Module: UVM-2
Register Abstraction Layer

RAL

Assignment 04

➢ **Write( ) method:**

```
virtual task write(
output      uvm_status_e      status,
input uvm_reg_data_t    value,
input uvm_path_e  path   =    UVM_DEFAULT_PATH,
input uvm_reg_map map    =    null,
input uvm_sequence_base parent =    null,
input int   prior  =    -1,
input uvm_object  extension    =    null,
input string      fname  =    "",
input int   lineno =    0);
```

The **write()** method in UVM is used to write a specific value to a register or memory in the DUT through the UVM Register Abstraction Layer (RAL). It can perform both front-door access (via an address map) and back-door access (directly to DUT).

Here is a detailed explanation of the purpose of each argument in the write() method:

➔ Arguments in the `write()` Method

1. **`output uvm_status_e status:`** This is an output variable that returns the status of the `write()` operation.
2. **`input uvm_reg_data_t value:`** This is the value that you want to write to the register or memory.
3. **`input uvm_path_e path = UVM_DEFAULT_PATH:`** Specifies the access path to be used for the write operation.
4. **`input uvm_reg_map map = null:`** Specifies the address map to be used for the write operation.
5. **`input uvm_sequence_base parent = null:`** Identifies the parent sequence initiating the write operation.

6. **`input int prior = -1:`** Specifies the priority of the write operation. If set to −1 (default), the default priority for the sequence is used.
7. **`input uvm_object extension = null:`** Allows passing additional user-defined data during the write operation.
8. **`input string fname = "":`** Indicates the filename where the `write()` call is made.
9. **`input int lineno = 0:`** Indicates the line number in the file where the `write()` call is made.

➢ **Register Write Sequences:**

cntrl_wr_sequence

```
class cntrl_wr_sequence extends uvm_sequence;

// Factory Registration
`uvm_object_utils(cntrl_wr_sequence)

// Register Model
reg_blk reg_model;

// Constructor
function new( string name = "cntrl_wr_sequence");
    super.new(name);
endfunction : new

// Body Task
task body();
    uvm_status_e status;
    bit [31:0] wdata;
    wdata = $urandom();
    reg_model.ctrl.write(status, wdata);
endtask

endclass : cntrl_wr_sequence
```

reg1_wr_sequence

```
   class reg1_wr_sequence extends uvm_sequence;

// Factory Registration
`uvm_object_utils(reg1_wr_sequence)

// Register Model
reg_blk reg_model;

// Constructor
```

```
function new( string name = "reg1_wr_sequence");
    super.new(name);
endfunction : new

// Body Task
task body();
    uvm_status_e status;
    bit [31:0] wdata;
    wdata = $urandom();
    reg_model.r1.write(status, wdata);
endtask

endclass : reg1_wr_sequence
```

reg2_wr_sequence

```
class reg2_wr_sequence extends uvm_sequence;

// Factory Registration
`uvm_object_utils(reg2_wr_sequence)

// Register Model
reg_blk reg_model;

// Constructor
function new( string name = "reg2_wr_sequence");
    super.new(name);
endfunction : new

// Body Task
task body();
    uvm_status_e status;
    bit [31:0] wdata;
    wdata = $urandom();
    reg_model.r2.write(status, wdata);
endtask

endclass : reg2_wr_sequence
```

reg3_wr_sequence

```
    class reg3_wr_sequence extends uvm_sequence;

// Factory Registration
`uvm_object_utils(reg3_wr_sequence)

// Register Model
reg_blk reg_model;

// Constructor
function new( string name = "reg3_wr_sequence");
    super.new(name);
endfunction : new

// Body Task
```

```
task body();
    uvm_status_e status;
    bit [31:0] wdata;
    wdata = $urandom();
    reg_model.r3.write(status, wdata);
endtask

endclass : reg3_wr_sequence
```

reg4_wr_sequence

```
class reg4_wr_sequence extends uvm_sequence;

// Factory Registration
`uvm_object_utils(reg4_wr_sequence)

// Register Model
reg_blk reg_model;

// Constructor
function new( string name = "reg4_wr_sequence");
    super.new(name);
endfunction : new

// Body Task
task body();
    uvm_status_e status;
    bit [31:0] wdata;
    wdata = $urandom();
    reg_model.r4.write(status, wdata);
endtask

endclass : reg4_wr_sequence
```