Go2UVM - UVM tit-bits

- Go2UVM.org – the premier UVM support site clarifies common UVM queries from users
- Read along to learn this tech-tip, contact us if you want to experiment these and many more hands-on UVM via:

training@cvcblr.com

+91-9620209223

http://www.fb.com/cvc.uvm

https://www.linkedin.com/company/cvc-pvt-ltd

http://www.twitter.com/cvcblr

UVM - m_sequencer vs. p_sequencer

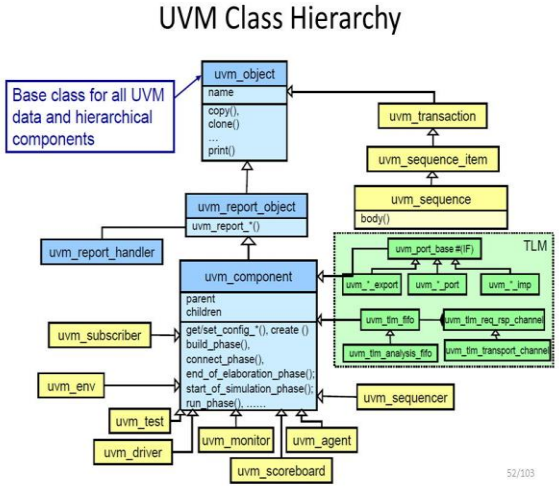
- During One of our international UVM training classes held at Vietnam, a customer asked about:
  - *We still confused about the usage of p_sequencer and m_sequencer*

m_sequencer

vs.

p_sequencer

UVM Sequence & SQR



- UVM has 2 key OOP inheritance trees
  - Transaction
  - Component
- UVM also provides "bridge" bet'n the these 2 trees
  - ***m_sequencer***

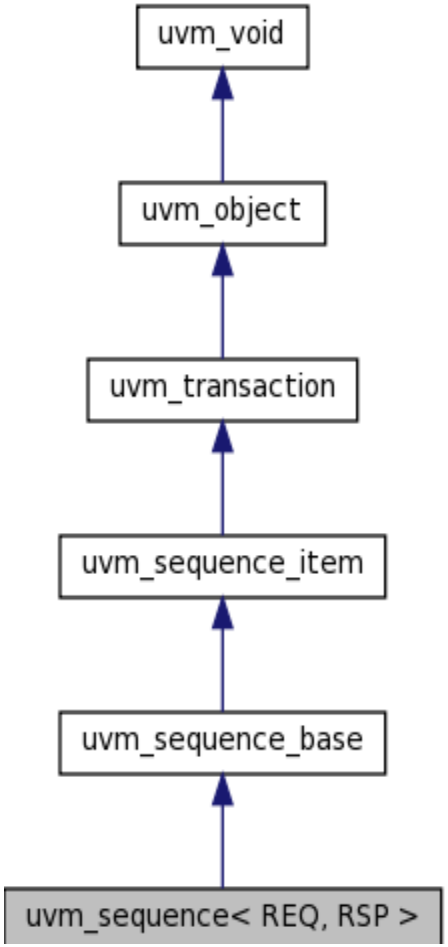
```
class uvm_sequence_item extends uvm_transaction;

  local      int                 m_sequence_id = -1;
  protected  bit                 m_use_sequence_info;
  protected  int                 m_depth = -1;
  protected  uvm_sequencer_base  m_sequencer;
  protected  uvm_sequence_base   m_parent_sequence;
  static     bit issued1,issued2;
  bit        print_sequence_info;
```

uvm_sequence_item.svh

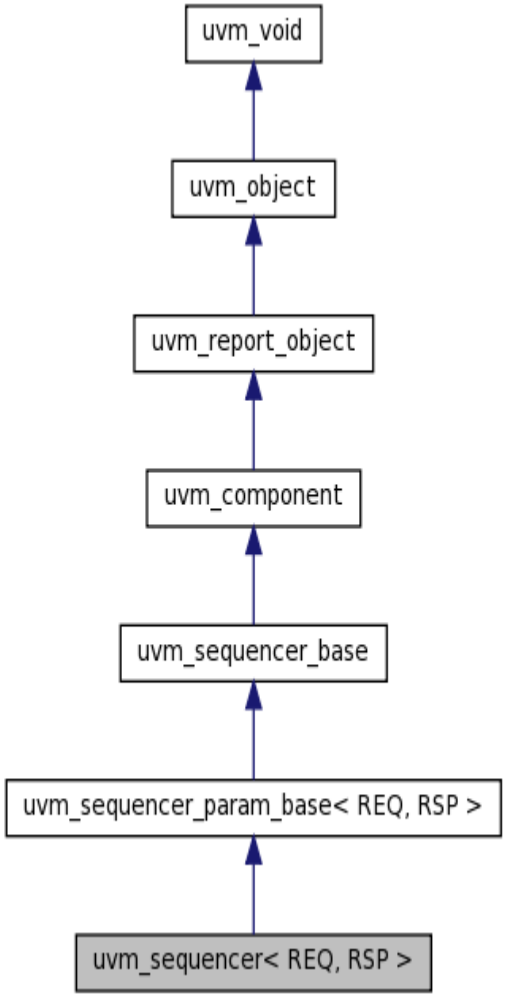
UVM m_sequencer

- This pointer *m_sequencer* is to be set before the sequence body can be executed
- Usually it is done via series of method calls internally.
- User code: ***uvm_test::main_phase() → seq.start (sqr_0)***
- Internally it calls ***seq.start(sqr_0)*** (Coded in *uvm_sequence_base* class)
  - start(**sqr_0**) -> set_item_context(**sqr_0**) -> set_sequencer(**sqr_0**)
  - // m_sequencer = **sqr_0**;



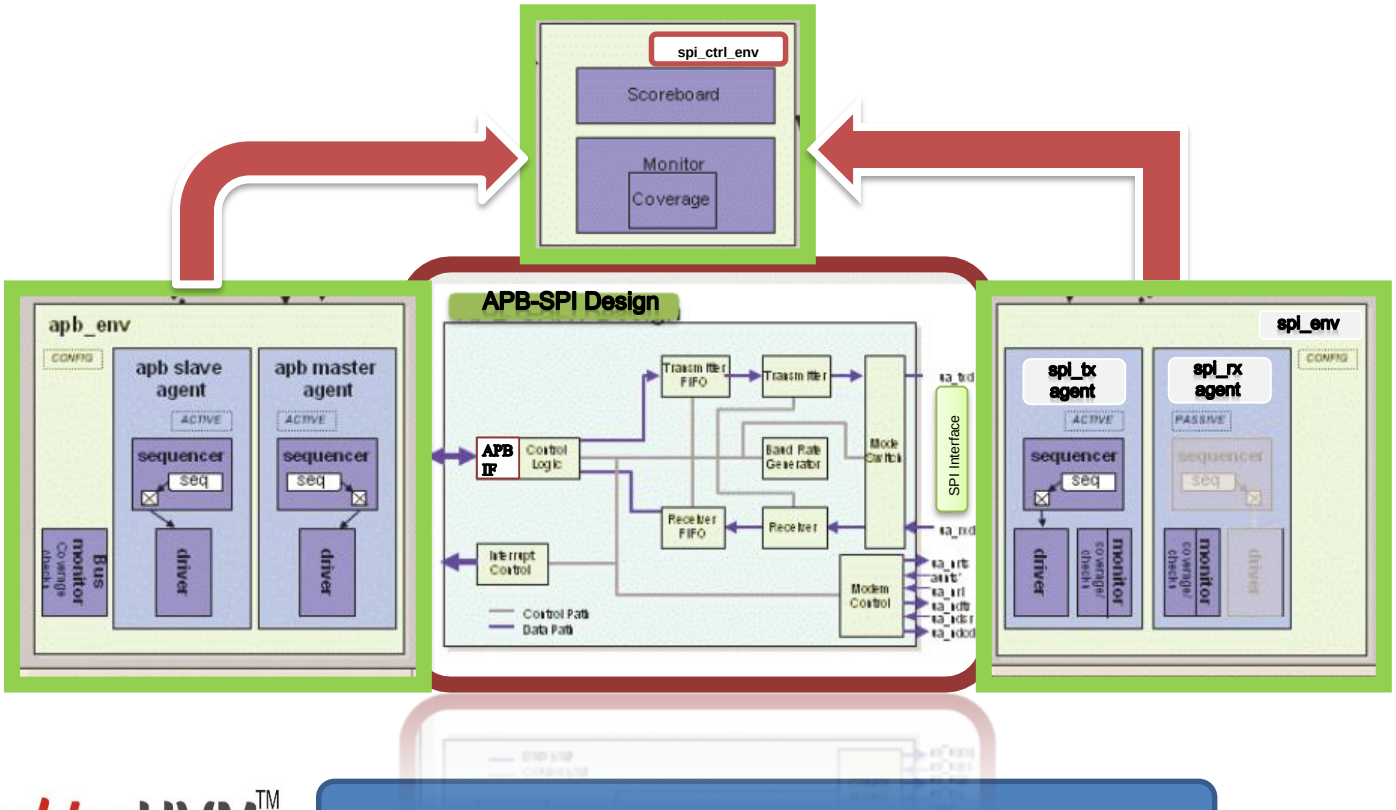
UVM m_sequencer

- Typical applications of *m_sequencer*:
  - Change arbitration mode of a sequencer (via ***m_sequencer.****set_arbitration()*)
  - Accessing component hierarchy from within SEQ body
- While built-in ***m_sequencer*** variable is handy, it is of type ***uvm_sequencer_base***
  - Hence no access to derived SQR members/variables
  - Can't be used to access VSQR members/variables



UVM Virtual Sequencer

- Typical IP verification requires more than one sequencer/interface
- UVM uses a term “virtual sequencer” to denote a wrapper around multiple sequencers



UVM Virtual Sequencer

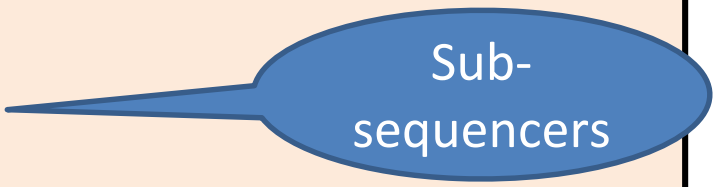
```
class vl_apb_spi_vsqr extends uvm_virtual_sequencer;

  apb_sqr apb_sqr_0;
  spi_sqr spi_sqr_0;

  function new (string name = " vl_apb_spi_vsqr",
                              uvm_component parent);
    super.new(name,parent);

  endfunction:new

  `uvm_component_utils(vl_apb_spi_vsqr)
endclass : vl_apb_spi_vsqr
```

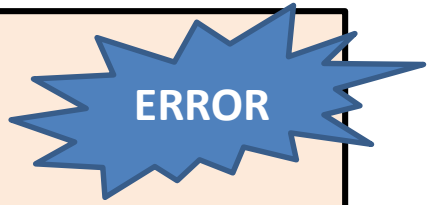


Why not m_sequencer for Virtual SEQ?

- A virtual sequence co-ordinates multiple sequences on multiple sequencers

```
class vl_apb_spi_vseq extends uvm_sequence;
 `uvm_object_utils(vl_apb_spi_vseq)

 task body();
  apb_seq.start(m_sequencer.apb_sqr_0);  // Run APB on apb_sqr_0
  spi_seq.start(m_sequencer.spi_sqr_0);    // Run SPI   on spi_sqr_0
```



m_sequencer* is of type *uvm_sequencer_base

Hence can NOT access user defined sub-sequencers ☹

```
class uvm_sequence_item extends uvm_transaction;

  local      int                  m_sequence_id = -1;
  protected  bit                  m_use_sequence_info;
  protected  int                  m_depth = -1;
  protected  uvm_sequencer_base   m_sequencer;
  protected  uvm_sequence_base    m_parent_sequence;
```

UVM Virtual SEQ & p_sequencer

- A virtual sequence co-ordinates multiple sequences on multiple sequencers
- UVM provides a ***p_sequencer*** option

```
`define uvm_declare_p_sequencer(SEQUENCER) \
 SEQUENCER p_sequencer;\
 virtual function void m_set_p_sequencer();\
   super.m_set_p_sequencer(); \
   if( !$cast(p_sequencer, m_sequencer)) \
```

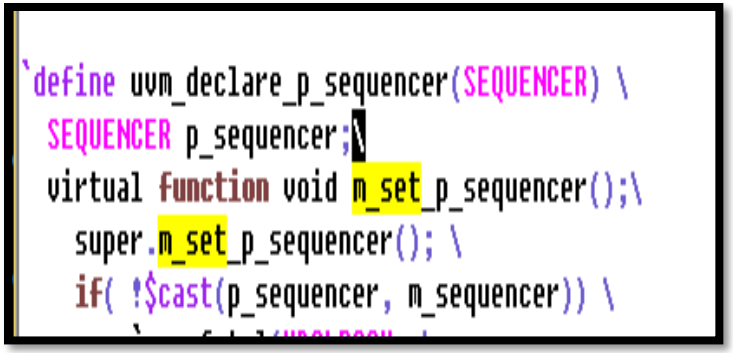
```
class vl_apb_spi_vseq extends uvm_sequence;

 `uvm_object_utils(vl_apb_spi_vseq)
 // Set the p_sequencer and other book keeping
stuff
 `uvm_declare_p_sequencer(vl_apb_spi_vsqr)
```

Use p_sequencer for Virtual SEQ

- A virtual sequence co-ordinates multiple sequences on multiple sequencers

```
class vl_apb_spi_vseq extends uvm_sequence;
 `uvm_object_utils(vl_apb_spi_vseq)
 // Set the p_sequencer and other book keeping stuff
 `uvm_declare_p_sequencer(vl_apb_spi_vsqr)
 task body();
   apb_seq.start(p_sequencer.apb_sqr_0);  // Run APB on apb_sqr_0
   spi_seq.start(p_sequencer.spi_sqr_0);    // Run SPI   on spi_sqr_0
```



```
`define uvm_declare_p_sequencer(SEQUENCER) \
 SEQUENCER p_sequencer;\
 virtual function void m_set_p_sequencer();\
   super.m_set_p_sequencer(); \
   if( !$cast(p_sequencer, m_sequencer)) \
```

Summary

- Go2UVM.org – the premier UVM support site clarifies common UVM queries from users
- Contact us if you want to experiment these and many more hands-on UVM via:

training@cvcblr.com

+91-9620209223

http://www.fb.com/cvc.uvm

https://www.linkedin.com/company/cvc-pvt-ltd

http://www.twitter.com/cvcblr