Translating and Adapting to the "real" world: SerDes Mixed Signal Verification using UVM

Akhila Madhu Kumar, Karl Herterich

Intel Of Canada

Outline

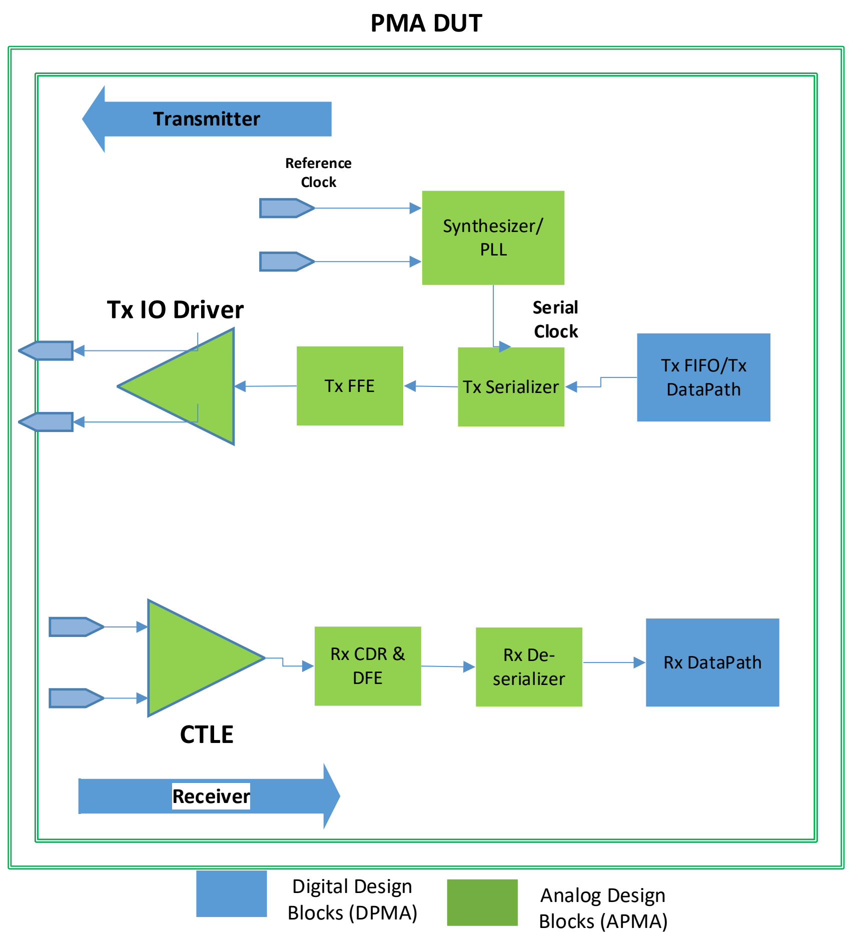
- Basics of SerDes Physical Media Attachment (PMA) layer
- Analog Behavioral model flow
- PMA Testbench architecture
- Translator module
- Pre-emphasis driver modelling
- Layered UVM adapter sequence
- Results
- Conclusion

Basics of SerDes

- SERDES: SERializer - DESerializer
- Used to transmit high speed IO data over a serial link at speeds greater than 2.5Gbps
- Tx: transmits parallel data to high speed receiver serial links by keeping data integrity
- Rx: receives data from serial link, recovers the clock using clock data recovery circuits (CDR) and sends the parallel data to the next-stage
- Tx and Rx data paths can have Built-in-Self-Test (BIST) engines to encode and decode a specific BIST data pattern in the stream and check error injection capability
- The datapath components make sure that the bit error rate (BER) is within the tolerance limit

SerDes PMA Layer

- PMA: Physical Media Attachment Layer
- Primary application of PMA is to transmit the data in Tx and Rx data paths by doing parallel-to-serial and serial-to-parallel conversions respectively.
- The final Rx parallel data to the Physical Coding Sublayer (PCS) should meet the BER(bit-error ratio) requirements, in order to get a proper eye at the receiver end.
- The data received at the Rx side is attenuated due to a lossy channel between transmitter and receiver.
- Continuous Time Linear Equalizer (CTLE) works as a high pass filter to compensate for channel attenuation.



Outline

- Basics of SerDes Physical Media Attachment (PMA) layer
- **Analog Behavioral model flow**
- PMA Testbench architecture
- Translator module
- Pre-emphasis driver modelling
- Layered UVM adapter sequence
- Results
- Conclusion

APMA BMOD Development flow

FAST Behavioral Models

- Netlisted down to major block level. Sub-blocks are modelled behaviorally
- **Faster simulation** performance and supports simulator flows like **Xprop**, hence improving code quality
- Very useful in finding **bugs in APMA<->DPMA interface** and quicker PMA simulation bring-up
- Interface data is of “logic” type

Accurate Behavioral models

- Netlisted down to leaf cell level. Transistor blocks are modelled
- Very close to the **actual schematics** from functionality standpoint
- Used to check schematic functionality and connectivity. Helps in finding **schematic bugs**
- Interface data is of “logic” type

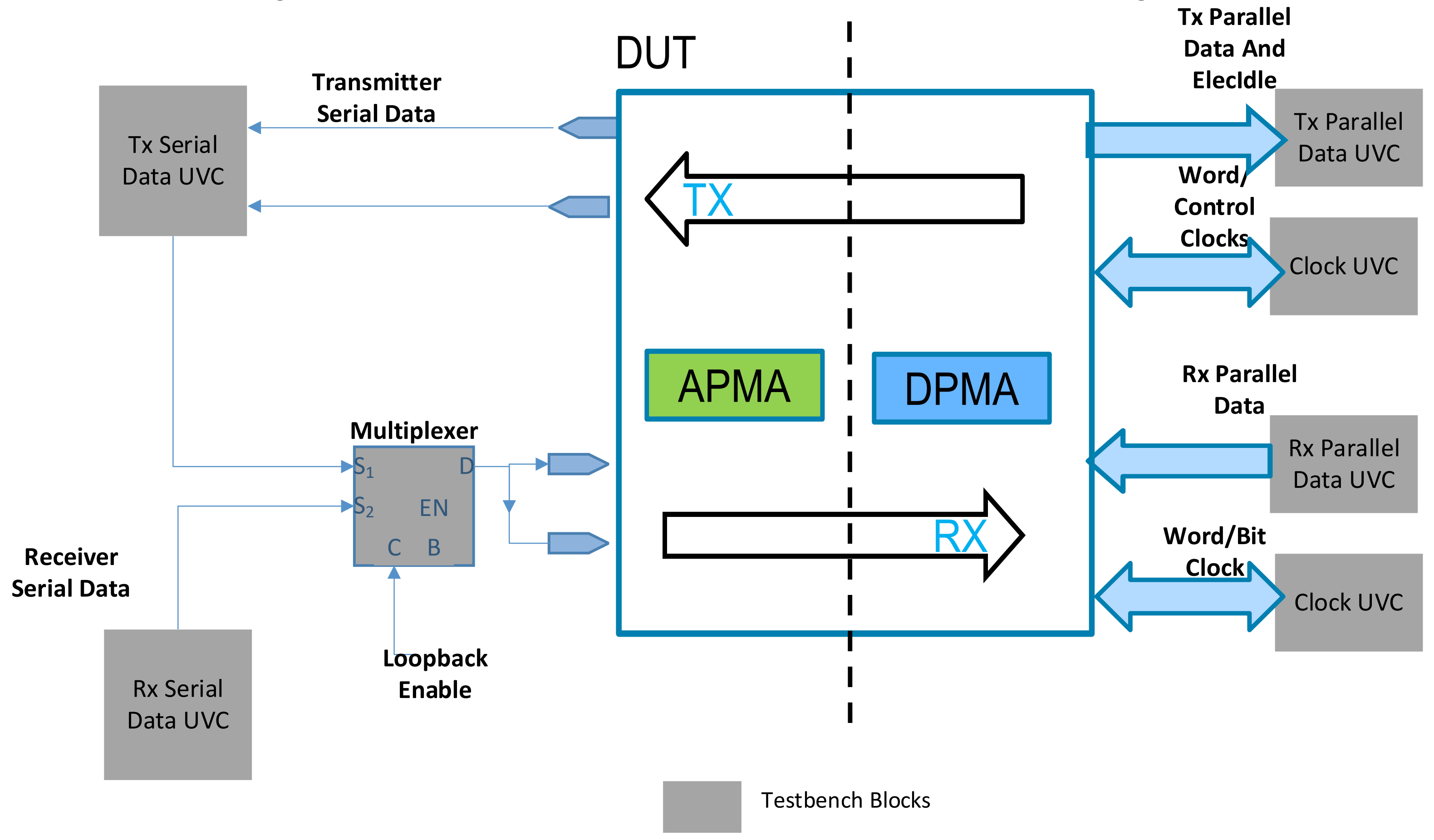
AMS “mode” in Behavioral models

- Based on analog schematic and **system simulation** data to derive ideal models which contain **voltage and current** information
- Verilog model enhanced to capture analog mixed signal behavior using a combination of **“logic” and “real”** signals
- Useful in validating critical features in PMA like **calibrations**, **Rx adaptation and equalization**

Outline

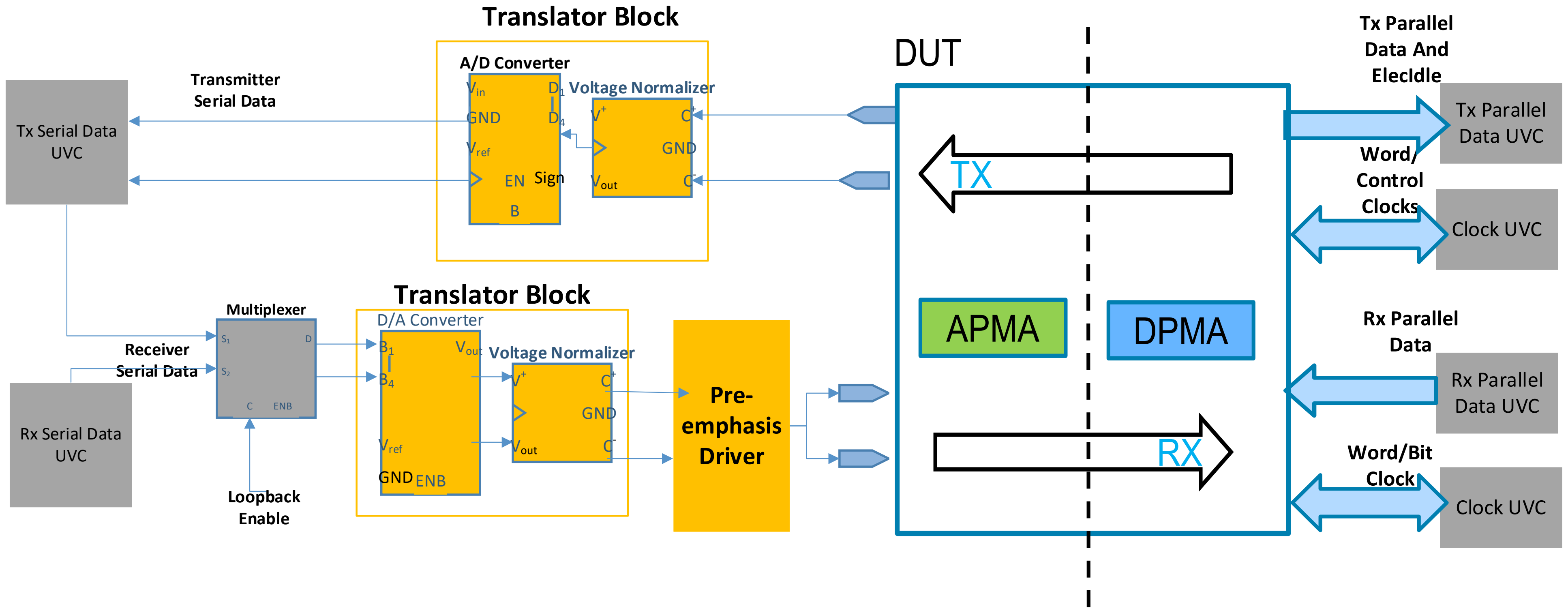
- Basics of SerDes Physical Media Attachment (PMA) layer
- Analog Behavioral model flow
- **PMA Testbench architecture**
- Translator module
- Pre-emphasis driver modelling
- Layered UVM adapter sequence
- Results
- Conclusion

PMA Testbench Architecture (with non-AMS BMOD)



PMA Testbench Architecture (with AMS BMOD)

▨ Testbench Blocks 🟨 Testbench Blocks focused in the paper

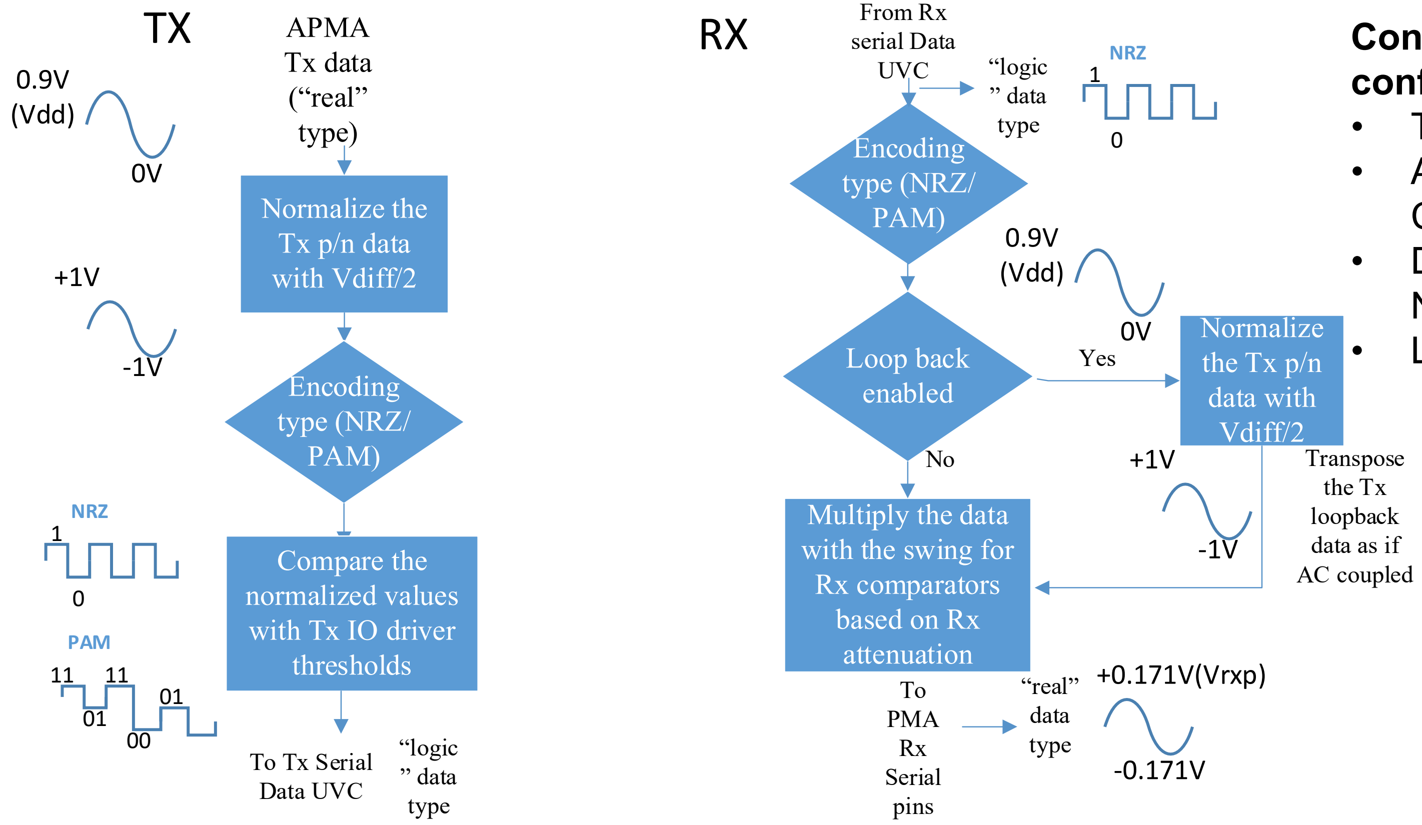


Outline

- Basics of SerDes Physical Media Attachment (PMA) layer
- Analog Behavioral model flow
- PMA Testbench architecture
- **Translator Blocks**
- Pre-emphasis Driver modelling
- Layered UVM adapter sequence
- Results
- Conclusion

Translator Blocks

- Application: Keeps the end-to-end data checkers and pattern generators same across BMODs, with and without AMS “real” mode
- Functionality: To convert the “real” type data to “logic” type and vice versa, taking Tx equalization characteristics and the CTLE and DFE tap gain values into account
- Algorithm:



TX
0.9V (Vdd)
0V
APMA Tx data (“real” type)
Normalize the Tx p/n data with Vdiff/2
+1V
-1V
Encoding type (NRZ/ PAM)
NRZ
1
0
PAM
11
11
01
01
00
Compare the normalized values with Tx IO driver thresholds
To Tx Serial Data UVC
“logic” data type
RX
From Rx serial Data UVC
“logic” data type
NRZ
1
0
Encoding type (NRZ/ PAM)
0.9V (Vdd)
0V
Loop back enabled
Yes
Normalize the Tx p/n data with Vdiff/2
No
+1V
-1V
Transpose the Tx loopback data as if AC coupled
Multiply the data with the swing for Rx comparators based on Rx attenuation
To PMA Rx Serial pins
“real” data type
+0.171V(Vrxp)
-0.171V

Configurability (using UVM configuration database):

- The Tx IO driver voltage levels
- ADC threshold based on Rx CTLE attenuation modelling
- Data encoding type: NRZ/PAM4
- Loop back mode enable

Real<->Logic Conversion Functions

```
function logic2_t pam4Real2Logic (real p_norm, real
m_norm) ;
  logic code[2] ;
  real lev_p_1, lev_p_0, lev_n_1, lev_n_0;
  begin
    if($test$plusargs("PAM_LEV_P_1")) begin
       $value$plusargs("PAM_LEV_P_1=%d",lev_p_1);
    if($test$plusargs("PAM_LEV_P_0")) begin
       $value$plusargs("PAM_LEV_P_0=%d",lev_p_0);
    if($test$plusargs("PAM_LEV_N_1")) begin
       $value$plusargs("PAM_LEV_N_1=%d",lev_n_1);
    if($test$plusargs("PAM_LEV_N_0")) begin
       $value$plusargs("PAM_LEV_N_0=%d",lev_n_0);
    end
      code[0] = (p_norm >  lev_p_1) ? 1'b1 :
                (p_norm > -lev_p_0) ? 1'b1 : 1'b0;
      code[1] = (m_norm >  lev_n_1) ? 1'b1 :
                (m_norm > -lev_n_0) ? 1'bz : 1'b0;
    return code ;
  end
  endfunction
```

```
function logic2_t NRZReal2Logic (real p_norm, real
m_norm, real threshold) ;
logic code[2] ;
  begin
    code[0] = p_norm >= threshold ? 1'b1 : 1'b0;

    code[1] = m_norm >= threshold ? 1'b1 : 1'b0;

    return code ;
  end
  endfunction
```

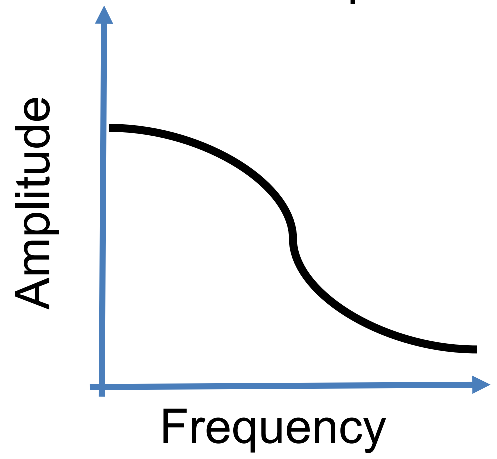
```
function real2_t pam4Logic2Real (logic p, logic n) ;
  real vnorm[2];
  begin
    if      (p===1'b1 && n===1'b1) vnorm = '{1.0,1.0} ;
    else if (p===1'b0 && n===1'b0) vnorm = '{-1.0,-1.0} ;
    else if (p===1'b1 && n===1'b0) vnorm = '{1.0,-1.0} ;
    else if (p===1'b1 && n===1'bz) vnorm = '{1.0/3.0,-1.0/3.0} ;
    else if (p===1'bz && n===1'b0) vnorm = '{1.0/3.0,-1.0/3.0} ;
    else if (p===1'b0 && n===1'bz) vnorm = '{-1.0/3.0,1.0/3.0} ;
    else if (p===1'bz && n===1'b1) vnorm = '{-1.0/3.0,1.0/3.0} ;
    else if (p===1'b0 && n===1'b1) vnorm = '{-1.0,1.0} ;
    else                           vnorm = '{0,0} ;

    return vnorm ;
  end
  endfunction
```

Outline

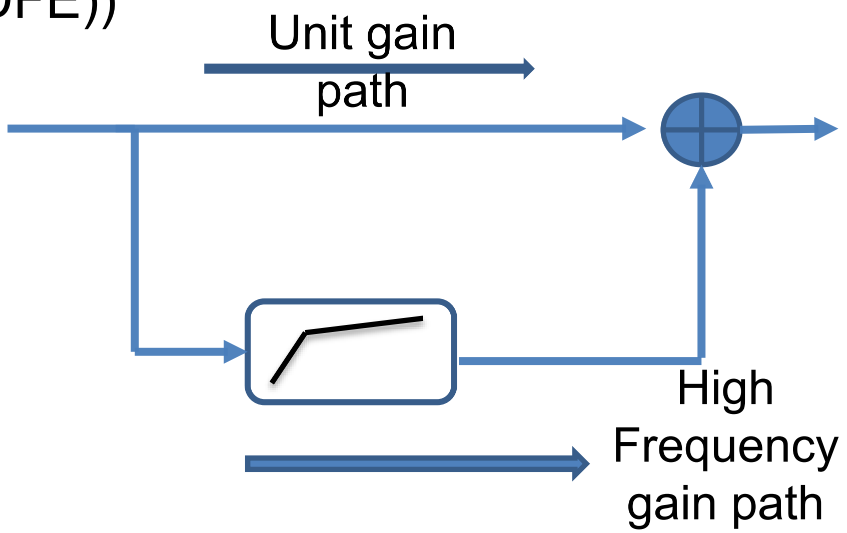
- Basics of SerDes Physical Media Attachment (PMA) layer
- Analog Behavioral model flow
- PMA Testbench architecture
- Translator Blocks
- **Pre-emphasis Driver modelling**
- Layered UVM adapter sequence
- Results
- Conclusion

Emphasis and Equalization

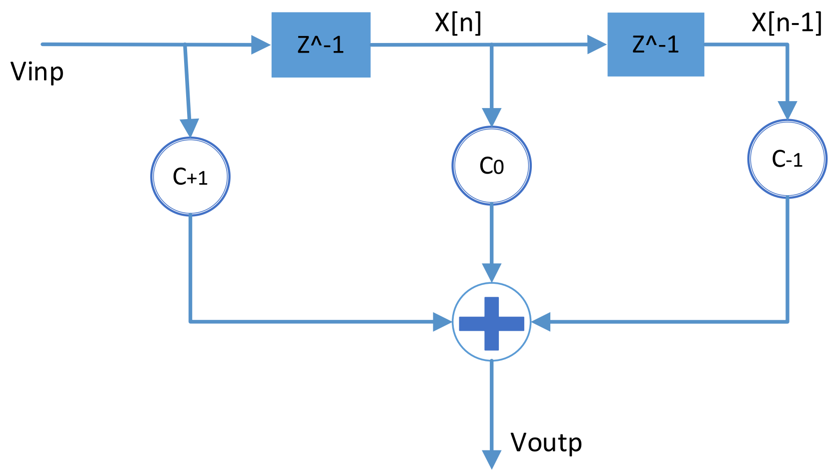
- Data transmission loss can be compensated for at the transmitting and the receiving end. At the transmitter, it can be compensated either by boosting the higher frequency content (pre-emphasis) or by decreasing the low frequency content (de-emphasis).
- Pre-emphasis and equalization are techniques to prevent data loss and invert the channel's frequency response i.e. invert of a low pass filter.
- Ideally implemented in Tx as Feed Forward Equalizer and in Rx as CTLE and Decision Feedback Equalizer (DFE))



Channel Response



CTLE filter



3-Tap FIR Filter

Tx Pre-Emphasized output

- Tx output voltage equation without Pre-emphasis:

  $$V_{outp} = |C_0| * x[n]$$

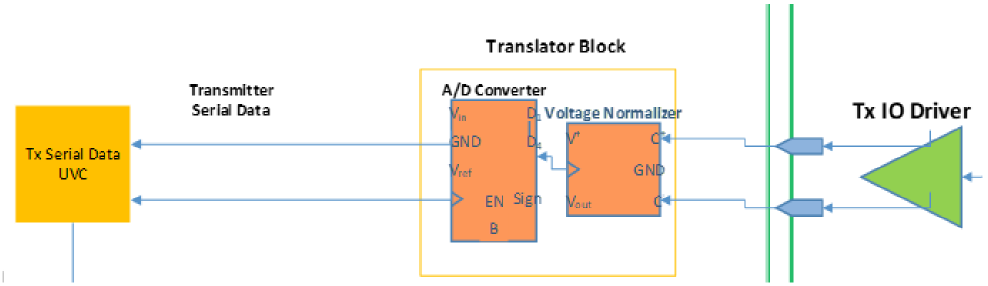
  – Sample output levels-> 0V to ($\mathbf{C_0}$ * Vpp)
- Tx output voltage equation with Pre-emphasis:

  $$V_{outp} = |C_{-1}| * x[n-1] + |C_0| * x[n] + |C_1| * x[n+1]$$

  – Sample output levels-> 0V, ($\mathbf{C_0}$ * Vpp), ($\mathbf{C_0}$ + C_{+1}) * Vpp etc.

X[n-1],X[n], X[n+1]	Pre-emphasized Tx serial output	Normalized Voltage post translator block (with pre-emp mid-threshold)	Final decoded bit value at the Tx transactor
010	$\mathbf{C_0}$	$\mathbf{C_0}$ * Vpp	1
011	$\mathbf{C_0}$ + C_{+1}	($\mathbf{C_0}$ + C_{+1}) * Vpp	1
100	C_{-1}	C_{-1} * Vpp	0
101	C_{-1} + C_{+1}	(C_{-1} + C_{+1}) * Vpp	0

- Thresholds are **input to PMA** and hence testbench configurable
- PCIe spec has set of rules to determine the values of coefficients(C_{-1} C_0 C_{+1}).
- **C_0 is always greater than C_{+1} and C_{-1}**



Pre-emphasis data on Tx handled by Translator block

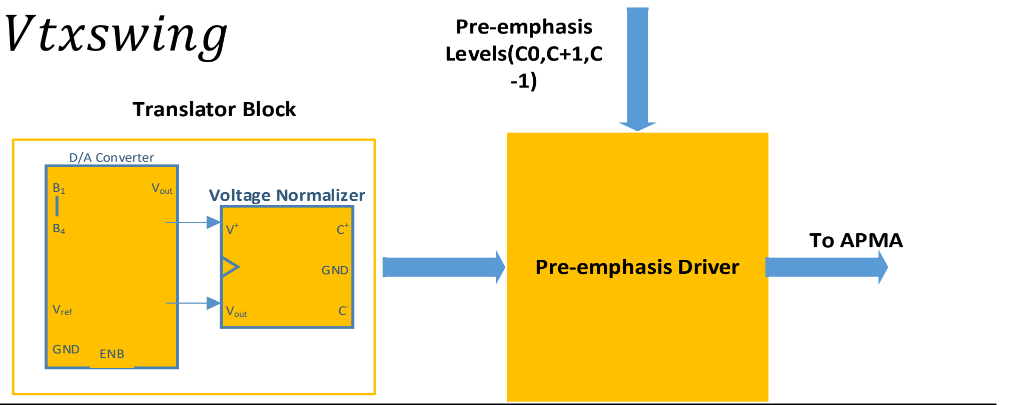
Pre-Emphasis Driver on Rx path

- Rx translator output: For a Tx voltage swing of $V_{txswing}$, pre-emphasized Tx output voltage value of t_{xp} and desired Rx voltage of V_{rxp} , the translator module output can be derived as:

$$rx_{xlator_p} = \frac{(tx_p - Vtxswing)}{Vtxswing} * Vrx_p$$

- Pre-emphasis driver output: Run a reverse function of scaling and normalizing done in the Rx translator block(to cancel the translation done) and output 1 if the results is a factor of C_0

$$rx_{premph_p} = \frac{rx_{xlator_p}}{Vrx_p} * Vtxswing + Vtxswing$$



Pre-emphasis data on Rx handled by Pre-emphasis driver

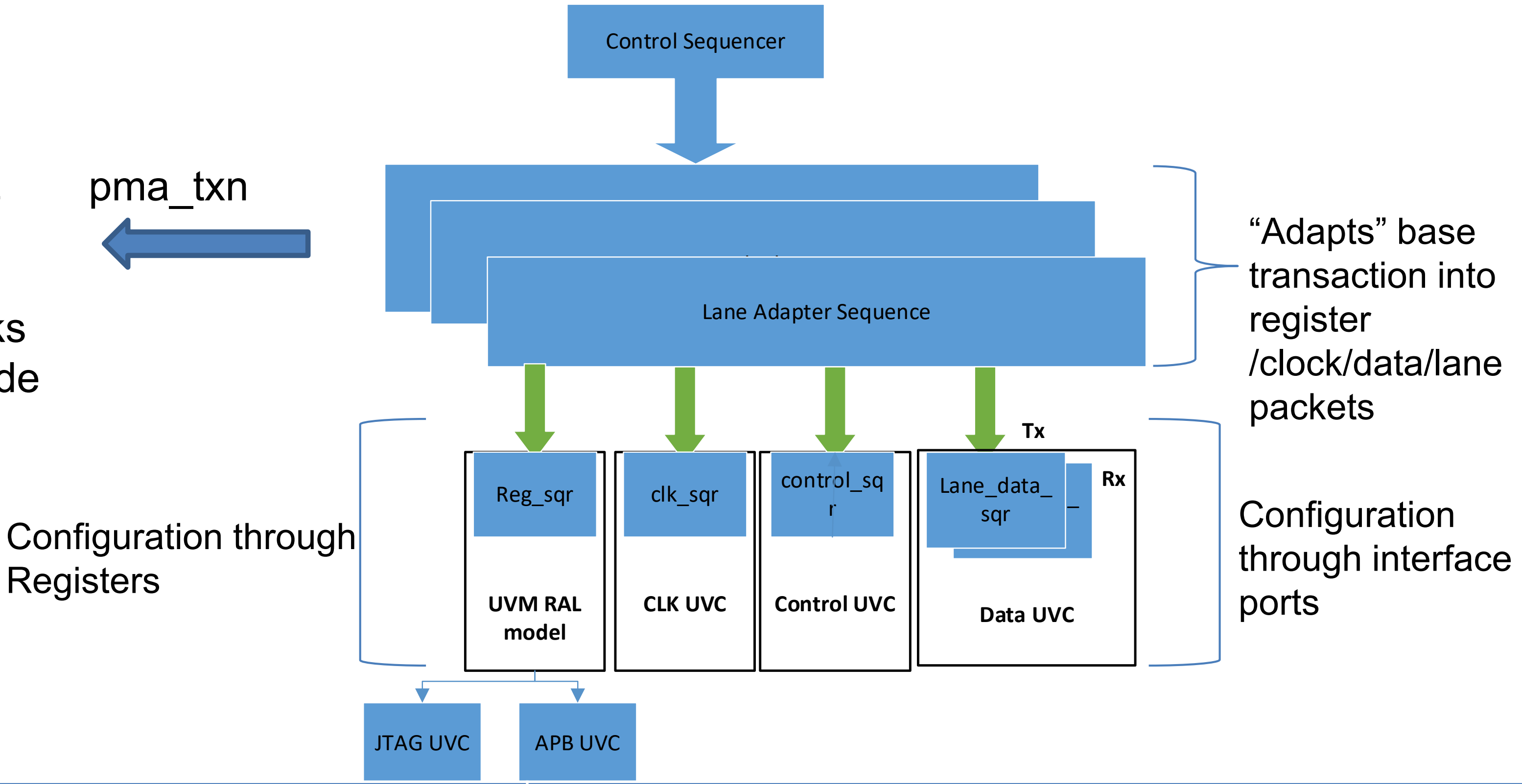
Outline

- Basics of SerDes Physical Media Attachment (PMA) layer
- Analog Behavioral model flow
- PMA Testbench architecture
- Translator Blocks
- Pre-emphasis Driver modelling
- **Layered UVM adapter sequence**
- Results
- Conclusion

UVM Adapter Sequence Architecture

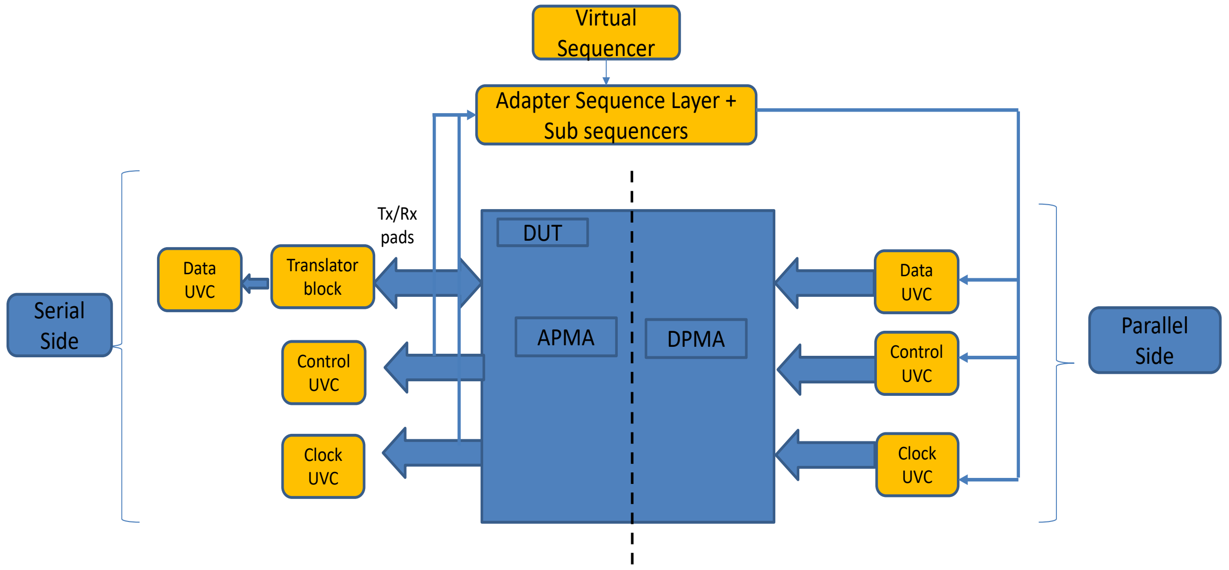
PMA Scoreboard

- Data integrity checks
- Beacon checks
- Loopback mode checks
- Transaction integrity

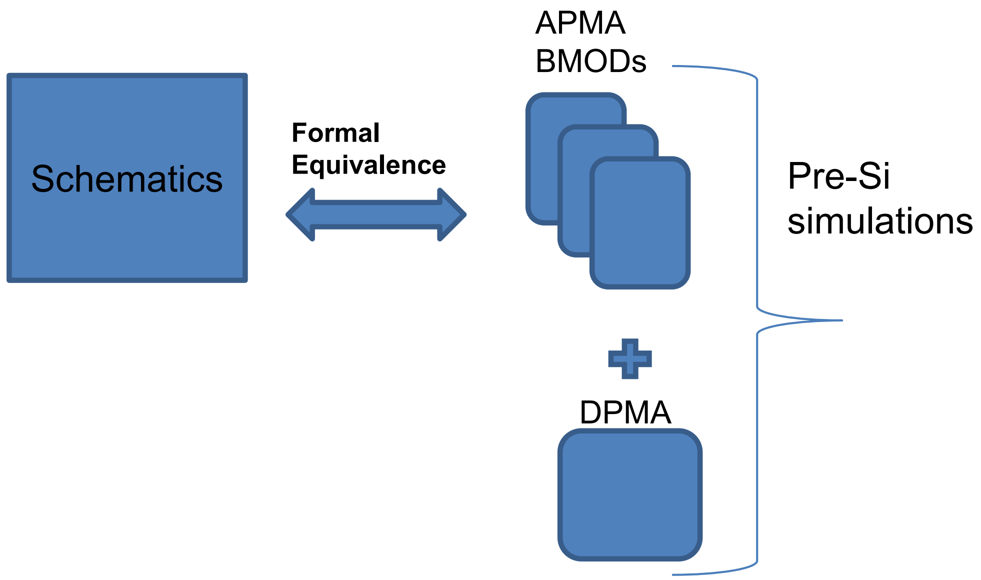


Features of Adapter Sequence	Remarks
Lane specific traffic generation and routing	Sending traffic across lanes, wait for CDR lock and checking lane-to-lane interactions (skew, latency etc.)
Demarcation between control, data and debug transfers	Setting up beacon, sending burst, checking clocks or bit errors
Centralized coverage sampling across model types	Coverage packet can be sent to coverage class or can be sampled here directly for control and data transfers
Passing packet to scoreboard/monitors	Each transaction from register or control sequencer comes to lane adapter and hence the packet contents can be used to add checks in scoreboard or monitor specification adherence

High Level TB Architecture for PMA



APMA Behavioral Model (BMOD) Usage Flow

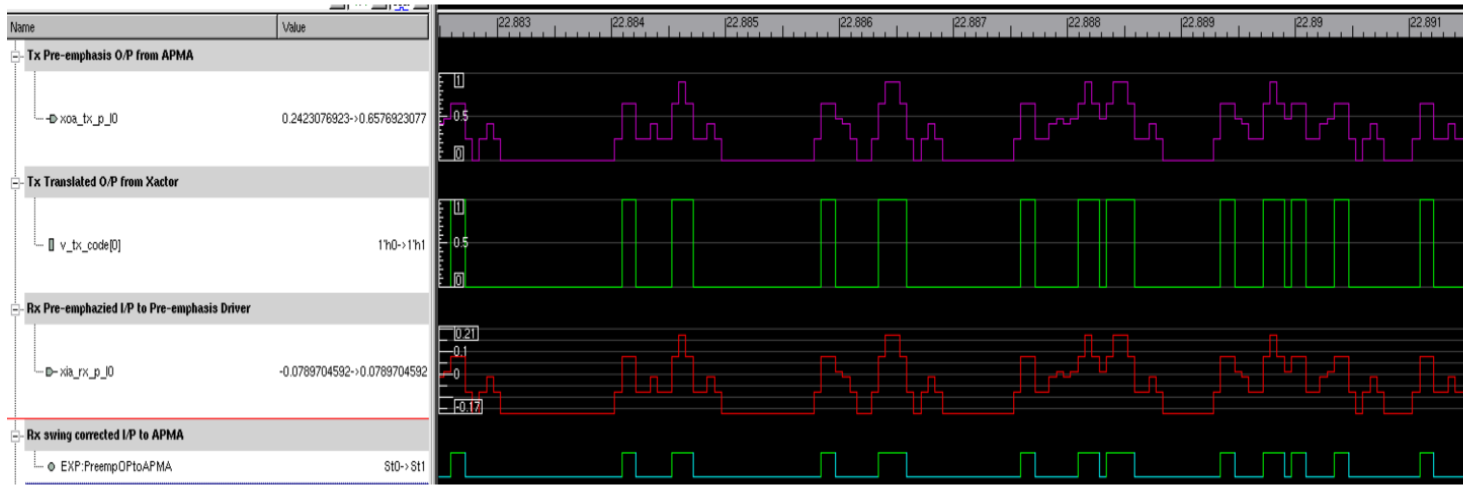


Outline

- Basics of SerDes Physical Media Attachment (PMA) layer
- Analog Behavioral model flow
- PMA Testbench architecture
- Translator Blocks
- Pre-emphasis Driver modelling
- Layered UVM adapter sequence
- **Results**
- Conclusion

Results – Pre-emphasized Data With Loopback

- The first group is the **pre-emphasized output** from APMA Tx serial pads
- The second group shows the **Tx translated output** from transactor which predicts the 0s and 1s correctly
- Third group shows the **Rx translated output** which uses the Tx pad data and scales based on translator Rx path algorithm
- Fourth group shows the final **output from pre-emphasis driver** which corrects the swing based on CTLE requirements (0 translated to -0.171 and 1 translated to 0.171 in the example shown)

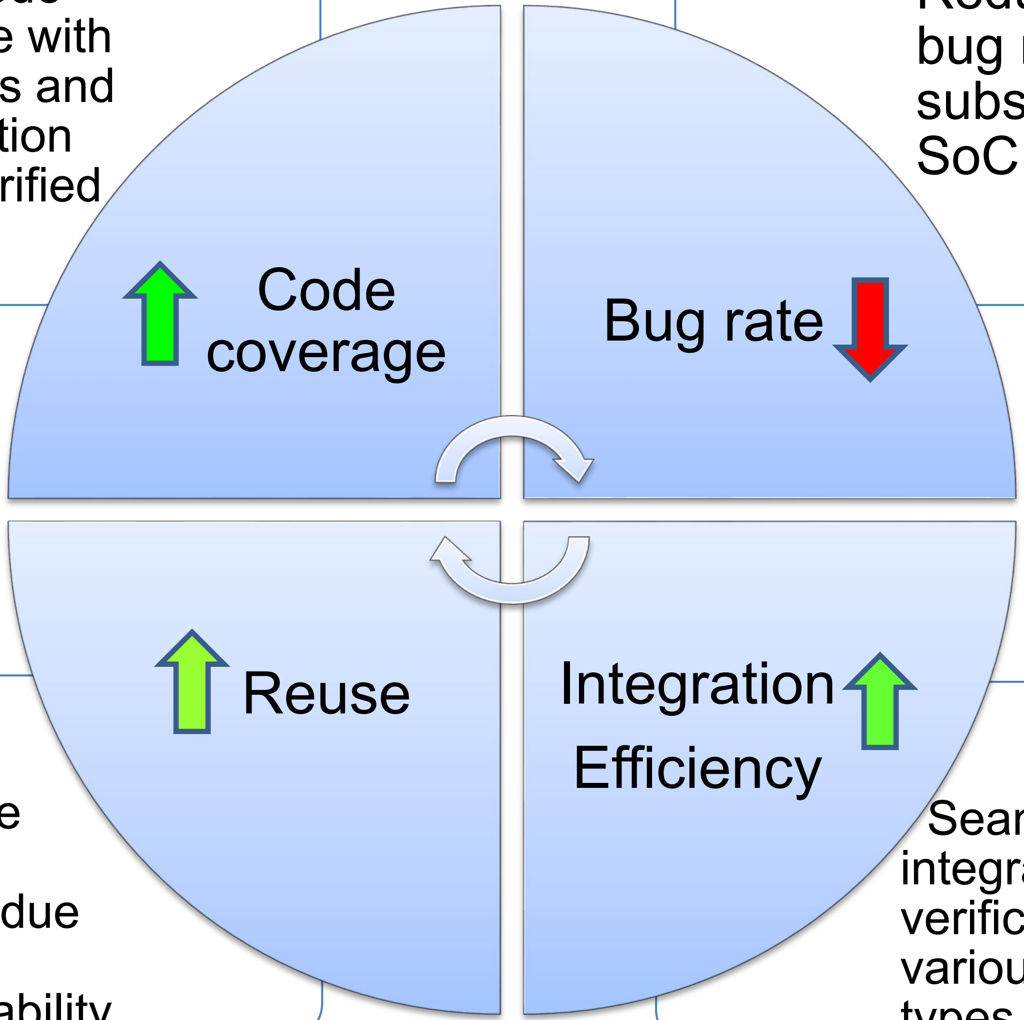


Name	Value
Tx Pre-emphasis O/P from APMA	
xoa_tx_p_l0	0.2423076923->0.6576923077
Tx Translated O/P from Xactor	
v_tx_code[0]	1'h0->1'h1
Rx Pre-emphazied I/P to Pre-emphasis Driver	
xia_rx_p_l0	-0.0789704592->0.0789704592
Rx swing corrected I/P to APMA	
EXP:PreempOPtoAPMA	St0->St1

Conclusion

- Better code coverage with emphasis and equalization being verified properly.

- Reduced bug rate at subsystem/ SoC level



- Reusable across projects due to high configurability

Seamless integration and verification of various BMOD types

MIN
MAX
CREATIVITY
MIN
MAX
EFFICIENCY
MIN
MAX
QUALITY

Thank You!

Questions ?

Backup

Future Work

- Mixed signal simulations with sub-modules replaced by schematics
- Extensive use of MSV control knobs present as part of APMA models
- Bring-in channel models along with pre-emphasis driver and fine tune the settings accordingly

Fast And Accurate BMOD Examples

```
module apma_rx_clkgen_FAST (
output  ock_ckd_n,
output  ock_ckd_p,
output  ock_divck_n,
output  ock_divck_p,
input  ick_n,
input  ick_p,
input  pa_vss,
input  vdd
);
reg state, stateb;
....
assign ock_divck_n = ck_div1_n;
assign ock_divck_p = ck_div1_p;
assign pd_clk_n = ~pd_clk_p;
...
endmodule
```

```
module apma_rx_clkgen (
output  ock_ckd_n,
output  ock_ckd_p,
output  ock_divck_n,
output  ock_divck_p,
input  ick_n,
input  ick_p,
input  pa_vss,
input  vdd
);
reg state, stateb;
....

bfr I0 ( .qn(ock_divck_n), .qp(ock_divck_p),
     .dn(ck_div1_n), .dp(ck_div1_p),
     .pa_vss(pa_vss), .vdd(vdd));
dff_2x1 I1 ( .q(state), .clk(pd_clk_p), .d(stateb),
     .pa_vss(pa_vss), .vdd(vdd));
inv2_1x1 I2 ( .z(pd_clk_n), .a(pd_clk_p), .pa_vss(pa_vss),
     .vdd(vdd));
....
endmodule
```

UVM Adapter Sequence Snippet

```
class pma_lane_adapter_seq extends uvm_sequence#(pma_transaction);
  pma_transaction pma_req;
  `uvm_object_utils(pma_lane_adapter_seq)
  `uvm_declare_p_sequencer(pma_lane_sequencer)
  // All Register Sequence's object created
  pma_init_state_reg_seq          init_state_reg_seq;
  pma_set_spec_reg_seq            set_spec_reg_seq;
  pma_configure_lane_reg_seq      configure_lane_reg_seq;
  virtual task body();
    forever begin
      p_sequencer.get_next_item(pma_req);
      if (pma_req.primitive_layer == pma_transaction::CTRL) begin
        drive_ctrl();
      end else if (pma_req.primitive_layer == pma_transaction::DATA) begin
        drive_data();
      end else begin
        drive_debug();
      end
      p_sequencer.item_done();
    end
  endtask : body
```

UVM Adapter Sequence Snippet (contd.)

```
task drive_ctrl();
    case (pma_req.primitive_name)
      pma_transaction::FORCE_PUP :
        begin
          if (pma_req.pma_cfg.ctrl_path == UVM_BACKDOOR) begin
            $cast(req_copy, pma_req.clone());
            req_copy.set_item_context(this, p_sequencer.cmn_ctrl_seqr);
            start_item(req_copy);
            finish_item(req_copy);
           return_response = 0;
          end else begin
            `uvm_do_on_with(force_pup_reg_seq,
             p_sequencer.reg_seqr, { block == pma_req.primitive_target; enable ==
pma_req.power_on; });

          end
        end
endtask
```

Configuration through interface ports

Configuration through Registers

Translator Voltage Scaling

- Translator on Tx Path scales the APMA output and sends to the transactors

Tx NRZ Output from APMA	Normalized Output(with Vpp/2)	Comparator Output (threshold is 0V)
0.9V (Vpp)	1V	1
0V	-1V	0

- On Rx path: without loopback
  - "logic" data is directly sent from Rx Serial UVC
  - This data is scaled before sending to APMA

Rx NRZ Input from Data UVC	Scaled output	CTLE Comparator Output (threshold is 0V)
1	0.171	1
0	-0.171	0

- Rx Path: with loopback
  - Tx data is normalized with Vpp/2
  - This output is scaled before sending to APMA

Tx NRZ Output from APMA	Normalized Output(with Vpp/2)	Scaled output	CTLE Comparator Output (threshold is 0V)
0.9V (Vpp)	1V	0.171	1
0V	-1V	-0.171	0

Conclusion

- Building a configurable translator module and the UVM layered structure can help reuse the blocks across IP variants and also across various analog models
- Using the adapter sequences rendered creating a vast set of sequence library for each data and lane configurations as per the specification
- This flow can render build a verification framework which models critical design features effectively and helps in focused verification of the same

Layered UVM Adapter Sequence

Features of Adapter Sequence	Remarks
Lane specific traffic generation and routing	Sending traffic across lanes, wait for CDR lock and checking lane-to-lane interactions (skew, latency etc.)
Demarcation between control, data and debug transfers	Setting up beacon, sending burst, checking clocks or bit errors
Centralized coverage sampling across model types	Coverage packet can be sent to coverage class or can be sampled here directly for control and data transfers
Passing packet to scoreboard/monitors	Each transaction from register or control sequencer comes to lane adapter and hence the packet contents can be used to add checks in scoreboard or monitor specification adherence

- A central layer for transaction ordering and control; helps in generating effective lane specific traffic
- Each transaction takes in the direction (Tx or Rx) as well as the lane number as a parameter which can be set from the test specific virtual sequence
- Lane-to-lane skew and elecidle entry/exit controlled for each lane