_count > 10)|| (ref_count < 2))
                                ref_count <=  4'd10;
                            else
                                ref_count <= ref_count - 1'b1;
                        end
                end
        end
    endtask
```

```
    virtual task start();
        fork
            begin
                forever
                    begin
                        wrmon2rm.get(w_data);
                        count_mod(w_data);
                        w_data.count = ref_count;
                        rm2sb.put(w_data);
                    end
            end
        join_none
    endtask

endclass
```

10.5.6 Scoreboard

```
class count_sb;

    count_trans rm_data;
    count_trans sb_data;

    static int ref_data, rm_data, data_verified;

    mailbox #(count_trans) ref2sb;
    mailbox #(count_trans) rdm2sb;

    function new(mailbox #(count_trans) ref2sb,
                 mailbox #(count_trans) rdm2sb);

        this.ref2sb = ref2sb;
        this.rdm2sb = rdm2sb;

    endfunction
```

```
virtual task start();
    fork
        forever
            begin
                ref2sb.get(rm_data);
                ref_data++;

                rdm2sb.get(sb_data);
                rm_data++;

                check(sb_data);
            end
    join_none
endtask
```

```
    virtual task check(count_trans rdata);
       begin
          if(rm_data.count == rdata.count)
             $display("Count Matches");
          else
             $display("Count not matching");
       end
       data_verified++;
       if(data_verified >= no_of_transactions + 2;
          begin
             ->DONE;
          end
    endtask

    virtual function void report();
          $display("............SCOREBOARD REPORT.................");
          $display(" Data_generated = %d",rm_data);
          $display(" Data_verified = %d",data_verified);
          $display("..........................................");
    endfunction

endclass
```

10.6 Environment

```
class count_env;

    virtual count_if.DRV dr_if;
    virtual count_if.WR_MON wrmon_if;
    virtual count_if.RD_MON rdmon_if;

    mailbox #(count_trans)gen2dr  = new();
    mailbox #(count_trans)rm2sb   = new();
    mailbox #(count_trans)mon2sb  = new();
    mailbox #(count_trans)mon2rm  = new();

    count_gen gen_h;
    count_wr_mon wrmon_h;
    count_driver dri_h;
    count_rd_mon rdmon_h;
    count_sb sb_h;
    count_model mod_h;
```

```
    function new(virtual count_if.DRV dr_if,
                 virtual count_if.WR_MON wrmon_if,
                 virtual count_if.RD_MON rdmon_if);

          this.dr_if = dr_if;
          this.wrmon_if = wrmon_if;
          this.rdmon_if = rdmon_if;

    endfunction
```

```
    virtual task build();
        gen_h    = new(gen2dr);
        dri_h    = new(dr_if,gen2dr);
        wrmon_h  = new(wrmon_if,mon2rm);
        rdmon_h  = new(rdmon_if,mon2sb);
        mod_h    = new(mon2rm,rm2sb);
        sb_h     = new(rm2sb,mon2sb);
    endtask


    virtual task reset_duv();
        @(dr_if.dr_cb);
            dr_if.dr_cb.resetn <= 1'b0;
            repeat(2)
                @(dr_if.dr_cb);
            dr_if.dr_cb.resetn <= 1'b1;
    endtask
```

```
    virtual task start();
        gen_h.start;
        dri_h.start;
        wrmon_h.start;
        rdmon_h.start;
        mod_h.start;
        sb_h.start;
    endtask

    virtual task stop();
        wait(sb_h.DONE.triggered);
    endtask

    virtual task run();
        reset_duv();
        start();
        stop();
        sb_h.report();
    endtask

endclass
```

10.7 Package

```
package count_pkg;

    int no_of_transactions = 1;

     `include "count_trans.sv"
     `include "count_gen.sv"
     `include "driver.sv"
     `include "write_mon.sv"
     `include "read_mon.sv"
     `include "count_model.sv"
     `include "count_score.sv"
     `include "count_env.sv"
      `include "count_test.sv"

endpackage
```

10.8 Tetcases

```
class test;

   virtual count_if.DRV dr_if;
   virtual count_if.WR_MON wrmon_if;
   virtual count_if.RD_MON rdmon_if;

   count_env env_h;

   function new(virtual count_if.DRV dr_if,
                virtual count_if.WR_MON wrmon_if,
                virtual count_if.RD_MON rdmon_if);
          this.dr_if    = dr_if;
          this.wrmon_if = wrmon_if;
          this.rdmon_if = rdmon_if;

          env_h = new(dr_if,wrmon_if,rdmon_if);

   endfunction
```

```
   virtual task build();
          env_h.build()
   endtask

   virtual task run();
          env_h.run()
   endtask

endclass
```

10.9 Top Module

```
module top();

   //Import count_pkg
    import count_pkg::*;

   reg clock;

   //Instantiate the interface
   count_if DUV_IF(clock);

   // Handle for test
   test t_h;

   // Instantiate DUV
   counter DUV(.clock(clock),
               .din(DUV_IF.din),
               .load(DUV_IF.load),
               .up_down(DUV_IF.up_down),
               .resetn(DUV_IF.resetn),
               .count(DUV_IF.count));
```

```
   initial
      begin
         clock = 1'b0;
         forever
            #10 clock = ~clock;
      end

   initial
      begin
         if($test$plusargs("TEST1"))
            begin
               t_h = new(DUV_IF, DUV_IF, DUV_IF);
               number_of_transactions = 200;
               t_h.build();
               t_h.run();
               $finish;
            end
      end

endmodule
```