Improving Data Monitoring In UVM

Tips & Recommendations

Yogish Sekhar

Dialog Semiconductors

May 22, 2014

Reading, United Kingdom

Agenda

Review Of Current UVM Test Bench Structure

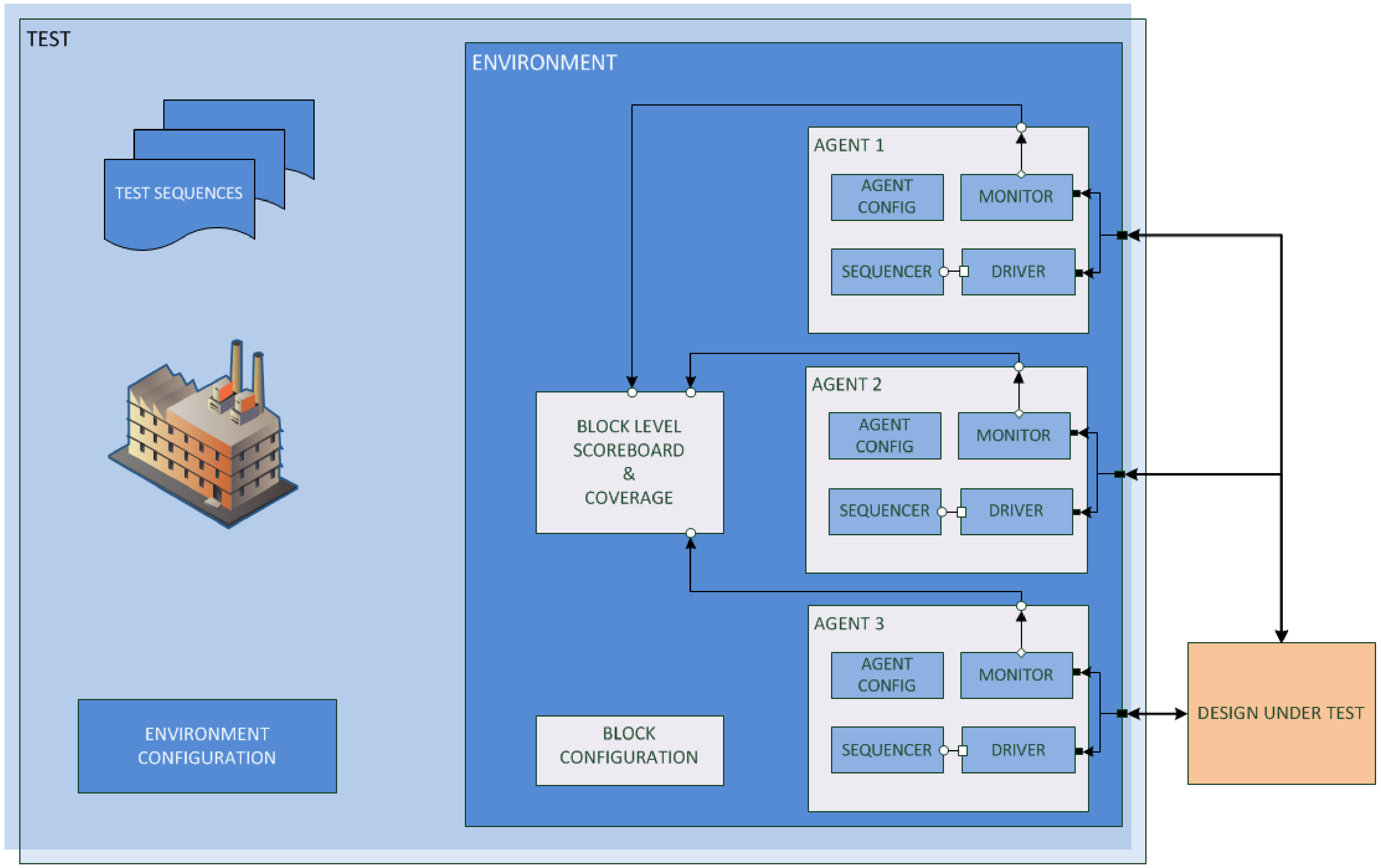
- Environment Architecture
- Limitations Of Current Analysis Ports / Port Macro’s

Improvements & Dynamic Connections

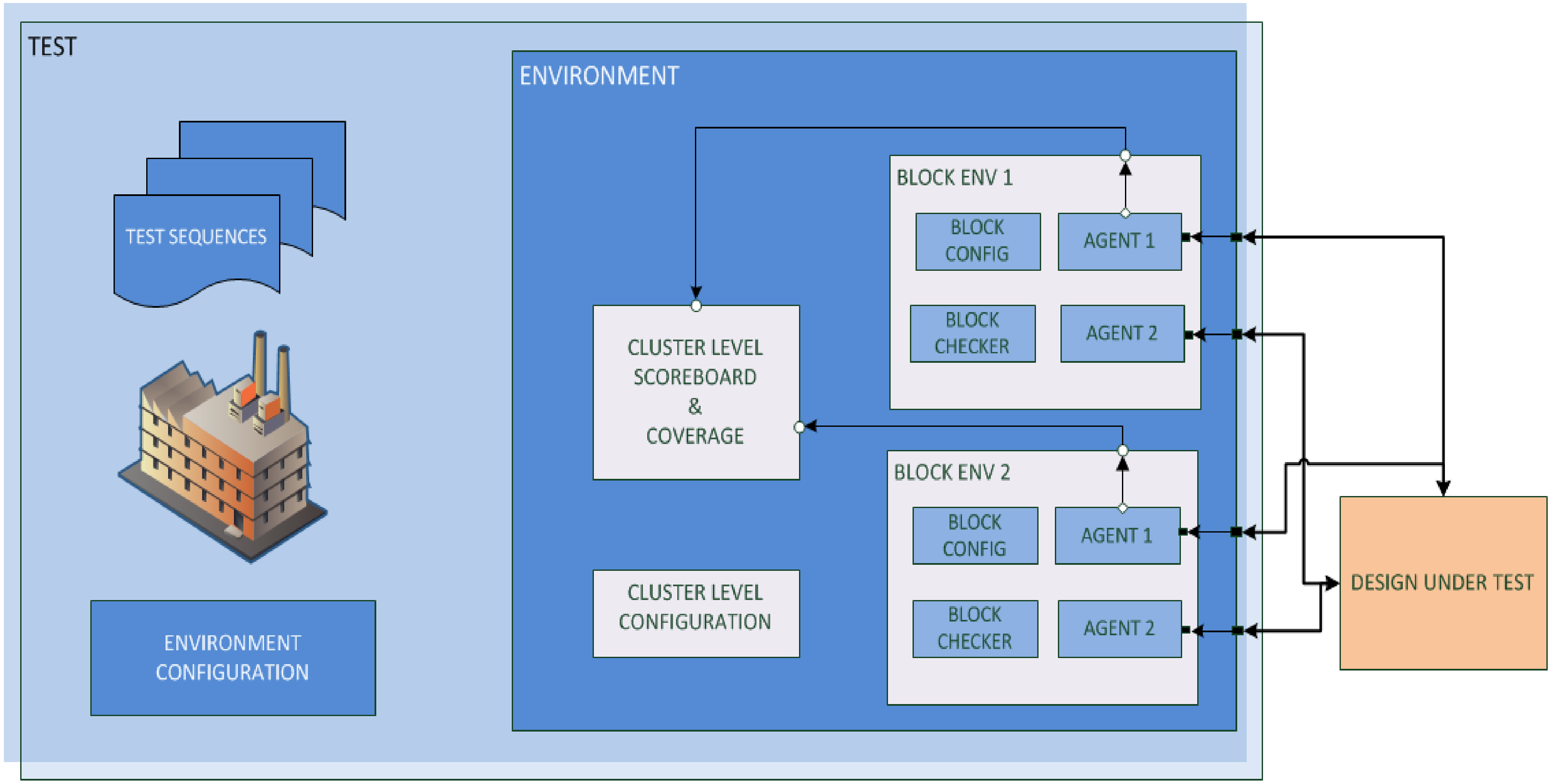
TLM & UVM

Review of Current UVM Testbench Structure

Current UVM TB Structure
- Block Level



Current UVM TB Structure
- Cluster Level



Current UVM TB Structure

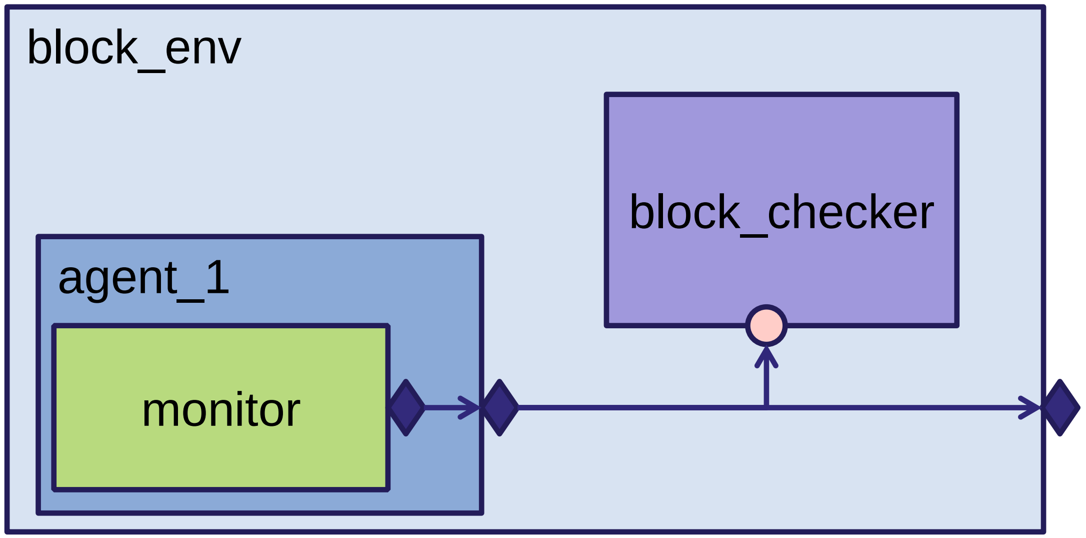
Characteristics

- Inherently Structural
- Quasi Static Connections
- Each Level Of Hierarchy Need To Support Data Transport

```
function void agent::connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    `uvm_info(name, "Inside agent::connect()", UVM_LOW);

    if (driver != null && sequencer != null) begin
      // connect the driver with the sequencer
      driver.seq_item_port.connect (sequencer.seq_item_export);
    end
    monitor.analysis_port.connect (monitor_ap);
endfunction : connect_phase
```

Current UVM TB Structure

Connect Phase - Environments

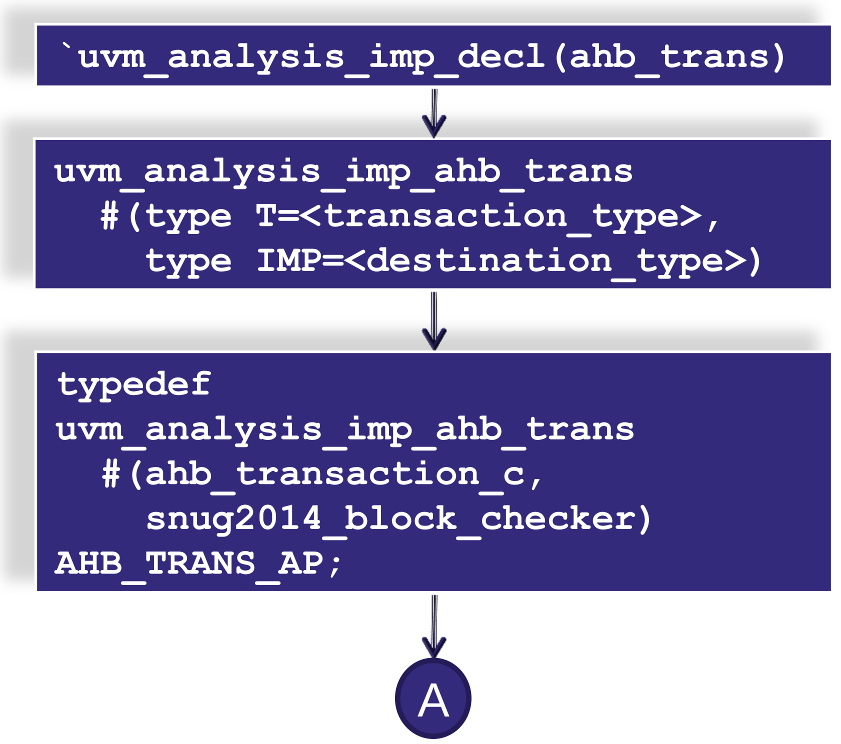


```
function void block_env::connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    `uvm_info(name, "Inside block_env::connect()", UVM_LOW);
    agent_1.monitor_ap.connect (block_checker.agent_1_ap);
    agent_2.monitor_ap.connect (block_checker.agent_2_ap);
    agent_1.monitor_ap.connect (this.agent_1_ap);
    agent_2.monitor_ap.connect (this.agent_2_ap);
endfunction : connect_phase
```

Current Connection Model

Subscriber Implementation

- 4 Stage Process
  - Stage 1
    - Provide an Implementation
    - Use one of the many `'*_imp_decl(<suffix>)'` macro
    - Customize the template class



```
`uvm_analysis_imp_decl(ahb_trans)
```

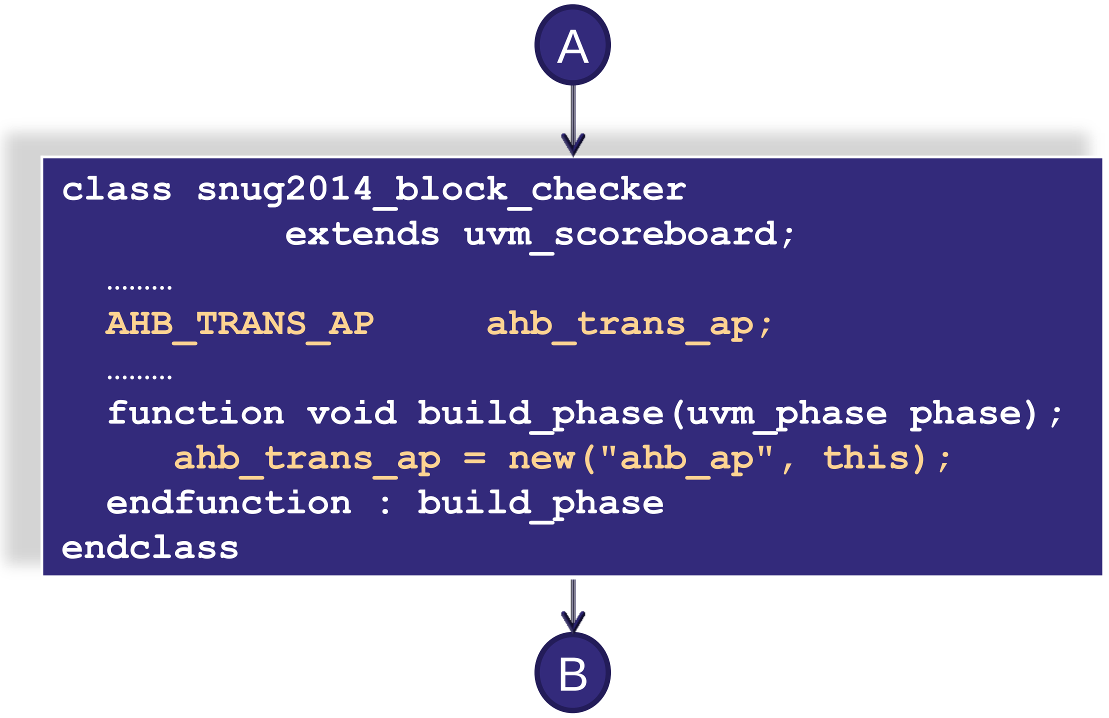
```
uvm_analysis_imp_ahb_trans
  #(type T=<transaction_type>,
    type IMP=<destination_type>)
```

```
typedef
uvm_analysis_imp_ahb_trans
  #(ahb_transaction_c,
    snug2014_block_checker)
AHB_TRANS_AP;
```

Current Connection Model

Subscriber

- 4 Stage Process
  - Stage 2
    - Instance the customized class
    - Provide it with reference handle of the subscriber



```
class snug2014_block_checker
            extends uvm_scoreboard;
  ………
  AHB_TRANS_AP      ahb_trans_ap;
  ………
  function void build_phase(uvm_phase phase);
     ahb_trans_ap = new("ahb_ap", this);
  endfunction : build_phase
endclass
```

Current Connection Model

Subscriber

- 4 Stage Process
  - Stage 3
    - Ensure ‘write_<suffix>(T trans)’ is implemented in the subscriber
  - Stage 4
    - Connect to a data source through a sequence of hierarchical port/export connections

B

```
class snug2014_block_checker
           extends uvm_scoreboard;
  .........
  AHB_TRANS_AP     ahb_trans_ap;
  .........
  function void write_ahb_trans
       (ahb_transaction_c  trans);
  endfunction
  .........
endclass
```

Current Connection Model

Subscriber

Limitations

- Need for template class customisation
  - This process can be defined inside the macro itself
- Name of the method in subscriber
- Using of connect_phase() through the hierarchy

B

```
class snug2014_block_checker
          extends uvm_scoreboard;
  .........
  AHB_TRANS_AP     ahb_trans_ap;
  .........
  function void write_ahb_trans
       (ahb_transaction_c  trans);
  endfunction
  .........
endclass
```

Improvements For Establishing Dynamic Connections

Elements For Active Connection

- To Create Active Connection
  - Data Source
    - Monitor
  - Transaction Type
    - Derived from `uvm_sequence_item`
  - Destination Type
    - Type of the subscriber
  - Destination Method
    - Function in the subscriber capable of processing transaction provided by the monitor

UVM Monitor Base Class

```
virtual class uvm_monitor extends uvm_component;
    function new (string name, uvm_component parent);
      super.new(name, parent);
    endfunction

    const static string type_name = "uvm_monitor";

    virtual function string get_type_name ();
      return type_name;
    endfunction
endclass
```

- It’s Just a wrapper on `‘uvm_component’`
- Specific Monitor Implementations
  - Customize the output transaction
  - Always associated with an interface

UVM Monitor Base Class

- Template the base monitor class
- Provides some common elements
  - Default analysis port
  - Broadcast method

```
virtual class dvm_monitor #(type T = uvm_sequence_item) extends uvm_monitor;
     T mon_trans;

     // Declare a default analysis port in the monitor
     uvm_analysis_port#(T) default_ap;

     // registration bit for callbacks
     static local bit m_register_cb;

     `uvm_component_param_utils(dvm_monitor #(T))
     extern function new (string name, uvm_component parent);
     extern function void notify(T trans);
endclass : dvm_monitor
```

Improved Analysis Macro

```
`uvm_analysis_imp_decl(ahb_trans)
```

```
`dvm_analysis_imp_decl(ANALYSIS_OBJ_TYPE, MONITOR_NAME, TRANS_TYPE, RCV_TYPE, RCV_FUNC)
```

Parameter	Usage
`ANALYSIS_OBJ_TYPE`	Unique name to identify the customised analysis object class type
`MONITOR_NAME`	Name string to uniquely identify the monitor component Can contain wild characters to identify hierarchical components
`TRANS_TYPE`	Data transaction class type used by the monitor
`RCV_TYPE`	Class type of the receiver
`RCV_FUNC`	User defined method available in the in the subscriber that processes a transaction of type `'TRANS_TYPE'`

Improved Analysis Macro

- `‘auto_connect()’`
  - Hierarchical `‘connect_phase()’` redundant
  - Connections
    - Established by subscribers that need the information
    - Can be established any time after `‘build_phase()’`
    - No hierarchical components is involved in data transport
    - UVM component registry to find the necessary data sources

Improved Analysis Macro

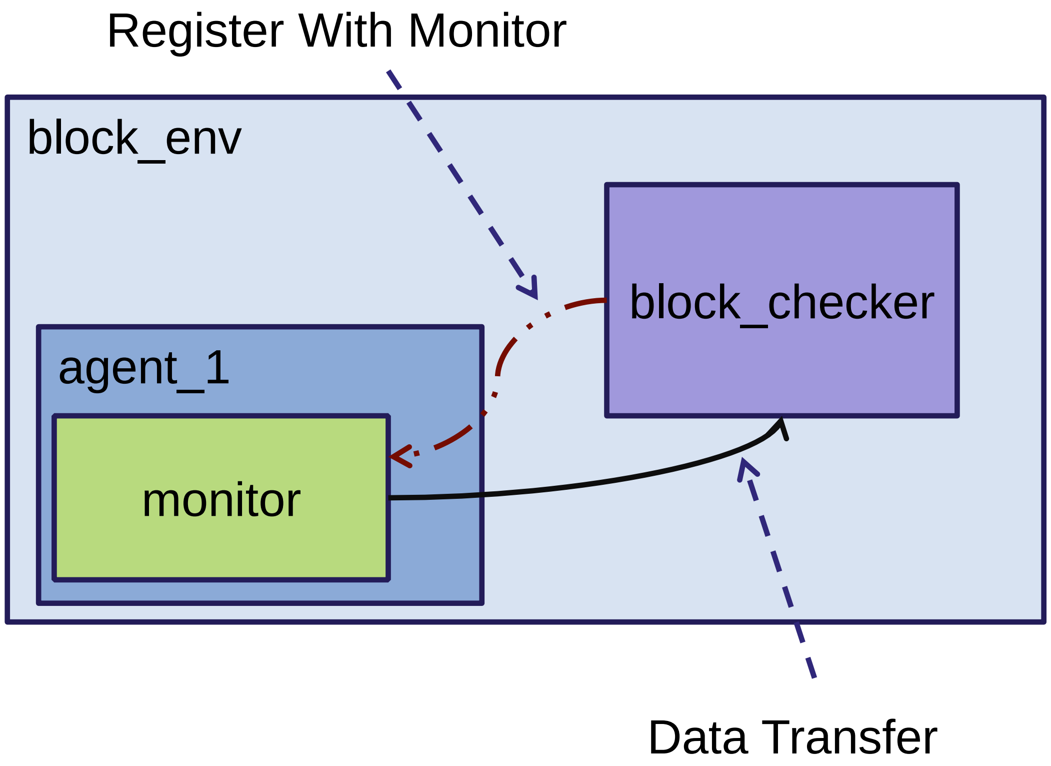
Block Level UVM Environment

```
class snug2014_checker_1 extends uvm_component;
    // Create the classes needed for analysis objects
    `dvm_analysis_imp_decl(
          CK1_AP_MON_1, "*.snug2014_block_env_1.*.agent_1_monitor",
          snug2014_agent_1_transaction, snug2014_checker_1, func_agent_1)

    // Declare the Analysis Objects that are to be used
    CK1_AP_MON_1 ap_mon_1;
    ……………

    // User Defined Callback functions for Analysis
    extern function void func_agent_1 (snug2014_agent_1_transaction trans);
endclass : snug2014_checker_1
```

Example Testbench

Block Level UVM Environment

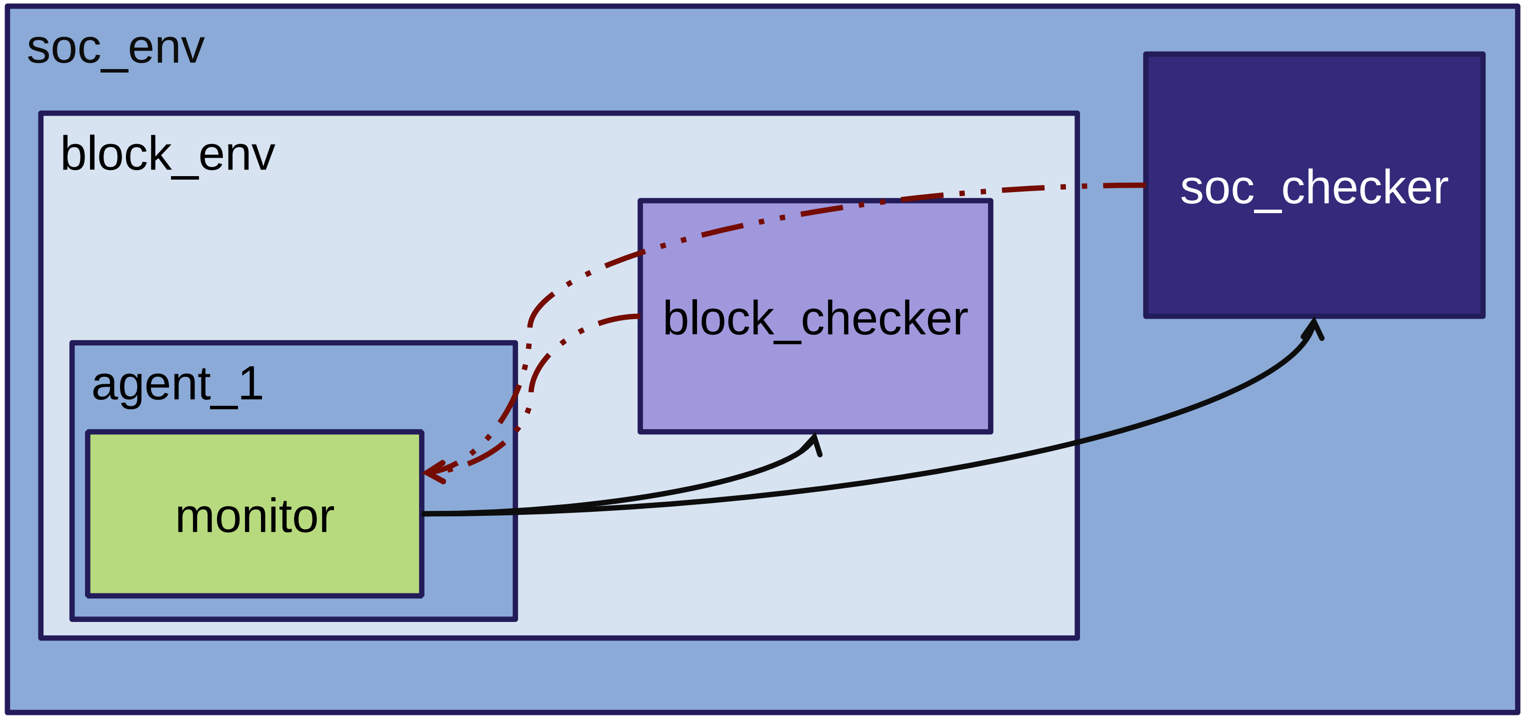
```
function void snug2014_checker_1::<connect | post_elab | run>_phase(uvm_phase phase);
    // Call auto_connect inorder to connect up the monitor and the checker
    ap_mon_1.auto_connect();
    ap_mon_2.auto_connect();
endfunction : connect_phase
```



Example Testbench

Block Level UVM Environment

```
function void snug2014_checker_1::<connect | post_elab | run>_phase(uvm_phase phase);
    // Call auto_connect inorder to connect up the monitor and the checker
    ap_mon_1.auto_connect();
    ap_mon_2.auto_connect();
endfunction : connect_phase
```



Example Testbench – Simulation - 1

```
UVM_INFO @ 0ns : [checker_1]                 : Inside snug2014_checker_1::connect_phase()
UVM_INFO @ 0ns : [ANALYSIS_PORT_AGENT_1]: CK1_AP_MON_1::auto_connect()
UVM_INFO @ 0ns : [ANALYSIS_PORT_AGENT_1]:
              Finding Monitor: *.SNUG2014_BLOCK_ENV_1.*.AGENT_1_MONITOR
UVM_INFO @ 0ns : [ANALYSIS_PORT_AGENT_1]:
              Connected     : uvm_test_top.snug2014_block_env_1.agent_1.agent_1_monitor
              To            : uvm_test_top.snug2014_block_env_1.snug2014_checker_1
              Transaction   : snug2014_agent_1_transaction
              Function Call: func_agent_1
UVM_INFO @ 0ns : [ANALYSIS_PORT_AGENT_2]: CK1_AP_MON_2::auto_connect()
UVM_INFO @ 0ns : [ANALYSIS_PORT_AGENT_2]:
              Finding Monitor: *.SNUG2014_BLOCK_ENV_1.*.AGENT_2_MONITOR
UVM_INFO @ 0ns : [ANALYSIS_PORT_AGENT_2]:
              Connected     : uvm_test_top.snug2014_block_env_1.agent_2.agent_2_monitor
              To            : uvm_test_top.snug2014_block_env_1.snug2014_checker_1
              Transaction   : snug2014_agent_2_transaction
              Function Call: func_agent_2
UVM_INFO @ 0ns : [snug2014_env_1] :
              Inside snug2014_block_env_1::connect_phase() - Nothing Done Here
```

```
UVM_INFO @ 100ns : [agent_1_monitor] :
                 Something Interesting Seen Here - Build Transaction
UVM_INFO @ 100ns : [agent_1_monitor] : dvm_monitor::notify() - Called
UVM_INFO @ 100ns : [checker_1] : Inside snug2014_checker_1::func_agent_1()

UVM_INFO @ 200ns : [agent_2_monitor] :
                 Something Interesting Seen Here - Build Transaction
UVM_INFO @ 200ns : [agent_2_monitor] : dvm_monitor::notify() - Called
UVM_INFO @ 200ns : [checker_1] : Inside snug2014_checker_1::func_agent_2()
```

Example Testbench – Simulation 2

```
UVM_INFO @ 0ns : [checker_soc] : Inside snug2014_soc_checker::connect_phase()
UVM_INFO @ 0ns : [SOC_CHK_CL_ANALYSIS_CONN_OBJ_4] : CK_SOC_C1_AP_MON_4::auto_connect()
UVM_INFO @ 0ns : [SOC_CHK_CL_ANALYSIS_CONN_OBJ_4] :
               Finding Monitor: *.SNUG2014_CLUSTER_ENV.*.AGENT_4_MONITOR
UVM_INFO @ 0ns : [SOC_CHK_CL_ANALYSIS_CONN_OBJ_4] :
               Connected   : uvm_test_top.snug2014_soc_env.snug2014_cluster_env
                             .snug2014_block_env_2.agent_4.agent_4_monitor
               To          : uvm_test_top.snug2014_soc_env.checker_soc
               Transaction : snug2014_agent_4_transaction
               Function    : cluster_func_agent_4
UVM_INFO @ 0ns : [SOC_CHK_BLK_ANALYSIS_CONN_OBJ_4]: CK_SOC_AP_MON_4::auto_connect()
UVM_INFO @ 0ns : [SOC_CHK_BLK_ANALYSIS_CONN_OBJ_4]:
               Finding Monitor: *.SNUG2014_BLOCK_ENV_3.*.AGENT_4_MONITOR
UVM_INFO @ 0ns : [SOC_CHK_BLK_ANALYSIS_CONN_OBJ_4]:
               Connected   : uvm_test_top.snug2014_soc_env.snug2014_block_env_3
                             .agent_4.agent_4_monitor
               To          : uvm_test_top.snug2014_soc_env.checker_soc
               Transaction : snug2014_agent_4_transaction
               Function    : block_func_agent_4
UVM_INFO @ 0ns : [snug2014_block_env_2] :
               Inside snug2014_block_env_2::connect_phase() - Nothing Done Here
UVM_INFO @ 0ns : [snug2014_cluster_env] :
               Inside snug2014_cluster_env::connect_phase() - Nothing Done Here
UVM_INFO @ 0ns : [snug2014_soc_env] :
               Inside snug2014_soc_env::connect_phase() - Nothing Done Here
```

Current Vs Proposed

```
`uvm_analysis_imp_decl(ahb_trans)
```

```
uvm_analysis_imp_ahb_trans
  #(type T=<transaction_type>,
    type IMP=<destination_type>)
```

```
typedef uvm_analysis_imp_ahb_trans
  #(ahb_transaction_c,
    snug2014_block_checker)
AHB_TRANS_AP;
```

```
class snug2014_block_checker
          extends uvm_scoreboard;
  ………
  AHB_TRANS_AP     ahb_trans_ap;
  ………
endclass
```

```
class snug2014_block_checker
          extends uvm_scoreboard;
  ………
  function void write_ahb_trans
      (ahb_transaction_c  trans);
endclass
```

```
ahb_trans_ap = new("AP");

// Use of Hierarchical
connections
```

```
`dvm_analysis_imp_decl(
CK_SOC_C1_AP_MON_2,
"*.snug2014_cluster_env.*.agent_2_monitor",
snug2014_agent_2_transaction,
snug2014_soc_checker,
cluster_func_agent_2)
```

```
class snug2014_soc_checker
          extends uvm_scoreboard;
  ………
  CK_SOC_C1_AP_MON_2     ck_soc_c1_ap_mon_2;
  ………
endclass
```

```
class snug2014_soc_checker
          extends uvm_scoreboard;
  ………
  function void cluster_func_agent_2
        (snug2014_agent_2_transaction  trans);
  ………
endclass
```

```
ck_soc_c1_ap_mon_2 = new( "AP", this);
ck_soc_c1_ap_mon_2.auto_connect()
```

TLM & UVM

TLM & UVM

- TLM 2.0
  - Standardized Approach
    - Creating Models & Transaction Level Simulations
    - Enables Model Exchange
    - Provides Common Ground For Interfacing
- UVM Implementation - Analysis Ports
  - Based On OSCI Standard & SystemC Implementation
  - SystemC Analysis Ports
    - Not derived from ‘sc_port’
    - Connections in SystemC analysis ports can be established even in the ‘run_phase’
  - UVM Restrictive In The Way Analysis Ports Can Be Bound Or Connected

TLM & UVM

Impacts

- Performance Impact When Executing SOC Level Hardware-Software Co-Simulations
- Increased Memory Footprint
- Inability To Selectively Enable Block Level Environments After SOC Initialisation
- Re-Initialisation Of SOC In Every Testcase If Enabling Of Block Level Checkers At Runtime By Use Of Plusargs

Solution

- Make Analysis Ports Independent Of ‘uvm_port_base’
  – Create A New Base Class Specifically For Analysis Ports
  – Bring In UVM Implementation Closer To SystemC Implementation

Conclusion

- Advantages
  - Reduction In Code To Support Data Movement
  - Establishes 1-1 Connection between Monitors and Checkers
    - Eliminating connect_phase() in hierarchical components
    - Subscribers Processing Transactions Responsible For Connections
    - Library Ensures Transaction Type Checking At Compile Time
  - Simplification In Connection Model
    - Allows Callbacks To Be Used Just As Analysis Ports
    - Allows Feedback To Active Sequences Needing Information From Monitors. Eg. Interrupt Monitoring
  - Use Of Wild Character Based Search String
  - Simplifies Debug As All Information Regarding Establishing Data Connection Available At A Single Location

Conclusion

- Limitations
  - Monitors Only Broadcast
    - Effectively Limited To A Push Model For Data Transmission
  - Monitor Search Path Strings Should Be Uniqifiable
    - Identify A Single Data Source

Thank You