UVM Scoreboards and Checkers
Memory, TLB and Cache

Rich Edelman, Siemens EDA, Fremont, CA US
C.H. Liu, Siemens EDA, Taiwan
What’s a Scoreboard or a Checker?
A UVM Testbench – The Egg Diagram
What’s an Analysis Port?

• What’s it good for?
• One-to-many connections
• Implements the Publisher/Subscriber pattern
• Unconnected analysis ports are OK
• They can be used in a monitor which must consume no time (they are implemented as functions)
Code for a UVM Monitor

```yml
class monitor extends uvm_monitor;
  `uvm_component_utils(monitor)

virtual bus_interface vif;
uvm_analysis_port #(transaction) ap;

function new(string name = "monitor", uvm_component parent = null);
  super.new(name, parent);
endfunction

function void build_phase(uvm_phase phase);
  super.build_phase(phase);
  ap = new("ap", this);
endfunction

transaction t;
```
task run_phase(uvm_phase phase);
    forever begin
        @(posedge vif.clk);
        if (vif.valid == 1) begin
            t = transaction::type_id::create("transaction");
            t.addr = vif.addr;
            if (vif.rw == READ)
                t.data = vif.odata;
            else
                t.data = vif.idata;
            t.rw = RW_T'(vif.rw);
            t.id = vif.id;
            t.serial_number = vif.serial_number;
            ap.write(t);
        end
    end
endtask
declass
An In-Order Scoreboard

• Expected and Actual
• The Model
A Memory scoreboard

class memory_scoreboard extends uvm_component;
  `uvm_component_utils(memory_scoreboard)

  uvm_analysis_imp_expected #(transaction, memory_scoreboard) expected_analysis_export;
  uvm_analysis_imp_actual  #(transaction, memory_scoreboard) actual_analysis_export;

  uvm_tlm_analysis_fifo   #(transaction)   expected_fifo;
  uvm_tlm_analysis_fifo   #(transaction)   actual_fifo;
Using `uvm_analysys_imp_decl`

- Convenient way to create analysis port connections
- Built into the UVM

```plaintext
class uvm_analysis_imp_expected #(type T=int, type IMP=int) extends uvm_port_base #(uvm_tlm_if_base #(T,T));
  local IMP m_imp;
  function new (string name, IMP imp);
    super.new (name, imp, UVM_IMPLEMENTATION, 1, 1);
    m_imp = imp;
    m_if_mask = (1<<8);
  endfunction
  virtual function string get_type_name();
    return "uvm_analysis_imp_expected";
  endfunction
  function void write( input T t);
    m_imp.write_expected( t);
  endfunction
endclass

'\uvm_analysis_imp_decl(_expected)
'\uvm_analysis_imp_decl(_actual)

uvm_analysis_imp_expected#(transaction, memory_scoreboard) expected_analysis_export;
uvm_analysis_imp_actual #(transaction, memory_scoreboard) actual_analysis_export;
```
The Memory Itself

reg [7:0] mem [256];

function mem_write(reg [7:0] addr, reg [7:0] data);
  mem[addr] = data;
endfunction

function reg [7:0] mem_read(reg [7:0] addr);
  return mem[addr];
endfunction
write_expected and write_actual

```markdown
transaction write_expected_tr;
transaction write_actual_tr;

function void write_expected( input transaction expected_tran);

`uvm_info("SCOREBOARD","WriteExpected:",expected_tran.convert2string(),UVM_MEDIUM)
write_expected_tr = expected_tran;
if (expected_tran.rw == 1)
    expected_tran.data = mem_read(expected_tran.addr);
else
    mem_write(expected_tran.addr, expected_tran.data);
`uvm_info("SCOREBOARD","WriteExpected2:",expected_tran.convert2string(),UVM_MEDIUM)
    void'(expected_fifo.try_put(expected_tran));
endfunction

function void write_actual( input transaction actual_tran);

`uvm_info("SCOREBOARD","WriteActual:",actual_tran.convert2string(),UVM_MEDIUM)
write_actual_tr = actual_tran;
    void'(actual_fifo.try_put(actual_tran));
endfunction
```
transaction run_expected_tr;
transaction run_actual_tr;

task run_phase( uvm_phase phase );
    forever begin
        expected_fifo.get(run_expected_tr);
        actual_fifo.get(run_actual_tr);
        `uvm_info("SCOREBOARD", {"Expected:" , run_expected_tr.convert2string()}, ... 
        `uvm_info("SCOREBOARD", {" Actual:" , run_actual_tr.convert2string() }, ... 
        if (run_actual_tr.compare(run_expected_tr))
            `uvm_info("SCOREBOARD RESULTS","MATCH!",UVM_MEDIUM)
        else
            `uvm_error( "SCOREBOARD RESULTS","MISMATCH!")
    end 
endtask
UVM check_phase

// Check for memory_scoreboard empty at end of test if configured to do so

virtual function void check_phase( uvm_phase phase);
  `uvm_info(get_type_name(), "...starting check_phase", UVM_MEDIUM)
  if ((expected_fifo.size() != 0))
    `uvm_error("SCOREBOARD RESULTS","EXPECTED FIFO NOT EMPTY")
  if ((actual_fifo.size() != 0))
    `uvm_error("SCOREBOARD RESULTS","ACTUAL FIFO NOT EMPTY")
endfunction
An Out-of-Order Scoreboard

SCOREBOARD

COMPARE

Model of Expected Behavior

EXPECTED

ACTUAL

DUT

MEMORY

AGENT

MONITOR

DRIVER

Sequence

INPUT

OUTPUT

DUT

8,4,2,6
1,2,3,4

DUT

1,2,3,4
Changing the scoreboard for out-of-order

```plaintext
class out_of_order_memory_scoreboard extends uvm_component;
  `uvm_component_utils(out_of_order_memory_scoreboard)

uvm_analysis_imp_expected#(transaction, out_of_order_memory_scoreboard) expected_analysis_export
uvm_analysis_imp_actual #(transaction, out_of_order_memory_scoreboard) actual_analysis_export;

  transaction expected_associative_array[int];
  transaction actual_associative_array[int];
```
write_actual and write_expected

```verilog
transaction write_expected_tr;
transaction write_actual_tr;

function void write_expected( input transaction expected_tran);
    write_expected_tr = expected_tran;

    expected_associative_array[expected_tran.serial_number] = expected_tran;
    if (expected_tran.rw == 1)
        expected_tran.data = mem_read(expected_tran.addr);
    else
        mem_write(expected_tran.addr, expected_tran.data);
    `uvm_info("SCOREBOARD","WriteExpected2:",expected_tran.convert2string()),...

    checkcheck(expected_tran.serial_number);
endfunction
```
write_actual and write_expected

function void write_actual( input transaction actual_tran);
    write_actual_tr = actual_tran;

    actual_associative_array[actual_tran.serial_number] = actual_tran;

    checkcheck(actual_tran.serial_number);
endfunction
The checkcheck() routine

```md
function void checkcheck(bit [31:0] serial_number);
    if (actual_associative_array.exists(serial_number) &&
        expected_associative_array.exists(serial_number)) begin

        run_actual_tr = actual_associative_array[serial_number];
        run_expected_tr = expected_associative_array[serial_number];
        if (out_of_order_compare(run_expected_tr, run_actual_tr)) begin
            `uvm_info("OUT_OF_ORDER_SCOREBOARD RESULTS","MATCH!",UVM_MEDIUM)
        end
        else begin
            `uvm_error("OUT_OF_ORDER_SCOREBOARD RESULTS","MISMATCH!")
            ...
        end
        actual_associative_array.delete(serial_number);
        expected_associative_array.delete(serial_number);
    end
endfunction
```
What’s a TLB?

• Simple model
• Maps a Virtual address to a Physical address

• Virtual Address

```
struct {
    bit [31:12] tag;
    bit [11: 0] offset;
}
```
class tlb;
    typedef struct packed {
        bit [31:12] tag;
        bit [11: 0] offset;
    } vaddress_t;

typedef bit [31: 0] address_t;
typedef bit [31:12] tag_t;

tag_t lookup_table[tag_t];
function void create_a_mapping(address_t virtual_address);
    // Paw through the OS and find the mapping...
    vaddress_t paddress;
    vaddress_t vaddress;
    paddress = virtual_address;
    paddress.tag = paddress.tag + 1;
    vaddress = virtual_address;
    lookup_table[vaddress.tag] = paddress.tag;
endfunction

function void evict_a_page_if_necessary(address_t virtual_address);
    // Check if there is room. We may have to evict a mapping.
    // ...
endfunction
function void find_new_mapping(address_t virtual_address);
    evict_a_page_if_necessary(virtual_address); // No mapping for this address
    create_a_mapping(virtual_address);       // There is room now.
endfunction

function address_t find_physical_address(address_t virtual_address);
    vaddress_t paddress;
    vaddress_t vaddress;
    vaddress = virtual_address;
    if (!lookup_table.exists(vaddress.tag))
        find_new_mapping(vaddress);

    paddress = vaddress;
    paddress.tag = lookup_table[vaddress.tag];
    return paddress;
endfunction
endclass
What’s a Cache?

• Simple model
What’s a Cache?

```c
typedef struct packed {
    bit [25:0] tag; // 26 bits - lots of values
    bit [1:0] set; // 2 bits - 4 values
    bit [3:0] word; // 4 bits - 16 values
} address_t;
```
typedef enum {M, S, I} MSI_STATE;
typedef struct {
    MSI_STATE state;
    reg [7:0] bytes[16]; // The offset picks the byte
} cache_line_t;

typedef struct {
    cache_line_t cache_line[int]; // The tag picks the line
} cache_set_t;

typedef struct {
    cache_set_t cache_sets[4]; // The set picks the cache
} cache_t;

typedef struct packed {
    bit [25:0] tag; // 26 bits - lots of values
    bit [1:0] set; // 2 bits - 4 values
    bit [3:0] word; // 4 bits - 16 values
} address_t;
The class containing the structs

class CACHE;
  cache_t cache;

  function void cache_set(bit [31:0] address, bit [7:0] data);
      address_t cache_address;
      cache_address = address_t'(address);
      cache.cache_sets[cache_address.set].cache_line[cache_address.tag].bytes[cache_address.word] = data;
  endfunction

  function bit [7:0] cache_get(bit [31:0] address);
      address_t cache_address;
      cache_address = address_t'(address);
      return cache.cache_sets[cache_address.set].cache_line[cache_address.tag].bytes[cache_address.word];
  endfunction
endclass
MSI Cache

```verilog
function MSI_STATE msi_next_state(MSI_STATE state, MSI_ACTIVITY activity);
    MSI_STATE next_state;

    case (state)
        M: begin
            if (activity == PrRd) || (activity == PrWr)
                next_state = M;
            else if (activity == BusRd)
                next_state = S;
            else if (activity == BusRdX)
                next_state = I;
            end
        S: begin
            if (activity == PrRd)
                next_state = S;
            else if (activity == PrWr)
                next_state = M;
            else if (activity == BusRd)
                next_state = S;
            else if ((activity == BusRdX) || (activity == BusUpgr))
                next_state = I;
            end
        I: begin
            if (activity == PrRd)
                next_state = S;
            else if (activity == PrWr)
                next_state = M;
            else if ((activity == BusRd) || (activity == BusRdX) || (activity == BusUpgr))
                next_state = I;
            end
        endcase
    return next_state;
endfunction
```
Conclusion

• A scoreboard in the UVM is easy to build and easy to understand
• Use an analysis port strategy with a monitor
• Build an accurate model (can be hard)
• Provide verbose messages in the scoreboard and model
• All source code is available from the authors (rich.edelman@siemens.com)

• Questions?