A Robust Framework for High Speed Link Verification with a Proficient Way to Close the Functional Coverage

Piyush Tankwal, Arnab Ghosh, Piyush Agnihotri, Mukesh Gandhi, Parag S Lonkar

Samsung Semiconductor India R&D Center

Introduction

- The first MIPI PHY standard, D-PHY, was introduced in 2005 and can be found on most mobile device. D-PHY is commonly used to connect an applications processor to a camera using the MIPI High speed Camera Serial Interface (CSI). It is also used to connect the applications processor to the mobile device's display using the MIPI High speed Display Serial Interface (DSI).

- D-PHY's reach has not extended beyond the camera and display due to limitation on the transmission rate to 1.5 Gbps. This was not seen as a limitation several years ago when the standard was conceived, but is simply too slow to handle the data traffic in today's smartphones and tablets.

- To address the speed limitation, the M-PHY with a serial output, low pin count, multiple transmission and power-saving modes is released. The latest M-PHY5.0 supports data rates from 10kbps up to 23.32 Gbps per lane per direction.

- MPHY data rates provides enough bandwidth to support all types of high speed data interfaces within a mobile device. Interfaces employing M-PHY now include several MIPI standards (CSI-3, DSI-2, LLI, UniPro, and DigRF), the USB SSIC standard, the Universal Flash Storage (UFS) standard, and the M-PCIe standard.

- For showing our proposed work, We are taking MIPI UniPro (High Speed Link) as a case study which is act as interconnect layer for Universal Flash storage(UFS) as shown in Figure 2.

- Newly released MIPI UniPro 2.0 uses MPHY 5.0, HS-G5 to increase bandwidth to 23.32 Gbps per lane and per direction to satisfy the memory storage ecosystem’s growing data rates requirements.

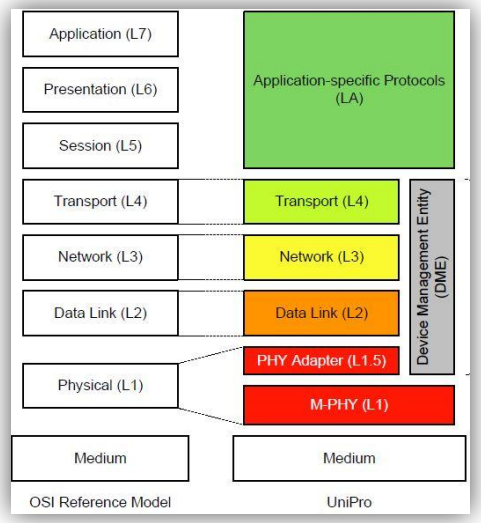


Figure 1. Comparative Layered Architecture of OSI Model and MIPI UniPro (High Speed Link).

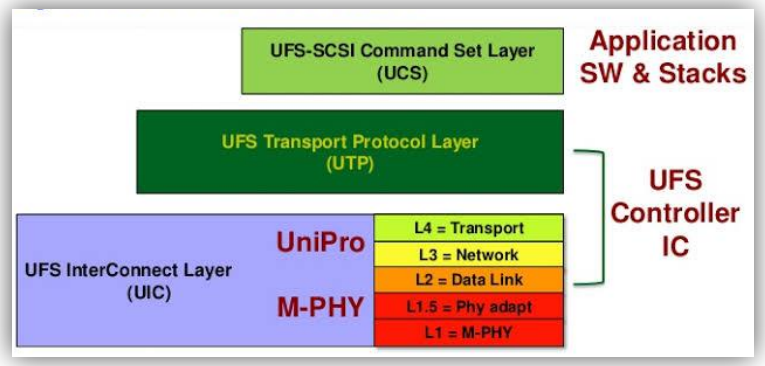


Figure 2. Layered Architecture of UFS and MIPI UniPro (High Speed Link).

Motivation

Skew

- Routing delay introduce the skew in digital circuits that impact the functionality of design.

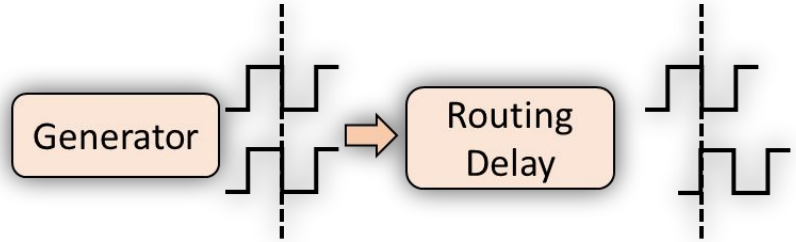
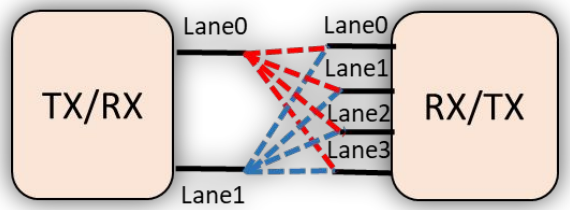


Figure 3. (a) Skew in digital circuit

Lane Mapping

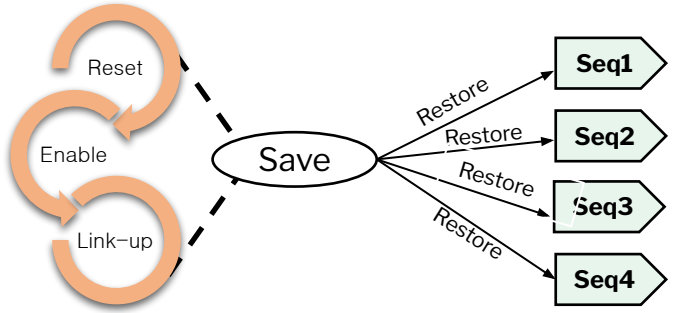
- Design can have unconnected lanes, cross connected lanes.



(b) Lane Mapping

Save and Restore

- In Link level design initialization itself can take 50-60 % of the simulation time that delays the verification process.



(c) Initialization Sequence Save and Re-store

Coverage Closure

- Functional Coverage closure is a essential part for a any complex design verification and that generally takes high bandwidth.

(d) Coverage Closure

Skew Insertion

- Inter-lane skew is a common phenomenon at the serial lanes and standards are mentioned in the UniPro specification.
- Injected skew at serial lines in UniPro full stack test bench and at RMMI interface in UniPro standalone test bench.
- This skew is passed to the upper layer and it is responsibility of the Link layer to merge the data correctly
- Skew injection at High Speed Link-PHY interface is required considering all possible scenario and adequate randomization for proper verification sign-off

Table I. Max allowed Lane to Lane Skew. (Reference:UniPro2.0 Link-PHY interface)

Gear	Max Lane to Lane Skew
PWM G1	30 UI
HS G1	30 UI
HS G2	30 UI
HS G3	30 UI
HS G4	60UI
HS G5	120 UI

```
rand int SKEW0;
rand int SKEW1;
constraint lane0_skew  {if(mode==LS) SKEW0 inside {[0:30]};
                        else if (mode==HS && (gear==G1||
                        gear==G2 || gear==G3)
                        SKEW0 inside {[0:30]};
                        else if (mode==HS && (gear==G4)
                        SKEW0 inside {[0:60]};
                        else (mode==HS && (gear==G5)
                        SKEW0 inside {[0:120]};}

constraint lane1_skew  {if(mode==LS) SKEW1 inside {[0:30]};
                        else if (mode==HS && (gear==G1||
                        gear==G2 || gear==G3)
                        SKEW1 inside {[0:30]};
                        else if (mode==HS && (gear==G4)
                        SKEW1 inside {[0:60]};
                        else (mode==HS && (gear==G5)
                        SKEW1 inside {[0:120]};}
```

Figure 4.(a) Constraints on Skew Generation (Reference: UniPro2.0 Link-PHY interface).

```
wait(pwrmode_ind) //power mode change done
if (mode==HS) begin //mode is high speed
  if (gear==G1 && Series==A) T_UI=`HSG1A_T_UI;
  else if (gear==G1 && Series==B) T_UI=`HSG1B_T_UI;
  else if (gear==G2 && Series==A) T_UI=`HSG2A_T_UI;
  else if (gear==G2 && Series==B) T_UI=`HSG2B_T_UI;
  else if (gear==G3 && Series==A) T_UI=`HSG3A_T_UI;
  else if (gear==G3 && Series==B) T_UI=`HSG3B_T_UI;
  else if (gear==G4 && Series==A) T_UI=`HSG4A_T_UI;
  else if (gear==G4 && Series==B) T_UI=`HSG4B_T_UI;
  else if (gear==G5 && Series==A) T_UI=`HSG5A_T_UI;
  else if (gear==G5 && Series==B) T_UI=`HSG5B_T_UI;
end
else T_UI=`PWM_T_UI; //mode is slow

if(MAX_SKEW0==1 && MAX_SKEW1==0) begin //max skew lane 0
  if((mode==HS && (gear==G1 ||gear==G2||gear==G3)) ||(mode==LS
    && (gear==G1)))
    SKEW0=30;
  else if (mode==HS && (gear==G4))
    SKEW0=60;
  else if (mode==HS && (gear==G5))
    SKEW0=120;
end //}
if(MAX_SKEW1==1 && MAX_SKEW0==0) begin //{ //max skew lane 1
  if((mode==HS && (gear==G1 ||gear==G2||gear==G3)) ||(mode==LS
    && (gear==G1)))
    SKEW1=30;
  else if (mode==HS && (gear==G4))
    SKEW1=60;
  else if (mode==HS && (gear==G5))
    SKEW1=120;
end//}

SKEW_LANE0=SKEW0*T_UI;
SKEW_LANE1=SKEW1*T_UI;

//LANE0
if (SKEW_ENABLE==1 && (MAX_SKEW0 == 1 || RAND_SKEW == 1))begin //{
  //Skew Enabled - Random Skew or Max Skew on Lane 0
  assign #(SKEW_LANE0) rx_symbolclk0=rx_symbolclk[0];
  assign #(SKEW_LANE0) rx_symbol0=rx_symbol[0];
  assign #(SKEW_LANE0) rx_burst0=rx_burst[0];
  assign #(SKEW_LANE0) rx_cfgrdyn0=rx_cfgrdyn[0];
end //}
else begin //{
  //Skew Disabled or Max Skew on Lane 1
  assign  rx_symbolclk0=rx_symbolclk[0];
  assign  rx_symbol0=rx_symbol[0];
  assign  rx_burst0=rx_burst[0];
  assign  rx_cfgrdyn0=rx_cfgrdyn[0];
end //}

//LANE1
if (SKEW_ENABLE==1 && (MAX_SKEW1 == 1 || RAND_SKEW == 1))begin //{
  //Skew Enabled - Random Skew or Max Skew on Lane 1
  assign #(SKEW_LANE1) rx_symbolclk1=rx_symbolclk[1];
  assign #(SKEW_LANE1) rx_symbol1=rx_symbol[1];
  assign #(SKEW_LANE1) rx_burst1=rx_burst[1];
  assign #(SKEW_LANE1) rx_cfgrdyn1=rx_cfgrdyn[1];
end //}
else begin //{
  //Skew Disabled or Max Skew on Lane 0
  assign  rx_symbolclk1=rx_symbolclk[1];
  assign  rx_symbol1=rx_symbol[1];
  assign  rx_burst1=rx_burst[1];
  assign  rx_cfgrdyn1=rx_cfgrdyn[1];
end //}
```

Assigning TUI of selected modes gears/series

Max Skew

Inserting Skew in Lane0

Inserting Skew in Lane1

Figure 4.(b) Code for generating the skew.(Reference: UniPro2.0 Link-PHY interface)

Skew Insertion

- M-PHY 5.0 supports up to 120UI of inter-lane skew at serial lanes for HS-G5 that translates to 6 PA symbols skew at RMMI interface which is bounded by L15 layer of UniPro2.0.

- Here, We are showing injected maximum skew for HS-Gear5; HS-Gear4; HS-Gear3 at RMMI Interface in Figure 5(a), 5(b) and 5(c)

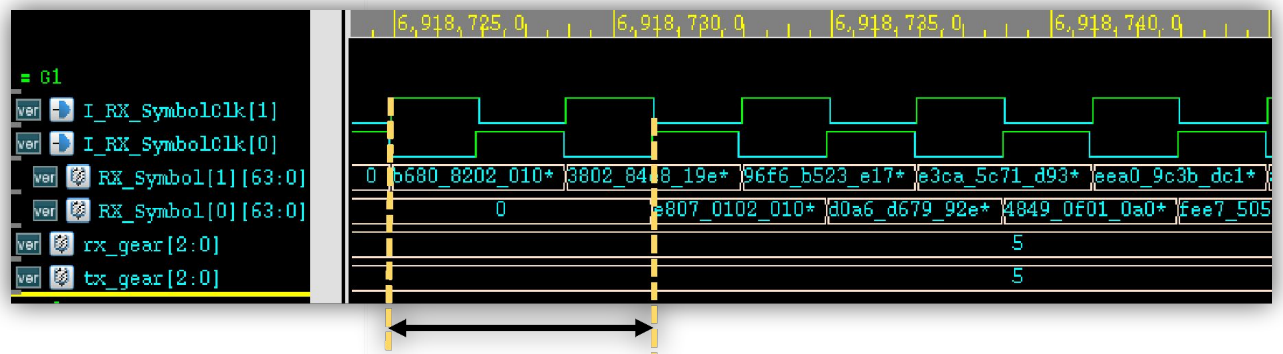


Figure 5. (a) Skew insertion between Lane0 and Lane1 at RMMI for HS-G5 in Rx symbol.(Reference: UniPro2.0 Link-PHY interface).

Figure 5. (b) Skew insertion between Lane0 and Lane1 at RMMI for HS-G4 in Rx symbol.(Reference: UniPro2.0 Link-PHY interface).

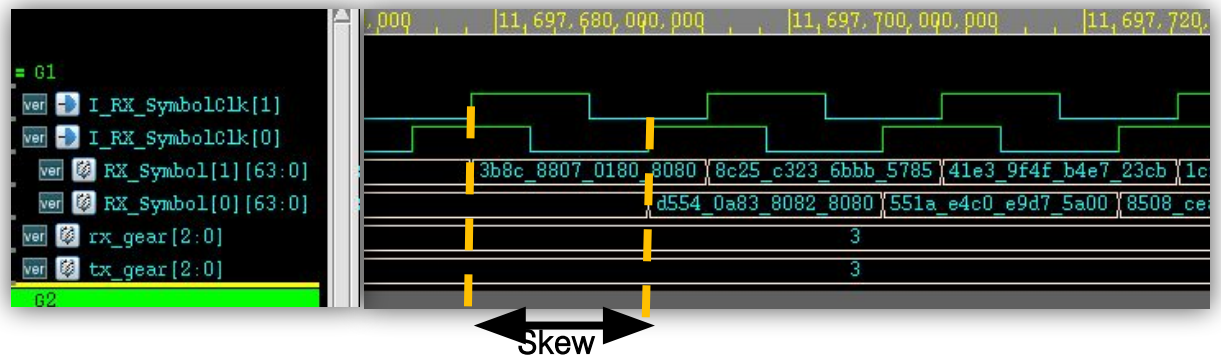


Figure 5. (c) Skew insertion between Lane0 and Lane1 at RMMI for HS-G3 in Rx symbol.(Reference: UniPro2.0 Link-PHY interface).

Lane Mapping

- The Lane Mapping functionality is used to discover the usable physical lanes on the link and there connection with respect to peer device.
- Often, testing is limited to fixed connections or in some cases suiting only the application or use cases, but in a hardcoded way.
- Random lane connection framework provides flexibility to connect DUT lanes to any random number of lanes of peer device and in any random way suiting specification requirement.
- DUT Lanes can be unconnected, one-lane or two lane straight connection, and one-lane or two lane cross connection.
- In Figure 6. “vip_tx_lane0/1==4” is reflecting the specific lane is unconnected.

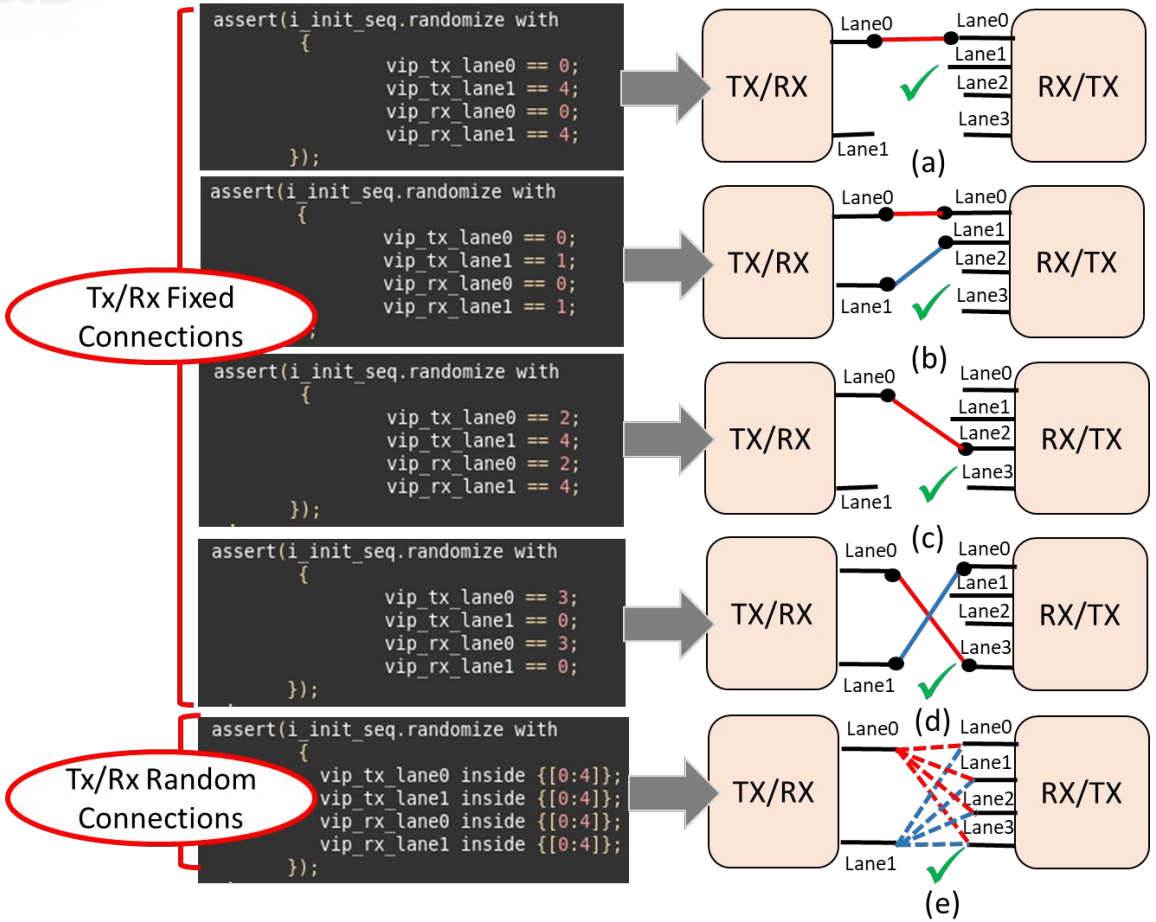


Figure 6. Random lane connection example for UniPro 2.0 DUT (2 Lane) and Peer Device (4 Lane) (a) Single-Lane Straight Connection (b) Single-Lane Cross Connection (c) Dual-Lane Straight Connection (d) Dual-Lane Cross Connection (e) Random Lanes Connection

Lane Mapping

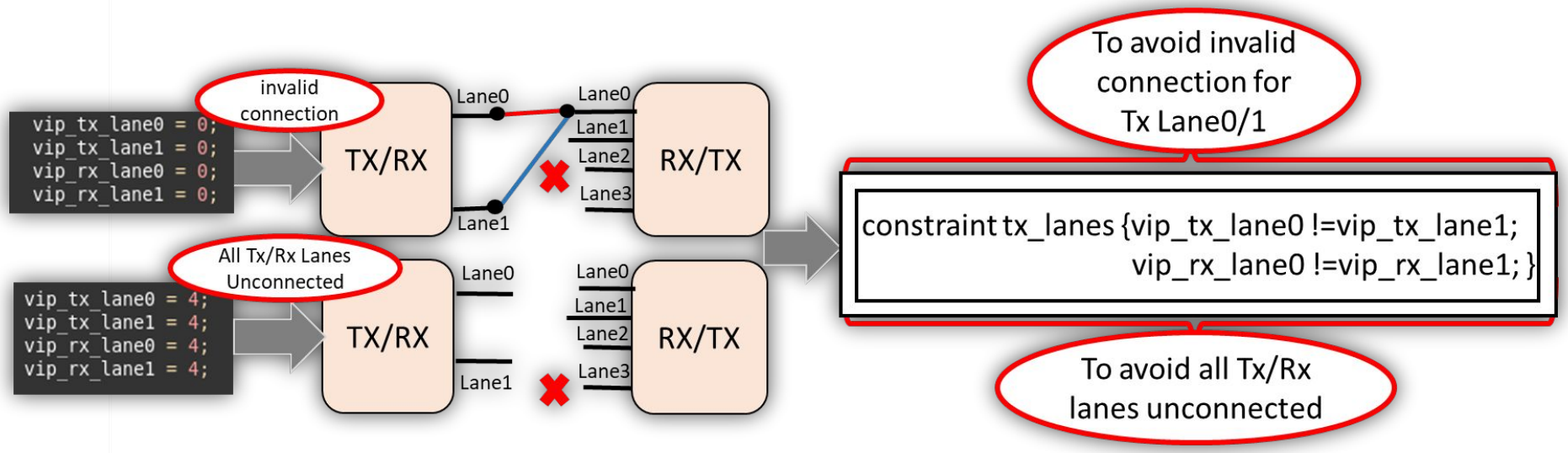


Figure 7. Code to avoid invalid or unconnected lane connection (Reference: UniPro2.0 Link-PHY interface) .

```
//Normal Connections
assign WITH_VIP_LANE0_TX_P = (vip_tx_lane0==0)? txdp0:((vip_tx_lane0==1)? txdp1:((vip_tx_lane0==2)? txdp2:((vip_tx_lane0==3)? txdp3:'hz')));
assign WITH_VIP_LANE1_TX_P = (vip_tx_lane1==0)? txdp0:((vip_tx_lane1==1)? txdp1:((vip_tx_lane1==2)? txdp2:((vip_tx_lane1==3)? txdp3:'hz')));
assign WITH_VIP_LANE0_TX_N = (vip_tx_lane0==0)? txdn0:((vip_tx_lane0==1)? txdn1:((vip_tx_lane0==2)? txdn2:((vip_tx_lane0==3)? txdn3:'hz')));
assign WITH_VIP_LANE1_TX_N = (vip_tx_lane0==0)? txdn0:((vip_tx_lane0==1)? txdn1:((vip_tx_lane0==2)? txdn2:((vip_tx_lane0==3)? txdn3:'hz')));
```

Figure 8. MUX logic for random lane connections from UniPro2.0 Test bench.

Save and Re-store

- In High Speed Link, initialization itself can take 50-60% of the simulation time that delays the verification process. So, to avoid that we can save the repetitive process of Link Up (Link Enable and Training) and re-store it when required.
- Tool option need to be add at elaboration time then $save is available for use in Verilog code through which the snapshot can be saved. User can call $save after the completion of the process which user want to save as shown in Figure 11 (a) & (b).
- Once the snapshot is saved after sim finished, you can find it in your sim directory with same name which is passed through "$save("name ")".

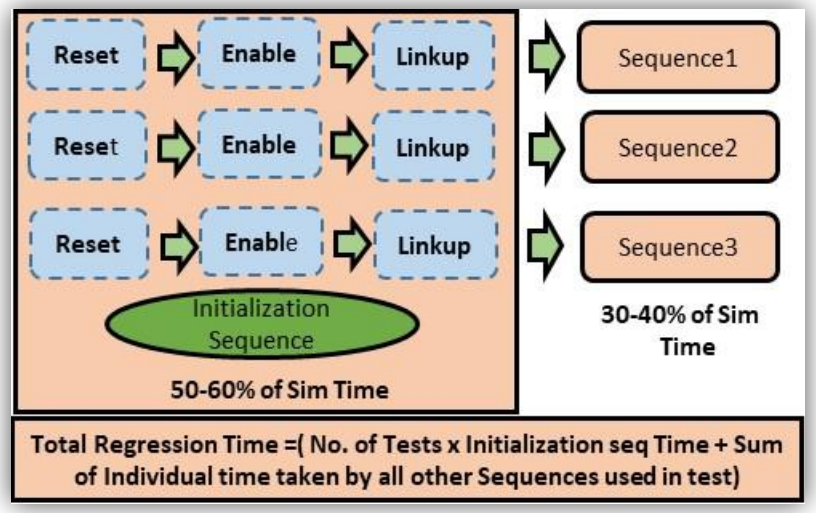


Figure 9. Without Save & Restore

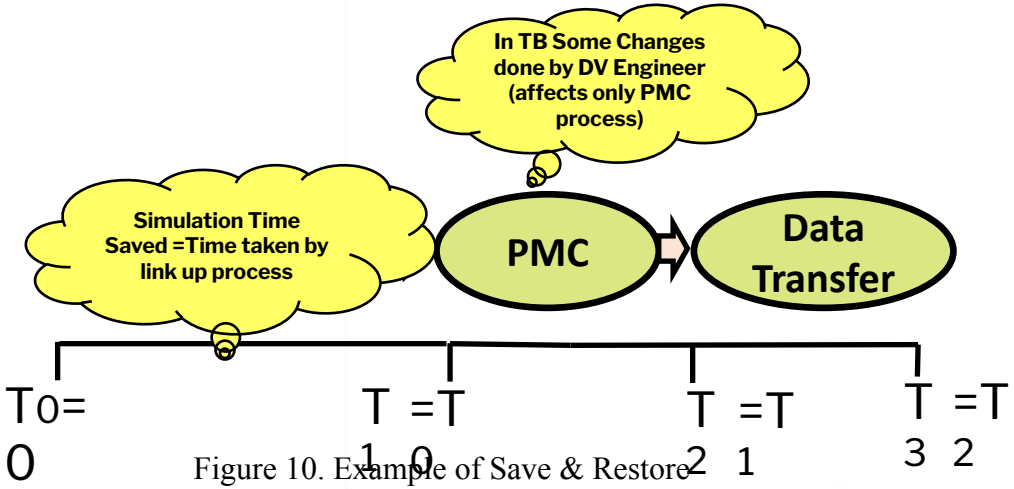
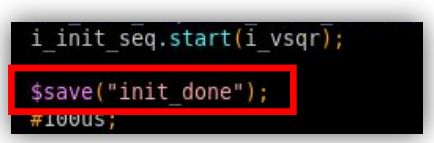
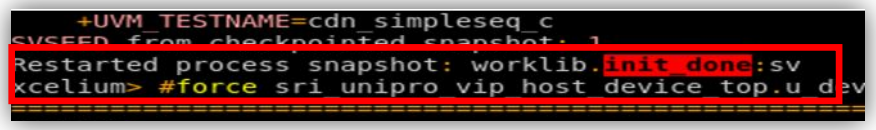


Figure 10. Example of Save & Restore



(a)



(b)

Figure 11.(a) Use of $save after process to be save (b) Evidence of UniPro linkup process stored. (Reference:UniPro2.0)

Functional Coverage Closure Using Script

- The coverage ranking script ranks the regression runs based on the coverage contribution and creates a test suite to target the obtained coverage.
- This comes very handy during not so uncommon cases of multiple RTL releases, where verification team has to iteratively rush for quick coverage closures.

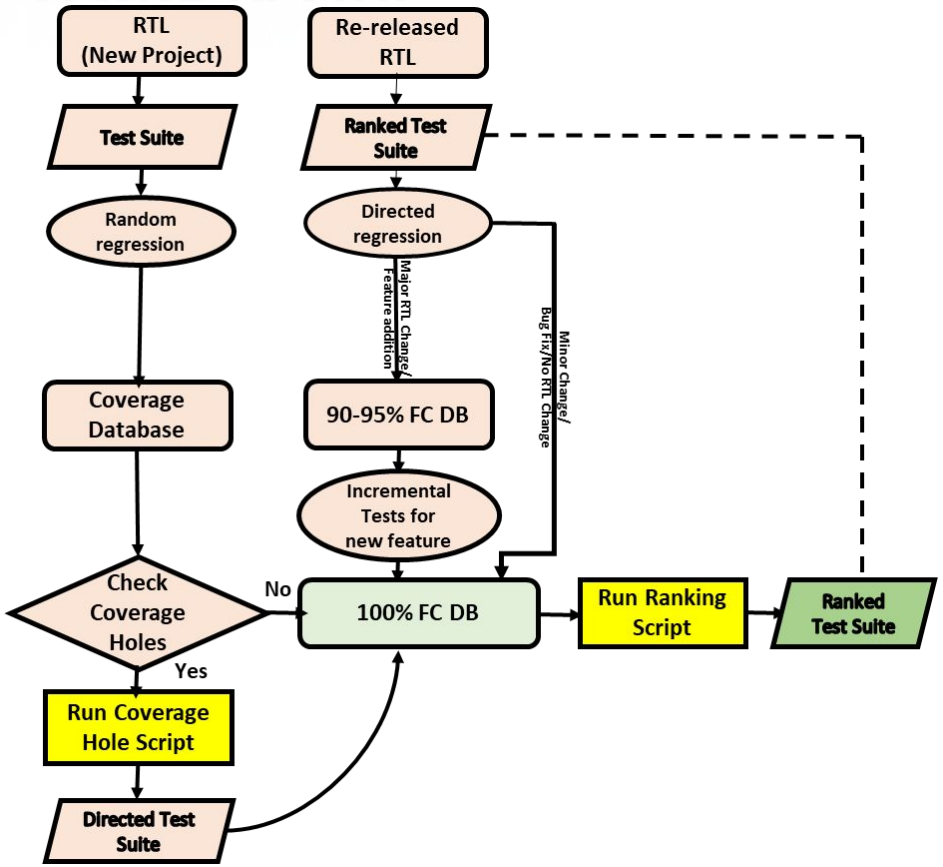


Figure 12. Test bench integrated with Coverage Hole Perl script for coverage.

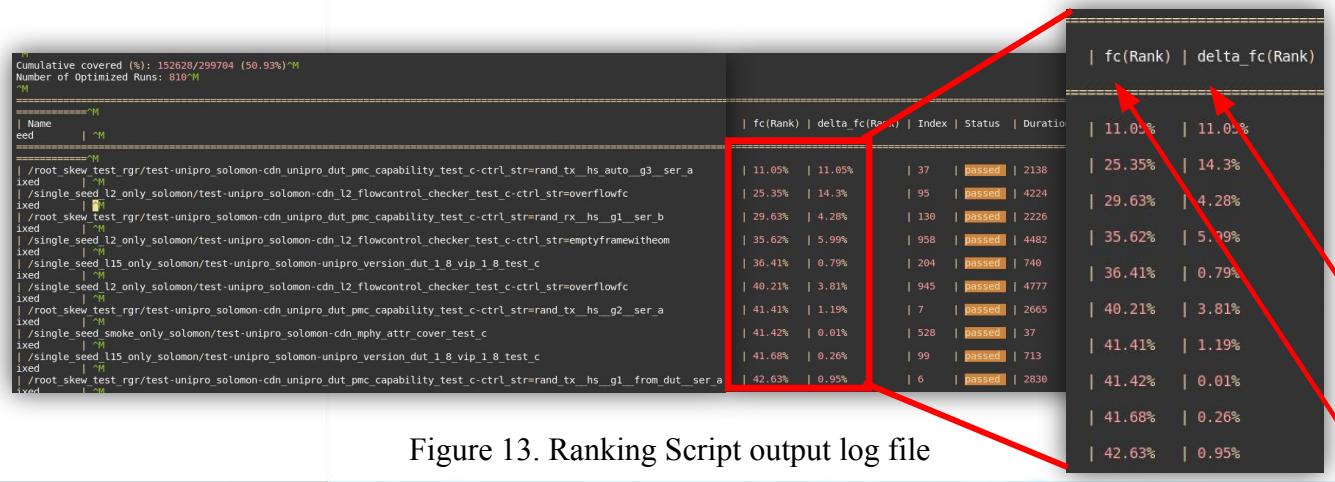


Figure 13. Ranking Script output log file

Contribution of a individual test case in total function coverage in %

Total function coverage in %

Functional Coverage Closure Using Script

- The main function of the Hole script agent is to obtain the latest coverage database, extract the information of un-hit bins and create directed test suite to hit the remaining bins with single run

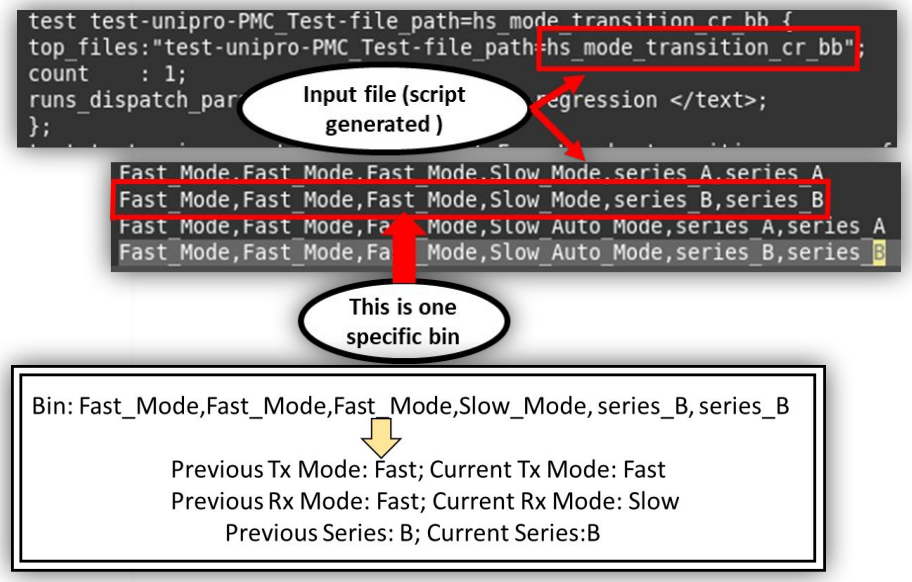


Figure 14. Input file generated by Coverage Holes Script (Reference:UniPro2.0)

```
for(int i=0;i<n;i++)begin//[
  `ifdef FC_Coverage_Script
      tx_mode_cnst = tx_mode.pop_back();
      rx_mode_cnst = rx_mode.pop_back();
      series_cnst  = series.pop_back();
      void'(pwr_mode_seq_h.randomize with
      {
        seq_start_side == FROM_DUT;
        Tx_Mode == tx_mode_cnst;
        Rx_Mode == rx_mode_cnst;
        Series == series_cnst;
      });
  `else
      assert(pwr_mode_seq_h.randomize());
  `endif
      pwr_mode_seq_h.start(i_vsqr);
     #100us;
end//]
```

These queues get the inputs from the script generated input files.

Figure 15. Example code of Power Mode Change (PMC) of UniPro2.0

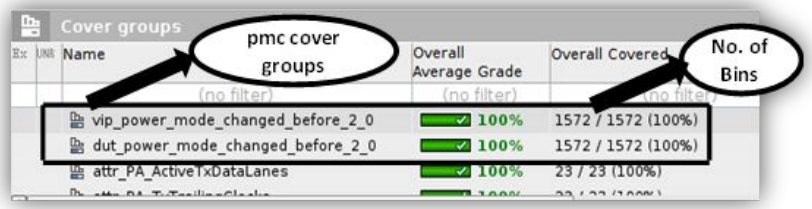


Figure 16. PMC related Major Cover groups (Reference: UniPro2.0)

Results & Conclusion

- With proposed scenarios which are close to real environment (Lane Mapping, and Skew Injection). We found out critical bugs in RTL. Like With lane mapping we found out when Lane1 alone was connected RTL was not able to decode data properly.
- Also, with skew Injection we are able to verify tolerance of the DUT beyond the spec limits. This kind of verification gives us a confidence on our IP's.
- Here, we published results in Table III by taking MIPI UniPro (high speed link) as our case study and its clearly visible the saved simulation time.
- This saved time in verification cycle can be use in hunting the corner bugs or it's also reduce the time to market of the product.
- Also, with proposed Coverage Hole Script and Ranking Script DV engineers can close the DV in quick succession.
- Same seed does not create same test vector in case of major RTL and thus TB change. As Coverage Hole script store the test vector thus reducing coverage slip.

Table II. Comparison of Number of Test Runs with and without Script (Reference : UniPro2.0 DV)

Approximate Number of Test Runs to Achieve 100 % FC Coverage			
Method	**Random Regression**	**After Adding Coverage Hole Script**	**After Running Ranking Script**
Total Test Run	20000-25000	8000-10000	5000-6000

Table III : Comparison of Run Time of Different Test Suites with and without Save-and-Restore (Reference: UniPro2.0 DV)

Comparison between run times with Save & Restore and Without Save & Restore		
Test Vectors	**Run time without Save & Restore setup**	**Run time with Save & Restore setup**
Simple Data Transfer	50mins-60mins	15-20mins
PMC followed by Data Transfer	60mins-70mins	20-25mins
Sanity Regression after minor updates in Test-bench	20hrs-24hrs	6hrs-8hrs
Full Regression	7days-10days	3days-4days

Table IV : Comparison of Coverage Replay with Ranking Test Suite (Reference: UniPro2.0 DV)

Approximate Coverage Replay with Ranking Test Suite regression (New RTL release)		
Method of generation	**Ranking Script (only)**	**Ranking + Coverage Hole Script**
FC (%)	80-87%	90-95%

Questions/Comments

Thank You