VERIFICATION ACADEMY

Advanced OVM (& UVM)

Understanding TLM

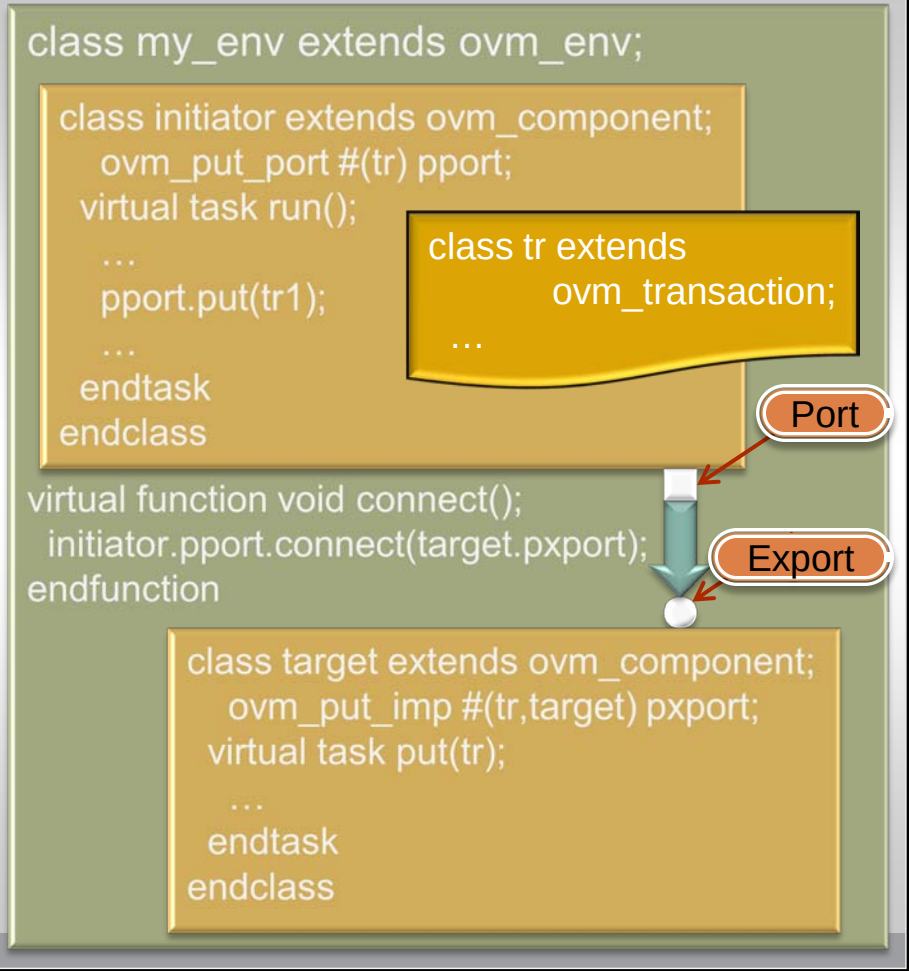
Tom Fitzpatrick
Verification Technologist

academy@mentor.com
www.verificationacademy.com

Mentor Graphics®

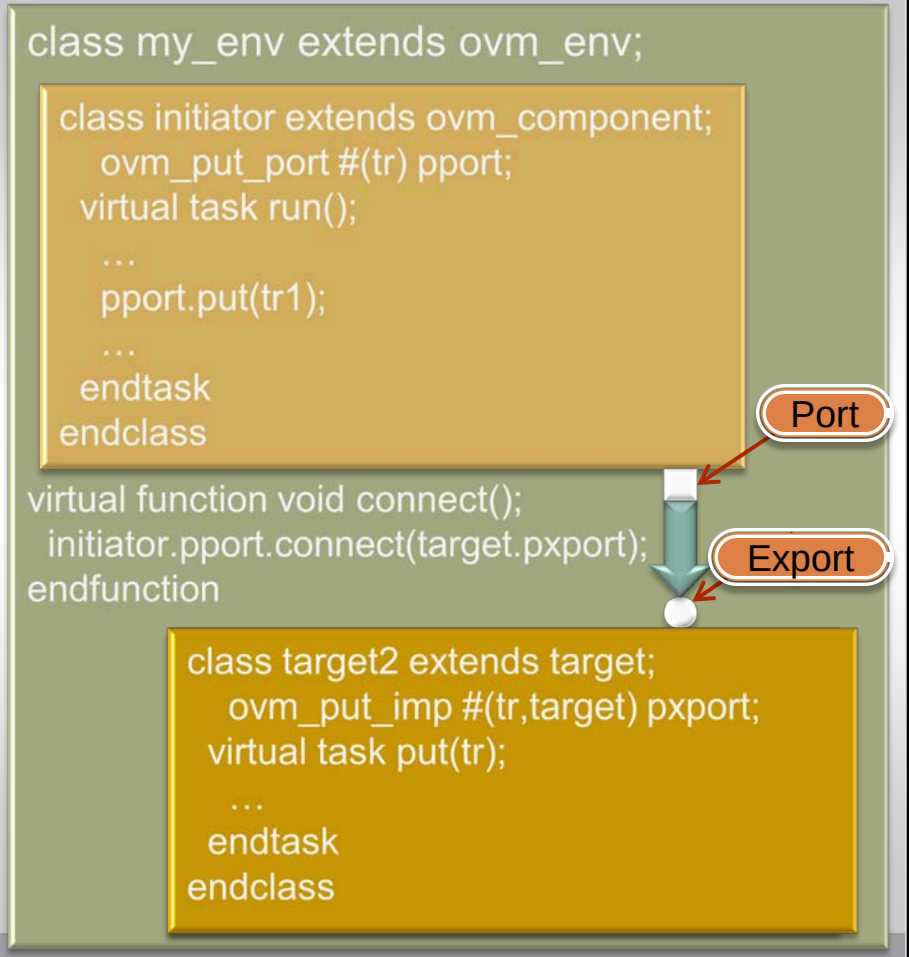
How TLM Works

- **TLM is all about *communication* through *method calls***
  - A TLM *port* specifies the "API" to be used
  - A TLM *export* supplies the implementation of the methods
- **Connections are between ports/exports, not components**
- **Transactions are *objects***
- **Ports & exports are parameterized by the transaction type being communicated**



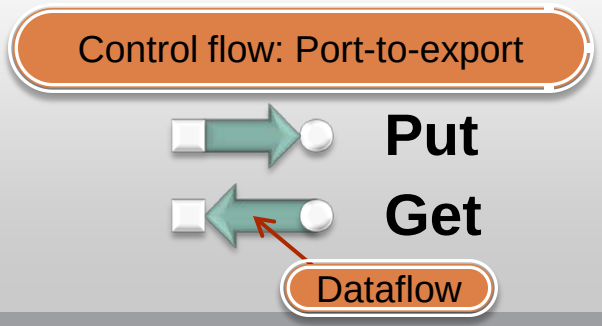
How TLM Works

- **TLM is all about *communication* through *method calls***
  - A TLM *port* specifies the “API” to be used
  - A TLM *export* supplies the implementation of the methods
- **Connections are between ports/exports, not components**
- **Transactions are *objects***
- **Ports & exports are parameterized by the transaction type being communicated**



TLM Interfaces Specify Directionality & Completion

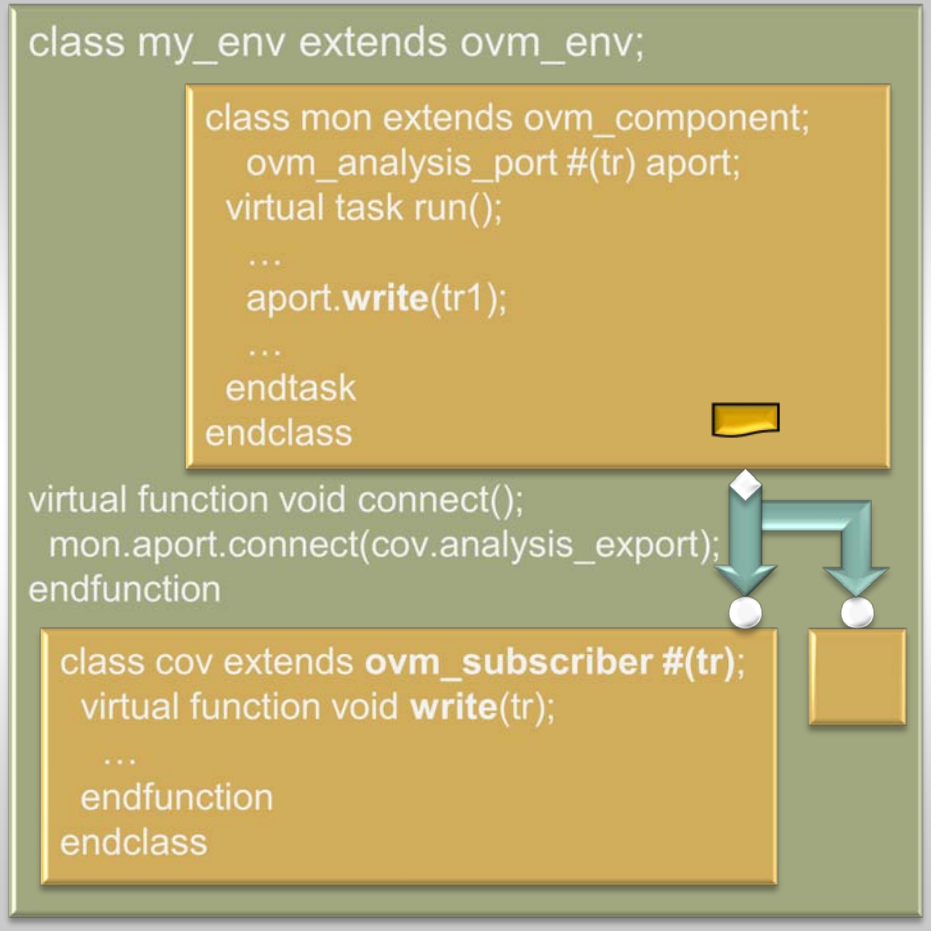
- **Unidirectional**
  - Blocking (tasks)
    - Put
    - Get/peek
  - Nonblocking (functions)
    - try_put
    - try_get/peek
    - write



- **Bidirectional**
  - Master
    - put_request
    - get_response
  - Slave
    - get_request
    - put_response
  - Transport
    - transport
    - nb_transport (function)
  - Sequence/Driver
    - get_request
    - put_response

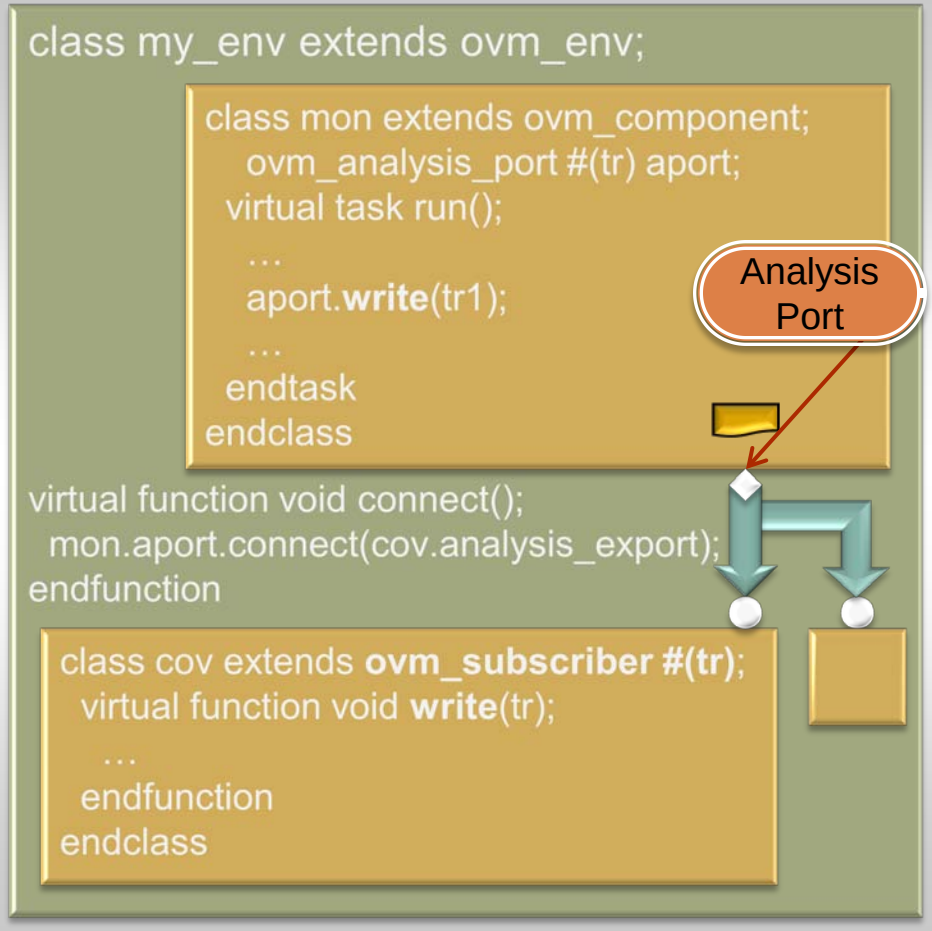
Analysis Communication

- **Analysis ports support 1:many connections**
  - All write() functions called in zero time
- **Used by coverage collectors and scoreboards**
  - ovm_subscriber has built-in analysis_export



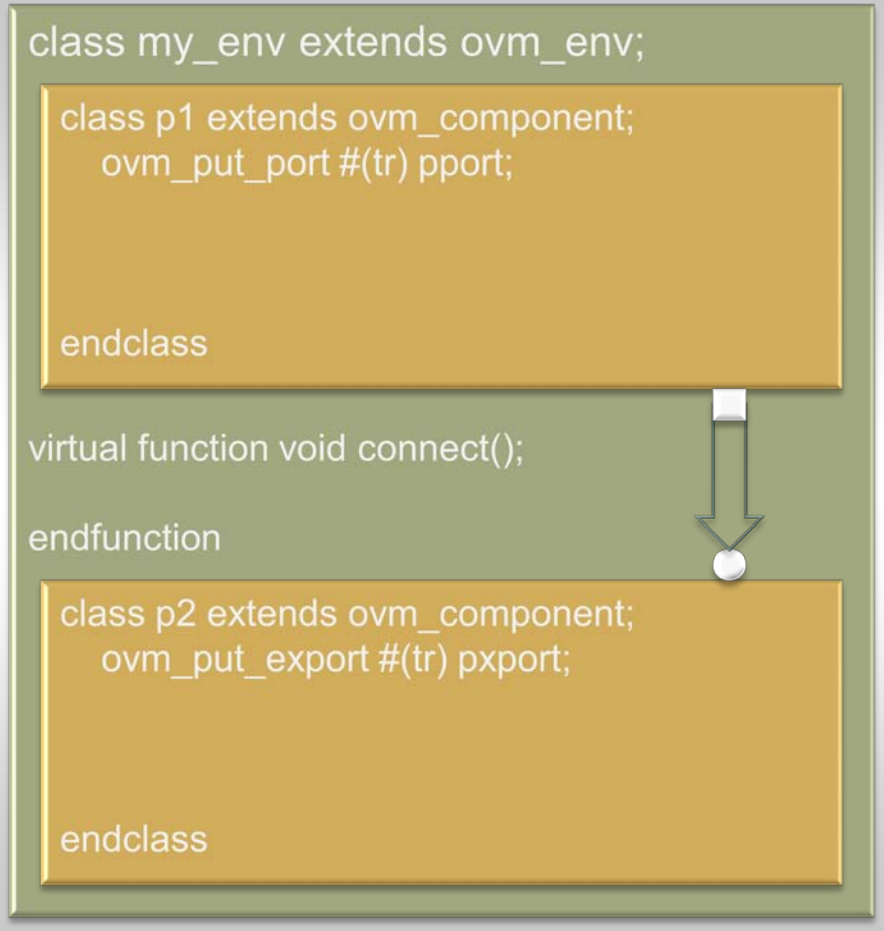
Analysis Communication

- **Analysis ports support 1:many connections**
  - All write() functions called in zero time
- **Used by coverage collectors and scoreboards**
  - ovm_subscriber has built-in analysis_export



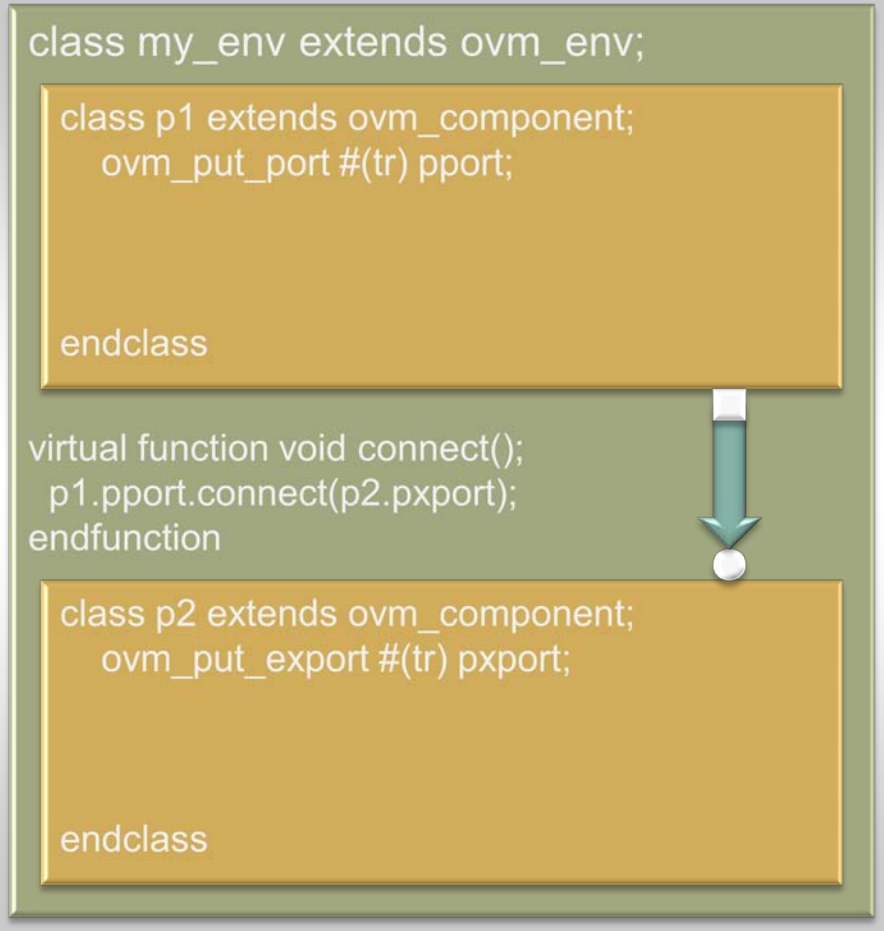
Hierarchical Connections

- **Port-to-Export**



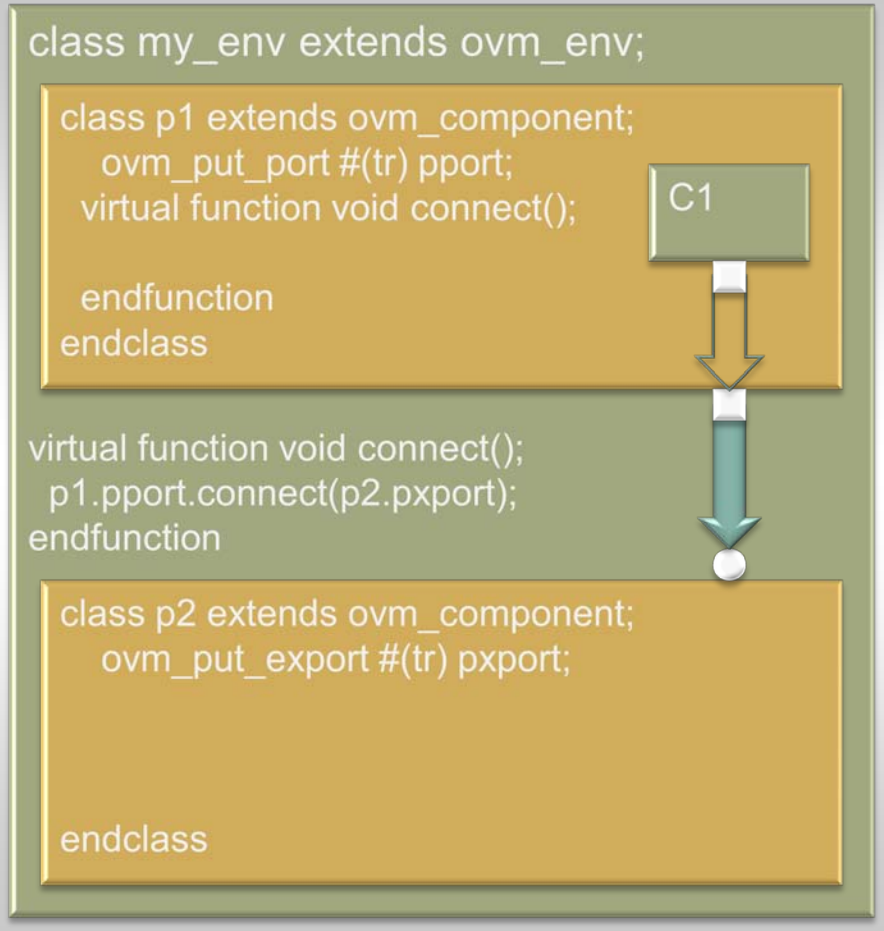
Hierarchical Connections

- **Port-to-Export**
  port.connect(export);



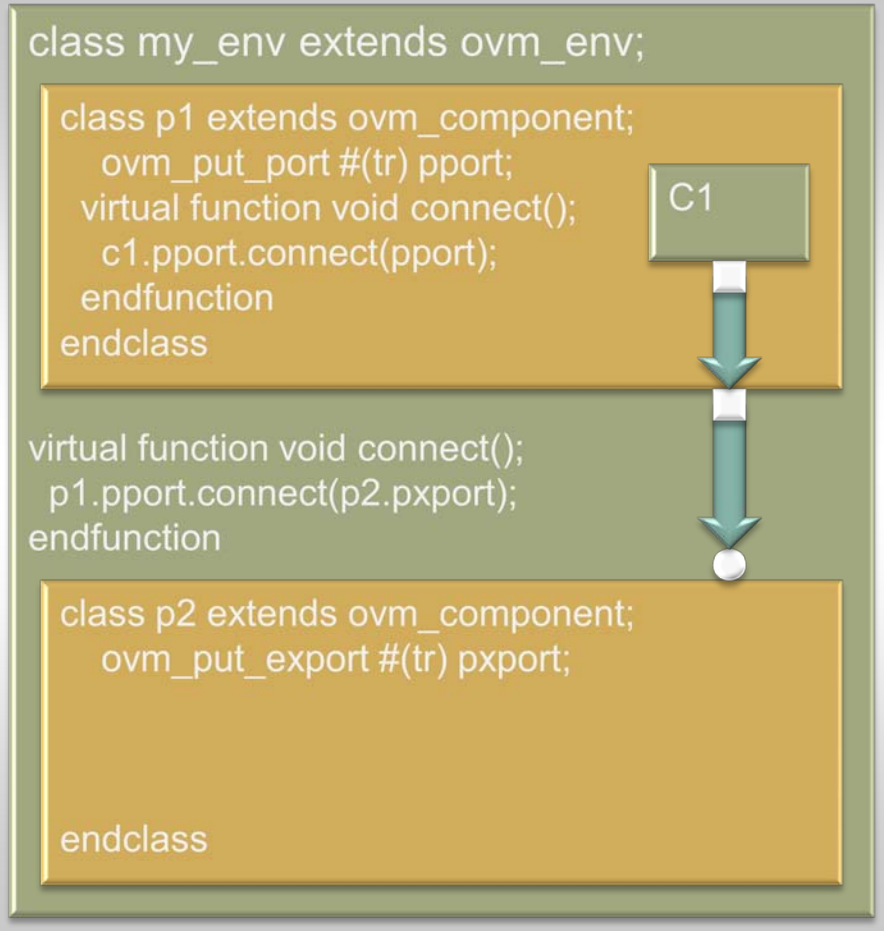
Hierarchical Connections

- **Port-to-Export**
  port.connect(export);
- **Port-to-Port**



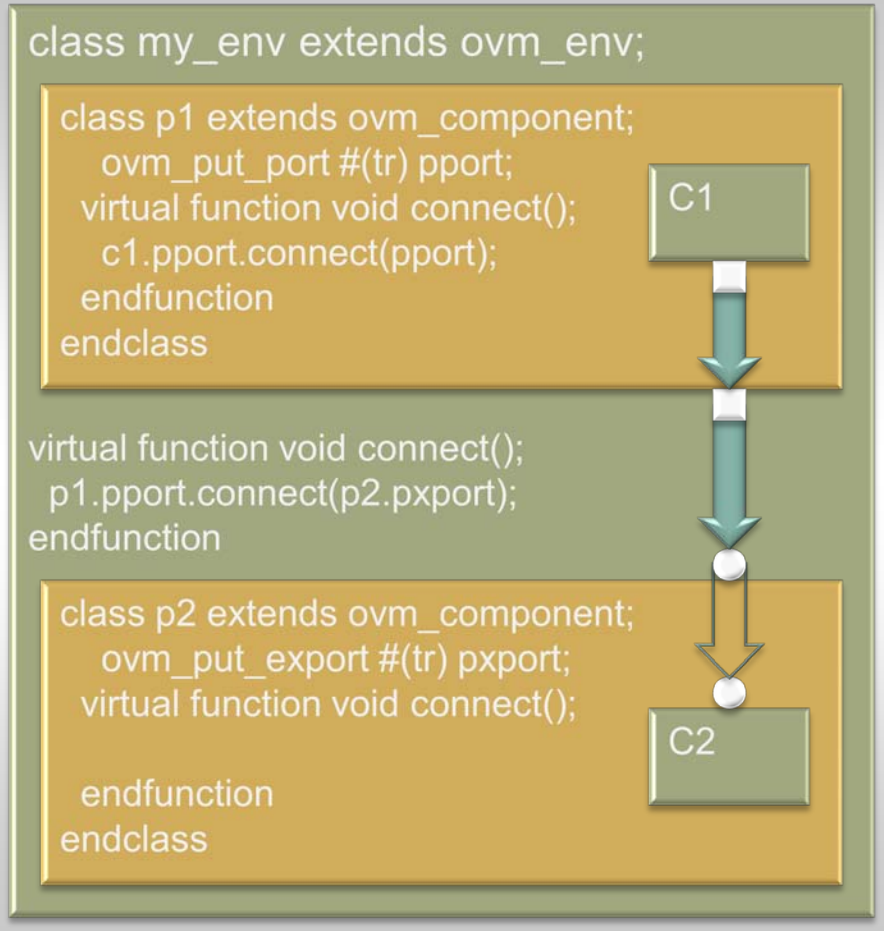
Hierarchical Connections

- **Port-to-Export**
  port.connect(export);
- **Port-to-Port**
  child.port.connect(parent_port);



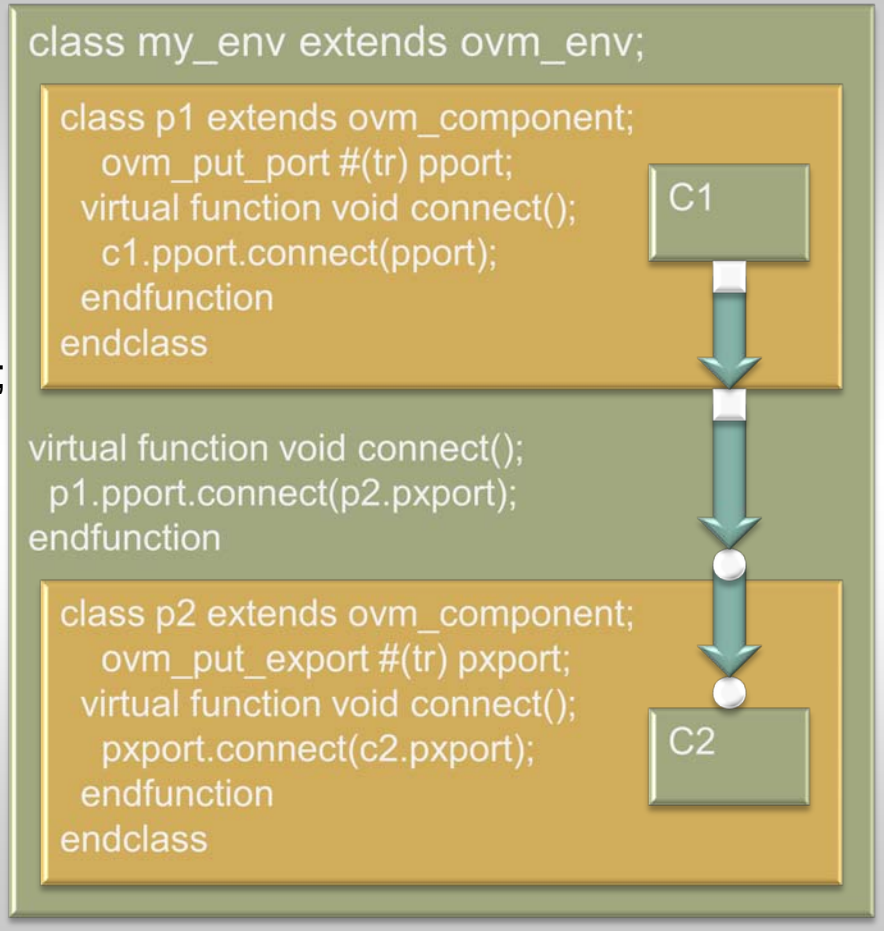
Hierarchical Connections

- **Port-to-Export**
  port.connect(export);
- **Port-to-Port**
  child.port.connect(parent_port);
- **Export-to-Export**



Hierarchical Connections

- **Port-to-Export**
  port.connect(export);
- **Port-to-Port**
  child.port.connect(parent_port);
- **Export-to-Export**
  parent_export.connect(child.export);



Export vs. Imp

- **Every port must eventually connect to an implementation (imp)**
- **Most imps you'll need are built into base classes**
  - analysis_imp in ovm_subscriber
  - seq_item_pull_imp in ovm_sequencer
  - blocking/nonblocking put/get/peek in tlm_fifo
- **Occasionally, you may need to make your own imp**

Creating Your Own Imp

1. Declare the appropriate imp
2. Declare the implementation of the appropriate TLM method(s)
3. Construct the imp in build()

```
class my_comp extends ovm_component;

  ovm_put_imp #(T, my_comp) put_export;

  function new(string name,
               ovm_component parent);
    super.new(name, parent);
  endfunction: new
  virtual task put(T tr);
    ...
  endtask

  function void build;
    super.build();
    put_export = new("put_export",
                     this);
    ...
  endfunction
...
endclass
```

Transaction type

Parent type

Factory not required

Implementing a Scoreboard

- **Need multiple analysis exports**
  - Each export needs its own write() method
  - Can only have one write() method in the component

```
class my_sbd extends ovm_component;
  ovm_analysis_export #(tr) exp_export;
  ovm_analysis_export #(tr) act_export;

  function void build();
    exp_export = new("expected export", this);
    act_export = new("actual export", this);


  endfunction
  virtual function void connect();


  endfunction
  virtual task run();



  endtask
```

Implementing a Scoreboard

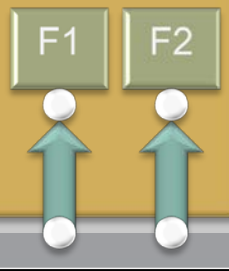
- **Need multiple analysis exports**
  - Each export needs its own write() method
  - Can only have one write() method in the component
- **Use analysis_fifos to supply the exports**

```
class my_sbd extends ovm_component;
  ovm_analysis_export #(tr) exp_export;
  ovm_analysis_export #(tr) act_export;
  local tlm_analysis_fifo #(tr) F1, F2;

  function void build();
    exp_export = new("expected export", this);
    act_export = new("actual export", this);
    F1 = new("expected fifo", this);
    F2 = new("actual fifo", this);
  endfunction

  virtual function void connect();


  endfunction

  virtual task run();


  endtask
```

Implementing a Scoreboard

- **Need multiple analysis exports**
  - Each export needs its own write() method
  - Can only have one write() method in the component
- **Use analysis_fifos to supply the exports**

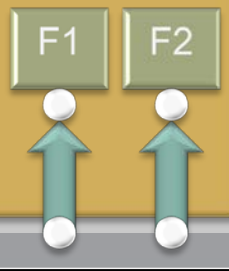
```
class my_sbd extends ovm_component;
  ovm_analysis_export #(tr) exp_export;
  ovm_analysis_export #(tr) act_export;
  local tlm_analysis_fifo #(tr) F1, F2;

  function void build();
    exp_export = new("expected export", this);
    act_export = new("actual export", this);
    F1 = new("expected fifo", this);
    F2 = new("actual fifo", this);
  endfunction

  virtual function void connect();
    exp_export.connect(F1.analysis_export);
    act_export.connect(F2.analysis_export);
  endfunction

  virtual task run();



  endtask
```



Implementing a Scoreboard

- **Need multiple analysis exports**
  - Each export needs its own write() method
  - Can only have one write() method in the component
- **Use analysis_fifos to supply the exports**
  - Scoreboard actively pulls from fifos

```
class my_sbd extends ovm_component;
  ovm_analysis_export #(tr) exp_export;
  ovm_analysis_export #(tr) act_export;
  local tlm_analysis_fifo #(tr) F1, F2;

  function void build();
    exp_export = new("expected export", this);
    act_export = new("actual export", this);
    F1 = new("expected fifo", this);
    F2 = new("actual fifo", this);
  endfunction

  virtual function void connect();
    exp_export.connect(F1.analysis_export);
    act_export.connect(F2.analysis_export);
  endfunction

  virtual task run();
    F1.get(exp_tr);
    F2.get(act_tr);
    // compare transactions
  endtask
```



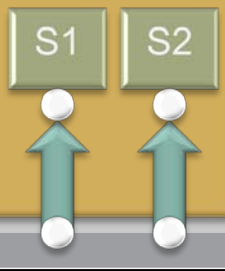
Alternate Scoreboard Implementation

- **Use subscribers to call methods in the parent**
- **Scoreboard is passive recipient of transactions**

```
class my_subscriber1 #(type tr)
      extends ovm_subscriber #(tr);
  virtual function void write(tr);
    m_parent.write1(tr);
  endfunction
endclass
```

```
class my_subscriber2 #(type tr)
      extends ovm_subscriber #(tr);
  virtual function void write(tr);
    m_parent.write2(tr);
  endfunction
endclass
```

```
class my_sbd extends ovm_component;
  ovm_analysis_export #(tr) exp_export;
  ovm_analysis_export #(tr) act_export;
  local my_subscriber1  #(tr) S1;
  local my_subscriber2  #(tr) S2;

  function void build();
    …
  endfunction

  virtual function void connect();
    exp_export.connect(S1.analysis_export);
    act_export.connect(S2.analysis_export);
  endfunction

  virtual function void write1();
    …
  endfunction

  virtual function void write2();
    …
  endfunction
```



VERIFICATION ACADEMY

Advanced OVM (& UVM)

Understanding TLM

Tom Fitzpatrick
Verification Technologist

academy@mentor.com
www.verificationacademy.com

Mentor Graphics®