DAY 82 - 100 DAYS VERIFICATION CHALLENGE

Topic: UVM classes & UVM Factory

DAY 82 CHALLENGE:

1. Which are the three main types of classes in UVM?
2. Explain following classes in uvm:
   i. uvm_transaction
   ii. uvm_sequence
   iii. uvm_sequence_item
   iv. uvm_component
   v. uvm_object
3. What is UVM Factory? What is the need of UVM Factory?
4. Why do we need to register a class with uvm_factory?
5. What is the advantage of `uvm_component_utils() and `uvm_object_utils() ?
6. What is the difference between uvm_transaction and uvm_seq_item?
7. What is the difference between new() and create?
8. Explain the following coding conventions for UVM Factory:
   i. Registration
   ii. Constructor Defaults
   iii. Component and Object Creation

Astitvyogi-uvm

DAY : 32

Topic : UVM classes & UVM Factory

Sol^n ① which are the three main types of classes in UVM?

- uvm provides three base classes to implement message reporting
  ① uvm report-object ② uvm report handler ③ uvm report server

① uvm_report_object :-
  ∴ All the reporting methods are implemented in this class.
  • interface to uvm reporting facility, through which components issue various message during simulation.

② uvm_report_handler :-
  ∴ Store the reporting configuration, determines whether issued massage should be printed or not.
  • decision making component, it has reference values, which checks with messag values, decides message print or not.

③ uvm_report_server :
  does actually formatting and printing of messages.

Report ID : ID string, severity, verbosity, textual message.

Sol^n② Explain following classes in uvm :

- uvm_transaction :

  represent data item that are processed by the verification environment. It contains the packet of information in form of variables to hold value to send to the DUT or receive from the DUT. It also has several method to copy, compare & print transactions.

⊙ uvm_sequence :

A uvm sequence is a group of abstract transactions that sequencer grab and puts them on driver which converts these abstract transaction to pin wiggle to drive stimulus to the DUT.

- The Task body () inside the user defined sequence is automatically called when sequence_item.start () called from uvm_Test.

⊙ uvm_sequence_item: Sequence-item consist of data field required for generating the stimulus. in order to generate the stimulus. ~~it's derived from uvm_object~~

→ The sequence item are randomized in sequences.

⊙ uvm_component: All those components which stay in a verification enviorment for a entire simulation duration are called dynamic components

eg : driver, sequences, monitor, scoreboard

- standard constructor:

```
function new (input string inst = "comp",
              uvm_component parent );
    sup.new (inst, parent);

endfunction
```

⊙ uvm_object: All those component which do not stay in a verification Enviroment for an entire simulation duration are called dynamic components.

eg : transaction class

standard constrctor

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

Sol

Sol^n ③ what is uvm factory ? what is the need of Uvm Factory?

- uvm factory is a class that will create the Components and object based on the type-id & type-name. But typeid is preferred one.
- factory is a global database of all TB Component and object definitions.
- uvm factory allows to override the class without editing or recompiling the code.
- uvm_component_utills or uvm_object_utils add the class defination to the factory. This defination can be accessed using type_id. factory registration automatically create type id.

o need of uvm factory:

- in system verilog we create the object if transaction class is requred classes.
- if we want to add methods in transaction class we have to extende the existing trasaction class with the new transaction class & we override the tx class present in generator with new transaction class from test. so to avoid that factory is used.
- If we use Constructor new for override the parent, its difficult but by using the factory its easy to extends the class using Type_id.

Sol^{n}④ Why do we need to register a class with uvm_factory?

- since all the defination are registered to the factory they can created by reffering to the factory defination.

- factory registered defination Can be overridden from anywhere in Testbench.

'uvm_Component_Utils (agent)

↑ agent class is registerd to the factory

Sol^n ⑤ What is the advantage of 'uvm_component_utils() & 'uvm ~~component~~/object_utils()?

- Component and objects registers to the factory 'uvm_component_utils() 'uvm_object_utils()
- using the 'uvm_component_utils() macro, the class is automatically registered with the uvm factory and can be dynamically created and configured at run-time.
- 'uvm_object_utils() is used to register a class as a uvm_object. which is generic container for data used in a uvm testbench.

Sol^n ⑥ What is the diffrence Between uvm transaction & uvm seq item?

uvm_trasaction	uvm seq_item
• uvm transaction base classes used to define transactions.	• uvm seq_item are base classes used to define sequence items.
⊙ Inheritance: • um trasaction inherits from uvm_object.	• while uvm_seq_item inherits from uvm sequence item.
• This means uvm_trasaction can be used indopendentlly of any sequnce	• while uvm_sequence item is typically used within a sequence
⊙ Time of Creation: • uvm_trasaction is typically created and managed by a driver & mon.	while uvm_seq_item is created & managed by a sequences.
⊙ Reuse: • uvm_trasaction is designed to be reusable across multiples sequences.	while uvm_seq_item is typically specific to a single sequence.
• Data field: • uvm transaction includes several built-in data field for modeling common transaction property such as address, data & response.	• in uvm_seq_item does not include any built-in data fields & must be customized for each specific sequence item.

Sol^n ⑦ what is the difference between new() and create?

new()	create()
• new() is methods used to create objects dynamically at runtime.	• create() is are also methods create objects dynamically runtime
① Return type: → new() method returns an object refference, while	create() method returns a handle to the object
② memory allocation: →new() method allocates memory for the object on the heap runtime. → new() does not intilize the object	while create() method does not allocate memory. instead create() method requres an object to be passed as an argument & its initilize the object.

eg: // new() method
class_name object = new();
Object. some_method()
	eg: // using create() method
	object = class_name :: type_id :: create();

Sol^n ⑧ Explain the following coding Convention for uvm factory:

① Registration: in uvm_registering a class with the uvm factory is impoatant coz its enables the factory to create & manage instance of the registered class

• To register a class with the uvm factory, you need to use:
'uvm_component_utils().
OR
'uvm_component_utils (sequnce)

⑪ Constructor default: when you register a class with the factor you might want to set default values for certain parameters

• Thes defaults are appled when an object is created. function new() is the default Constructor.

⑪ Component and object creation: The create() method of the wrapper class is used to create objects for the uvm_object & uvm Component class.

• The build prose is used to create Component instances & build Component hierachy.