

# Layering Protocol Verification: A Pragmatic Approach Using UVM

Rahul Chauhan ([rchauhan@broadcom.com](mailto:rchauhan@broadcom.com))

Gurpreet Kaire ([gpsingh@broadcom.com](mailto:gpsingh@broadcom.com))

Broadcom, Inc. San Diego, CA - USA

[www.broadcom.com](http://www.broadcom.com)

Ravindra Ganti ([rkganti@synopsys.com](mailto:rkganti@synopsys.com))

Subhranil Deb ([sdeb@synopsys.com](mailto:sdeb@synopsys.com))



## **I. Introduction**

Communication protocols are modeled as layers, and these layers are often labelled using the popular Open systems interconnection model. For transmission, the information flows from upper layers downstream to lower layers and for reception, the information flows upstream from lower



```

virtual task run_phase (uvm_phase );
  fork
    this.tx_driver  ( );
    this.rx_driver  ( );
  join_none
endtask : run_phase

virtual task tx_driver  ( );
  forever begin

seq_item_port.get_next_item  (llc_frame  )
  // -- Wait Before
  if (llc_frame.wait_before  )

this.llc_frame_rcvd_ev.wait_pttrigger  ( );
  // -- Process frame for transmission
  if (!llc_frame.bypass_model  )
    this.send_frame  ( );
  else
    this.send_corrupt_frame  ( );
  // -- Wait After
  if (llc_frame.wait_after  )
    // -- Clear Existing Event
    if
( this.llc_frame_rcvd_ev.  is_on( ) )
      this.llc_frame_rcvd_  ev.reset  ( );

this.llc_frame_rcvd_ev.wait_pttrigger  ( );
  ...
  seq_item_port.done  ( );
  end
endtask : tx_driver

task send_frame (llc_frame  );
  // -- Convert LLC to MAC
  this.convert_llc2mac  (llc_frame  );
  // -- Send to MAC passthru  - sequence
  this.send_llc2mac  (llc_frame  );
  // -- Selective Enable Mechanism
  // -- Activate Timer for Flow Control
  if (!llc_frame.bypass_model  )
    this.set_timer  (llc_frame  );
endtask : send_frame

```

```

        // Known Bad Tr,      No Wait for
response
    tr.expect_error = 1;
    if(msg_cnt == 2)
        ...
        `uvm_send(tr);
        get_response(rsp);
    end
endtask : body

endclass : app_seq

class app_drv extends uvm_component;
    task main_phase();
        forever begin
            ...
        end
    end
endclass

```

```

//Use the monitor port to create wait
cond i tions for the sequences

//Wait task for waiting on one p          artic u-
lar frame_kind from the monitor ports
task wait_for_frame(frame_kind_e
frame_kind);
    wait_for_frame_event(frame_kind);
    «
    process_frame_for_sequence();
endtask : wait_for_frame

//Implentation port write function i          m-
plantation.

function write_app();
    if (frame.frame_kind == frame_kind)
        «
        emit_frame_event();
endfunction : write_app

function write_llc();
endfunction : write_llc

function write_mac();
endfunction : write_mac

endclass : virtual sqr

```

**Figure 6. Virtual Sequencer Class Code Block**

```

class virtual_seq extends uvm_sequence;

function new();
    get_handle_for_llc_passthru_seq();
    get_handle_for_mac_passthru_seq();
endfunction : new

task body();
    //Start application sequence on a          p-
plication s equencer
app_seq_1.start(p_sequencer.app_sqr);
app_seq_2.start(p_sequencer.app_sqr);

    //Wait for frame response for second
application seq by calling parent
sequencer task
p_sequencer.
wait_for_frame(app_frame_kind);

    // Optionally inject llc error using
pass through sequence in third
application s equence
fork
    llc_passthru_seq.inject_error = 1;
join_none
app_seq_3.start(p_sequencer.app_sqr)

```

## VI. Conclusion

The motivation for this paper is to share the concepts and simple techniques that were implemented and also share the benefits we achieved with the methodology. The techniques described in this paper can be extended to create even more robust and complex test pattern scenarios. The focus of this methodology was to have maximum controllability at every layer of abstraction while still having an automated test flow.

## VII. References

- [1] Synopsys, Inc., Beyond UVM: Creating Truly Reusable Protocol Layering.
- [2] Accellera, Universal Verification Methodol-  
R J \ 8 9 0      8 V H U ¶ V \* X L G H