Parameterized Class

Centurion University
Shaping Lives...
Empowering Communities...

Allows Generic Class to be Instantiated as Objects of Different Types

```
class stack #(parameter type T = int;);
  local T items[];
  task push( T a ); ... endtask
  task pop( ref T a ); ... endtask
endclass

stack i_s; // default: a stack of int's

stack#(bit[1:10]) bs; // a stack of 10-bit vector

stack#(real) rs; // a stack of real numbers

stack#(Vfour) vs; // stack of classes
```

Centurion University
Shaping Lives...
Empowering Communities...

Virtual Interfaces

- Classes cannot have modules or interfaces, need a specialized mechanism
  - Why?
- Virtual interfaces provide a way to connect dynamic classes to static world of modules
- Virtual interfaces provide a mechanism for separating test programs/BFM models from the actual signals.

Centurion
UNIVERSITY

Virtual Interface Example

```
// interface definition
interface Bus
(input logic clk);
    bit req;
    bit grant;
    logic [7:0] addr;
    logic [7:0] data;
endinterface: Bus

// interface instance
 Bus infc_b (clk);
// dut instance
 dut dut1 (infc_b, clk);
// class instance
BFM mybfm = new (infc_b);
```

```
 class BFM;
   virtual Bus bus;
   Xaction xaction;

   function new (virtual Bus b);
// need to initialize virtual interface
// in constructor
      bus = b;
      xaction = new;
   endfunction

   task req_bus();
     @(posedge bus.clk);
     bus.req <= 1'b1;
     $display("Req = %b @ %0t",
              bus.req, $time);
   endtask: req_bus
endclass: BFM
```

Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

Random Constraints

Centurion University
Shaping Lives...
Empowering Communities...

Simplest randomness

$urandom system tasks

$urandom() is SV, thread stable, deterministic

$urandom returns unsigned 32-bit integers

Procedural call can be inserted wherever needed

Better Randomness

Constraints are built onto the Class system

Random variables use special modifier:

- **rand** –random variable
- **randc** –random cyclic variable

Object is randomized by calling **randomize( )**method

- Automatically available for classes with random variables.

User-definable methods

- **pre_randomize()**
- **post_randomize()**

Centurion University
Shaping Lives...
Empowering Communities...

Constraints

Set of **Boolean** algebraic expressions

Example:

```
constraint a_le_b { a <= b; }

constraint c_eq_10 {c == 10;}

constraint b_in_range { b >= 2 && b <= 8; }

constraint all_gt_0 {a > 0; b > 0; c > 0;}
```

Conflicting constraints

What happens when you impose constraints that conflict in some way?

Example:

```
constraint x_gt_y { x > y; }
constraint y_gt_z { y > z; }
constraint z_gt_x { z > x; }
…
if ( x_inst.randomize() == 0 ) // Solver error
begin
  …
end
```

Constraint operators

Centurion University
Shaping Lives...
Empowering Communities...

Any Verilog boolean expression

i.e. **x < y+b-c*10>>20**

Other constraint operations

set membership **within**

implication **->** or **if…else…**

iterative constraint **foreach**

variable ordering **solve … before**

functions **func_x()**

Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

Implication constraint

Uses one boolean to decide if another constraint must hold

```
(trans_size == SMALL) -> (length < 10)
(trans_size == MED) ->
      (length >= 10 && length <= 100)
(trans_size == LARGE) -> (length > 100)
```

Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

Loop/array constraints

Constrain every element of an array in some fashion, including reference to other elements of array

```
constraint c1 {
   foreach ( A[i] )
      A[i] inside {2, 4, 6, 8, 10};
   }
constraint c2 {
   foreach ( A[k] )
      (k < A.size-1) -> A[k+1] > A[k];
   }
```

Centurion UNIVERSITY
Shaping Lives...
Empowering Communities...

std::randomize()

Procedural invocation of constraint solver

Any variables can be the random variables

“with” block for constraints

Normal .randomize() cannot include variables outside scope of class

Centurion UNIVERSITY
Shaping Lives...
Empowering Communities...

Inline Constraints

Specify inline constraints that are added to constraint set to solve

```
trans.randomize() with
    { x < 100;  z > buzz; };
```

Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

Enable/Disable rand/constraints

rand_mode – method to toggle the “rand” attribute off on a class variable

constraint_mode – method to toggle the application of a named constraint

Layered Constraints

Centurion University
Shaping Lives...
Empowering Communities...

Constraints Inherited via Class Extension

- Just like data and methods, constraints can be inherited or overridden

```
typedef enum{ low, high, other }AddrType;
class MyBusextends Bus;
rand AddrTypetype;
constraint addr_rang {
    ( type == low ) =>addrin { [ 0 : 15] };
    ( type == high ) =>addrin { [128 : 255] }; }
endclass
```

A Full Example

Centurion University
Shaping
Empowe

```
module test_rand;
typedef enum {SM, MED, LRG} trans_len;
class transaction {
  rand bit [19:0] addr;
  rand trans_len len;
  rand bit [3:0] byte_len;
  rand bit wr_or_rd_b;

  constraint c1 { wr_or_rd_b -> len != LRG; }
  constraint c2 { (len == SM)  -> (byte_len <= 4); }
  constraint c3 { (len == MED) -> (byte_len >  4 && byte_len <=
   8);}
  constraint c4 { (len == LRG) -> (byte_len >  8); }
  // Mem above 16-bit is write-only IO devices
  constraint c5 { (!wr_or_rd_b) -> (addr[19:16] != 0); }

  virtual function string toString()
    return sformat("%5H: %s %d bytes", addr, (wr_or_rd_b ? "WR" :
   "RD"), byte_len);
  endfunction : toString
endclass : transaction
<CONTINUED>
```

Centurion
UNIVERSITY

A Full Example

```
transaction trans = new();
intial
  begin : test_body
    int counter = 0;
    repeat (20)
      begin : rand_gen
        if (trans.randomize() == 0)
          begin
            $display("ERROR: Random constraints
   conflict");
            $finish;
          end
      $display("%d: %s", counter++, trans.toString());
       end : rand_gen
  end : test_body

endmodule : test_rand
```