Module: UVM-2
Register Abstraction Layer

RAL

Assignment 02

➢ **Configure Method:**

There are actually 9 arguments passed to a uvm_reg_field's configure method, each one of the argument and its working is briefly explained below:

1. **`uvm_reg parent:`** Specifies the parent register to which this field belongs.
2. **`int unsigned size:`** Defines the size of the field in bits.
3. **`int unsigned lsb_pos:`** Specifies the position of the least significant bit (LSB) of the field within the register. For example, if `lsb_pos = 0`, the field starts at the least significant bit of the register.
4. **`string access:`** Defines the field's access type. Common values include:
    - `"RW"`: Read-Write
    - `"RO"`: Read-Only
    - `"WO"`: Write-Only
    - `"W1"`: Write-One (write a 1 to set, writing 0 has no effect).
5. **`bit volatile:`** Indicates whether the field is volatile.
6. **`uvm_reg_data_t reset:`** This value determines the initial state of the field after a reset. It's useful for modeling reset behavior in registers.
7. **`bit has_reset:`** Indicates whether the field has an explicitly defined reset value.
8. **`bit is_rand:`** Specifies whether the field is randomized during constrained random testing.
9. **`bit individually_accessible:`** Indicates whether the field can be accessed independently of other fields in the register.

➢ **Modeling Memory:**

UVM allows us to model the DUT memory. Such memory is constructed using a class extended from the **uvm_mem** class as following:

```
class mem_mod extends uvm_mem;
```

```
  `uvm_object_utils(mem_mod)

  // Constructor
  function new( string name = "" );
    super.new(.name(name),    // Name of the Memory
              .size(256),     // Depth of the Memory
              .n_bits(32),    // Width of Memory
              .access("RW") // Access Policy
             );
  endfunction : new

endclass
```

➢ **Register Classes:**

Control Register

```
class cntrl extends uvm_reg;
    // Factory Registration
    `uvm_object_utils(cntrl)

    rand uvm_reg_field control;

    // Constructor
    function new ( string name = "cntrl");
        super.new( name, 4, UVM_NO_COVERAGE )
    endfunction : new

    virtual function void build();
        control = uvm_reg_field::type_id::create("control");
        control.configure( this, 4, 0, "RW", 0, 0, 1, 1, 0  );
    endfunction : build

endclass : cntrl
```

Register 1

```
class reg1 extends uvm_reg;
    // Factory Registration
    `uvm_object_utils(reg1)

    rand uvm_reg_field Register1;

    // Constructor
    function new ( string name = "reg1");
```

```
        super.new( name, 32, UVM_NO_COVERAGE )
    endfunction : new

    virtual function void build();
        control = uvm_reg_field::type_id::create("Register1");
        control.configure( this, 32, 0, "RW", 0, 0, 1, 1, 0  );
    endfunction : build

endclass : reg1
```

Register 2

```
class reg2 extends uvm_reg;
    // Factory Registration
    `uvm_object_utils(reg2)

    rand uvm_reg_field Register2;

    // Constructor
    function new ( string name = "reg2");
        super.new( name, 32, UVM_NO_COVERAGE )
    endfunction : new

    virtual function void build();
        control = uvm_reg_field::type_id::create("Register2");
        control.configure( this, 32, 0, "RW", 0, 0, 1, 1, 0  );
    endfunction : build

endclass : reg2
```

Register 3

```
class reg3 extends uvm_reg;
    // Factory Registration
    `uvm_object_utils(reg3)

    rand uvm_reg_field Register3;

    // Constructor
    function new ( string name = "reg3");
        super.new( name, 32, UVM_NO_COVERAGE )
    endfunction : new

    virtual function void build();
        control = uvm_reg_field::type_id::create("Register3");
        control.configure( this, 32, 0, "RW", 0, 0, 1, 1, 0  );
```

```
    endfunction : build

endclass : reg3
```

Register 4

```
class reg4 extends uvm_reg;
    // Factory Registration
    `uvm_object_utils(reg4)

    rand uvm_reg_field Register4;

    // Constructor
    function new ( string name = "reg4");
        super.new( name, 32, UVM_NO_COVERAGE )
    endfunction : new

    virtual function void build();
        control = uvm_reg_field::type_id::create("Register4");
        control.configure( this, 32, 0, "RW", 0, 0, 1, 1, 0  );
    endfunction : build

endclass : reg4
```