Sequence-Related Macros

Summary

Sequence Action Macros

These macros are used to start sequences and sequence items on the default sequencer, *m_sequencer*. This is determined a number of ways.

- the sequencer handle provided in the uvm_sequence_base::start method
- the sequencer used by the parent sequence
- the sequencer that was set using the uvm_sequence_item::set_sequencer method

\`uvm_create

```
`uvm_create(SEQ_OR_ITEM)
```

This action creates the item or sequence using the factory. It intentionally does zero processing. After this action completes, the user can manually set values, manipulate rand_mode and constraint_mode, etc.

`uvm_do

```
`uvm_do(SEQ_OR_ITEM)
```

This macro takes as an argument a uvm_sequence_item variable or object. The argument is created using `uvm_create if necessary, then randomized. In the case of an item, it is randomized after the call to uvm_sequence_base::start_item() returns. This is called late-randomization. In the case of a sequence, the sub-sequence is started using uvm_sequence_base::start() with *call_pre_post* set to 0. In the case of an item, the item is sent to the driver through the associated sequencer.

For a sequence item, the following are called, in order

```
`uvm_create(item)
sequencer.wait_for_grant(prior)  (task)
this.pre_do(1)                   (task)
item.randomize()
this.mid_do(item)                (func)
sequencer.send_request(item)     (func)
sequencer.wait_for_item_done()   (task)
this.post_do(item)               (func)
```

For a sequence, the following are called, in order

```
`uvm_create(sub_seq)
sub_seq.randomize()
sub_seq.pre_start()          (task)
this.pre_do(0)               (task)
this.mid_do(sub_seq)         (func)
sub_seq.body()               (task)
this.post_do(sub_seq)        (func)
sub_seq.post_start()         (task)
```

`uvm_do_pri

```
`uvm_do_pri(SEQ_OR_ITEM, PRIORITY)
```

This is the same as `uvm_do except that the sequene item or sequence is executed with the priority specified in the argument

`uvm_do_with

```
`uvm_do_with(SEQ_OR_ITEM, CONSTRAINTS)
```

This is the same as `uvm_do except that the constraint block in the 2nd argument is applied to the item or sequence in a randomize with statement before execution.

`uvm_do_pri_with

```
`uvm_do_pri_with(SEQ_OR_ITEM, PRIORITY, CONSTRAINTS)
```

This is the same as `uvm_do_pri except that the given constraint block is applied to the item or sequence in a randomize with statement before execution.

Sequence on Sequencer Action Macros

These macros are used to start sequences and sequence items on a specific sequencer. The sequence or item is created and executed on the given sequencer.

`uvm_create_on

```
`uvm_create_on(SEQ_OR_ITEM, SEQR)
```

This is the same as `uvm_create except that it also sets the parent sequence to the sequence in which the macro is invoked, and it sets the sequencer to the specified *SEQR* argument.

`uvm_do_on

```
`uvm_do_on(SEQ_OR_ITEM, SEQR)
```

This is the same as `uvm_do except that it also sets the parent sequence to the sequence in which the macro is invoked, and it sets the sequencer to the specified *SEQR* argument.

`uvm_do_on_pri

```
`uvm_do_on_pri(SEQ_OR_ITEM, SEQR, PRIORITY)
```

This is the same as `uvm_do_pri except that it also sets the parent sequence to the sequence in which the macro is invoked, and it sets the sequencer to the specified *SEQR* argument.

`uvm_do_on_with

```
`uvm_do_on_with(SEQ_OR_ITEM, SEQR, CONSTRAINTS)
```

This is the same as `uvm_do_with except that it also sets the parent sequence to the sequence in which the macro is invoked, and it sets the sequencer to the specified *SEQR* argument. The user must supply brackets around the constraints.

`uvm_do_on_pri_with

```
`uvm_do_on_pri_with(SEQ_OR_ITEM, SEQR, PRIORITY, CONSTRAINTS)
```

This is the same as `uvm_do_pri_with except that it also sets the parent sequence to the sequence in which the macro is invoked, and it sets the sequencer to the specified *SEQR* argument.

Sequence Action Macros for Pre-Existing Sequences

These macros are used to start sequences and sequence items that do not need to be created.

`uvm_send

```
`uvm_send(SEQ_OR_ITEM)
```

This macro processes the item or sequence that has been created using `uvm_create. The processing is done without randomization.  Essentially, an `uvm_do without the create or randomization.

`uvm_send_pri

```
`uvm_send_pri(SEQ_OR_ITEM, PRIORITY)
```

This is the same as `uvm_send except that the sequene item or sequence is executed with the priority specified in the argument.

`uvm_rand_send

```
`uvm_rand_send(SEQ_OR_ITEM)
```

This macro processes the item or sequence that has been already been allocated (possibly with `uvm_create).  The processing is done with randomization.  Essentially, an `uvm_do without the create.

`uvm_rand_send_pri

```
`uvm_rand_send_pri(SEQ_OR_ITEM, PRIORITY)
```

This is the same as `uvm_rand_send except that the sequene item or sequence is executed with the priority specified in the argument.

`uvm_rand_send_with

```
`uvm_rand_send_with(SEQ_OR_ITEM, CONSTRAINTS)
```

This is the same as `uvm_rand_send except that the given constraint block is applied to the item or sequence in a randomize with statement before execution.

`uvm_rand_send_pri_with

```
`uvm_rand_send_pri_with(SEQ_OR_ITEM, PRIORITY, CONSTRAINTS)
```

This is the same as `uvm_rand_send_pri except that the given constraint block is applied to the item or sequence in a randomize with statement before execution.

Sequencer Subtypes

`uvm_declare_p_sequencer

This macro is used to declare a variable *p_sequencer* whose type is specified by *SEQUENCER*.

```
`uvm_declare_p_sequencer(SEQUENCER)
```

The example below shows using the the `uvm_declare_p_sequencer macro along with the uvm_object_utils macros to set up the sequence but not register the sequence in the sequencer’s library.

```
class mysequence extends uvm_sequence#(mydata);
  `uvm_object_utils(mysequence)
  `uvm_declare_p_sequencer(some_seqr_type)
  task body;
    //Access some variable in the user's custom sequencer
    if(p_sequencer.some_variable) begin
      ...
    end
  endtask
endclass
```