Leveraging more from GLS: Using metric driven GLS stimuli to boost Timing Verification

Sowmya Ega, Richardson Jeyapaul, Kunal Jani
Technical Lead, Analog Devices Inc., Bangalore, India.
Sowmya.ega@analog.com

Abstract

In today's complex designs, to realize the vision of shorter time to market and bug free silicon, more and more emphasis is done on the Design Verification. Gate level simulations do play a very vital role in the Design Verification cycle. This paper emphasizes the need of developing a metric driven methodology to gauge the quality of GLS stimuli, which in turn boosts the Timing verification.

I. INTRODUCTION

GLS is a permanent fixture in ASIC design flow that enhances the confidence in pre-silicon stage of verification. It is mainly used for reset-verification, timing verification of multi cycle paths & asynchronous paths. It helps verify the dynamic circuit behavior, which cannot be verified accurately by static methods. It is virtually the only flow that catches the issues missed by other flows (like RTL Simulations, synthesis, STA, Logic Equivalence Check).

Given the huge run times associated with gate level simulations, the typical strategy for picking the GLS stimuli is to consult with designers and pick a few system level tests which will target the critical features and timing paths of the design, However, this approach seems ad hoc when you end up seeing silicon issues related to multi cycle paths & false paths. A similar experience with the quality of silicon forced us to change our strategy for choosing the GLS testlist.

This paper describes the metric driven approach that was adopted to enhance and ensure the quality of GLS stimuli.

II. MULTI CYCLE, FALSE & TIMING CRITICAL PATHS

A. *Multi cycle path*

Multi cycle path is one in which the data launched from one flop is allowed to take more than one clock cycle to reach the destination flop.

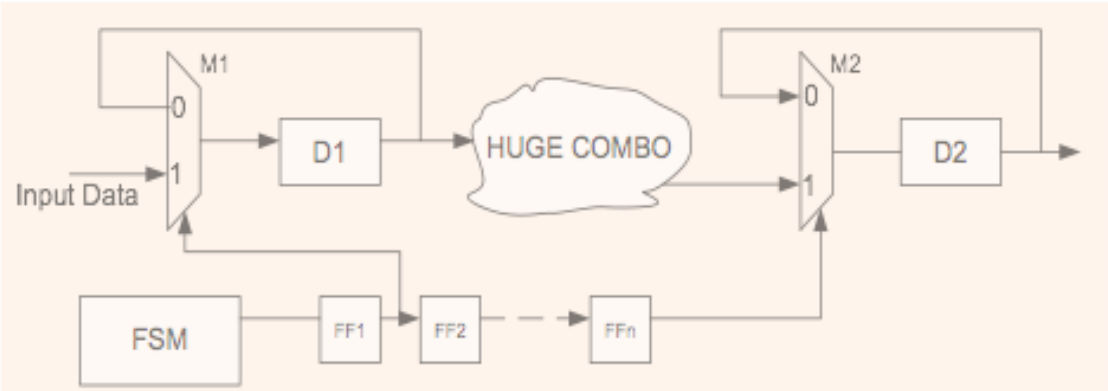

Figure 1: A sample Multi cycle path in a design

The above circuit in Figure 1 is an example of a Multi cycle path of “n-1” cycle. The input data becomes valid at M1 mux and the FSM generates a pulse for one clock cycle which enables the input data reach M1 Mux. The pulse then gets propagated through a shift register consisting of “n” number of flops and after “n-1” clock cycles, the M2 mux will enable the Input data reach D2 flop.

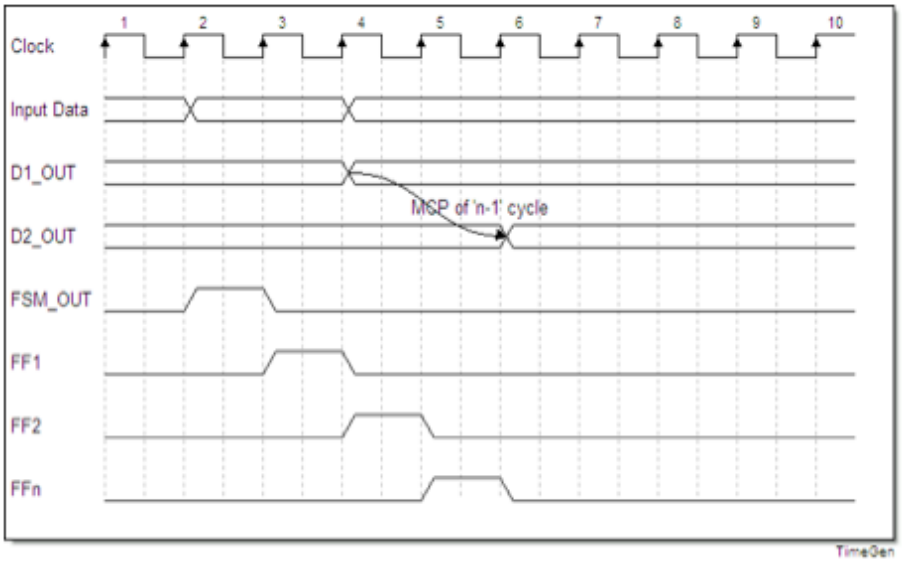

Figure 2 : Timing of a Multi cycle path of “n-1” cycle.

The timing diagram in Figure 2 corresponds to the circuit in Figure 1, whose N is equal to 3.

Multi cycle paths can be implemented in various ways. The most common ways of implementing a multi cycle path is through gating in data-path and gating in clock-path. Multi cycle paths can be specified such in STA. The STA tool will optimize these paths to achieve optimum area, power and timing. Specifying a multi cycle path as multi cycle is important, as the STA tool will use bigger drive strength cells to meet timing if not specified. This results in more area and power, and hence more cost.

B. Static False path

Static false paths are the timing paths where the change in source registers are not expected to get captured at the destination register within a particular time interval. These are the paths which exist in the design but those are logically/functionally incorrect. In short, false path is a timing path which cannot occur or a timing path where the timing can be ignored by STA tool.

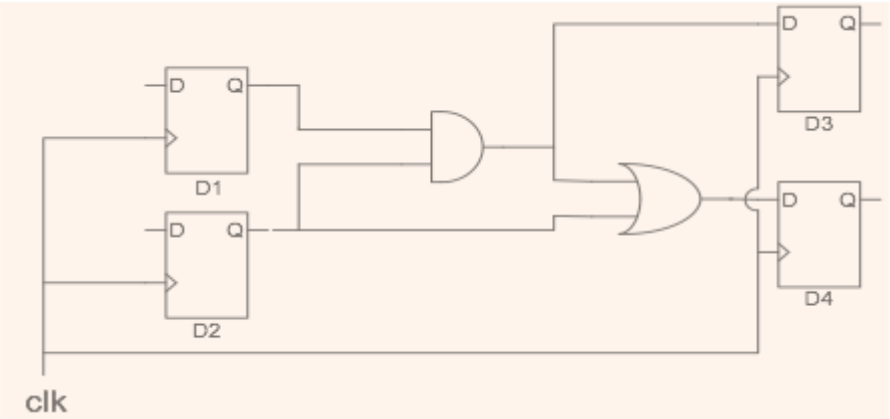

Figure3: A sample False path

In the circuit of figure 3, the change in D1.Q will never cause a change in D4.D. Hence, for that particular circuit D1->D4 can be treated as a false path. False paths can be specified such in STA, and is not optimized for timing by the STA tool.

C. Timing critical path

Critical timing paths are the timing paths whose timing is met marginally by STA tool, and are reported as such by STA tool. These timing paths can be critical on either setup, hold or both.

III. USING GLS FOR VERIFICATION OF MULTICYCLE, TIMING CRITICAL AND FALSE PATHS

Given the fact that the multi cycle and false paths are user defined, there is a good chance that a particular path might be wrongly declared as a Multi cycle or False path. If a path is not a multi cycle path, but declared as a multi cycle path by user, the timing is closed for multi cycle by STA tool. If a path is not a false path but declared as a one by user, the timing is ignored for such path by STA tool. This might result in an unexpected behavior of silicon. As a result, it is important to ensure that the GLS stimuli is indeed exciting the MCPs, false paths and other critical timing paths in the design. Else we might be running a risk of having paths which have not had any timing verification in both STA and GLS which can very well lead to a silicon bug.

We, therefore, decided to develop Functional cover properties to determine if the GLS test stimuli is indeed hitting the timing paths of interest. These Functional cover properties monitor if a particular timing path gets triggered in a simulation or not. As a first step towards developing these Functional cover properties, we started off with a list of begin and end points for the important design paths (multi cycle paths, false paths, timing critical paths) provided by the implementation team. Using this list as an input, the generation of functional cover properties for Multi-cycle paths, false paths and timing critical paths and the generation of assertions for false paths was automated. These properties, both cover and assert, are then integrated in the test-bench and while running the GLS regressions, the functional coverage is monitored. The coverage achieved defines the quality of GLS stimuli. Subsequently, the un-hit functional cover properties are analyzed and new tests are added to increase the coverage. This would help us understand if all the Multi cycle paths and False paths which were declared so by user are covered in the GLS regressions and are indeed Multi cycle and False paths. Hence improving the coverage will in turn improves the quality of GLS stimulus. It should be noted that the coverage would be nil for the Functional cover properties on static False-paths, as functionally those paths doesn't exist. If GLS stimuli exercises these paths, assert property will report it as an error.

A. Functional Cover property of a timing critical path with an example

Functional cover property for timing critical paths, monitors if a change in the Begin-point with respect to Begin-point clock resulted in a change at the Endpoint with respect to End-point clock.

The property can be explained in the below mentioned steps in detail.

1) Begin & End points of a timing path operates at different clocks.
2) Wait for a change in the transition of the begin point with respect to Begin-point clock & ignore any transition from "x".
3) Capture the End point value at the time, and the time at which Begin point is changed.
4) Wait for a change in the transition at the end-point at the nearest edge of the End-point clock & ignore any transition from "x". Also check, if the current value of end point, is not the same as that of captured in the point 3.
5) Capture the End point value at the time, and the time at which the End point gets transitioned.

The sample construct of the Functional cover property, which was described above is shown in figure 4. It also has the markings corresponding to the description mentioned above.

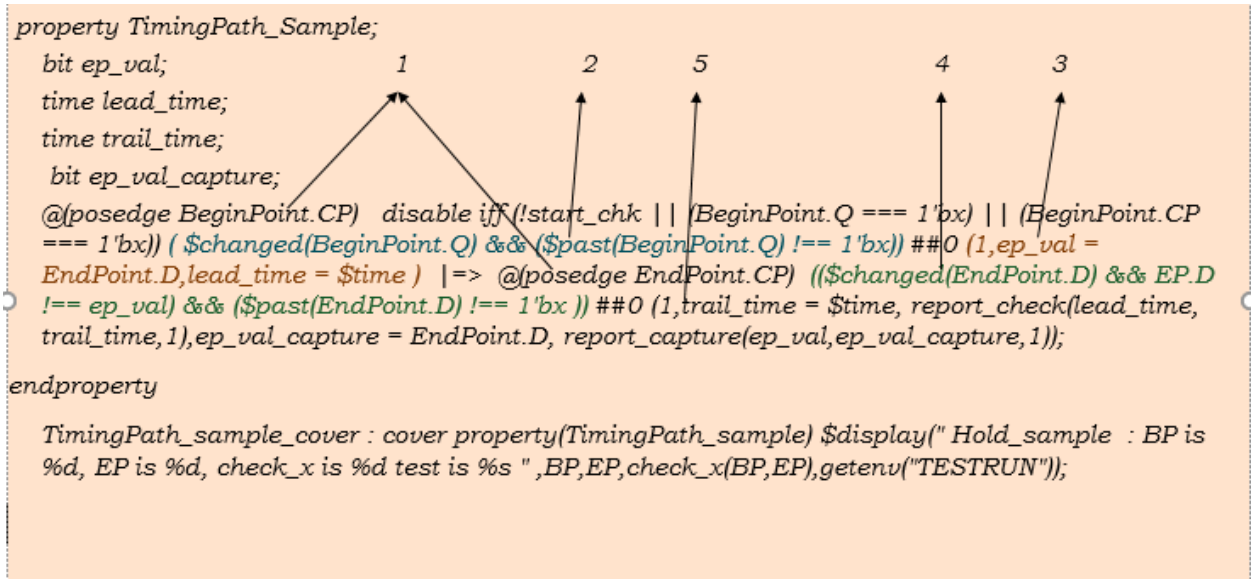

Figure 4: Sample Functional Cover property construct for a timing critical path

The Begin and End point pair given by Implementation team, is as shown in Figure5

```
Hold :  265
Endpoint:  u_muskaDigital_top/u_muskaDigital/u_digital_top_lv/u_dig_pwr_gate/u_dig_core/u_subsys/u_apb32_periph_top/pmg_clk_regs/
hfoscdivsyn_gen_d_reg/D
Beginpoint: u_muskaDigital_top/u_muskaDigital/u_digital_top_lv/u_dig_pwr_gate/u_dig_core/u_subsys/u_apb32_periph_top/pmg_clk_regs
/hfoscdivsyn_gen_reg/Q
Hold :  266
Endpoint:  u_muskaDigital_top/u_muskaDigital/u_digital_top_lv/u_dig_pwr_gate/u_dig_core/u_subsys/u_BusLogic/u_bridge_4dma/apbmst_
PSEL_reg_reg_2_/D
Beginpoint: u_muskaDigital_top/u_muskaDigital/u_digital_top_lv/u_dig_pwr_gate/u_dig_core/u_subsys/u_BusLogic/u_bridge_4dma/cmd_fi
fo_wrPtr_reg_0_/Q
```

Figure5: Begin and End point pair

The Begin and End point pair shown above in Figure5, is passed to the script to generate the Functional cover property as shown below in figure 6.

```
`define Launch_Path
proj_tb.u_top.u_projDigital_top.u_projDigital.u_digital_top..u_dig_core.u_subsys.u_periph_top.clk_regs.clk_gen_reg

`define Capture_Path
proj_tb.u_top.u_projDigital_top.u_projDigital.u_digital_top..u_dig_core.u_subsys.u_periph_top.clk_regs.clk_gen_d_re
g


property Hold_265;

 bit ep_val;  time lead_time;  time trail_time;  bit ep_val_capture;

 @(posedge Launch_Path.CP)   disable iff (!start_chk ||  (Launch_Path.Q === 1'bx) || (Launch_Path.CP === 1'bx) ) (
$changed(Launch_Path.Q ) && ($past(Launch_Path.Q) !== 1'bx)) ##0 (1,ep_val = Capture_Path.D,lead_time = $time)
|=>  @(posedge Capture_Path.CP) ##0   (($changed(Capture_Path.D)  &&Capture_Path.D !== signed'(ep_val)) &&
($past(Capture_Path.D) !== 1'bx )) ##0 (1,trail_time = $time, report_check(lead_time, trail_time,265),ep_val_capture =
Capture_Path.D, report_capture(ep_val,ep_val_capture,265));

endproperty

Hold_265_cover : cover property(Hold_265) $display(" 265  : BP is %d, EP is %d, check_x is %d test is %s " ,
Launch_Path.Q,Capture_Path.D,samp_fun::check_x(Launch_Path.Q,Capture_Path.D),getenv("TESTRUN"))
```

Figure6: Functional cover property of a timing critical path

The Hold265 cover property mentioned above in Figure6, is exercised in GLS and the waveform snippet is shown as below in Figure 7a & Figure 7b. Figure7a shows the transition of the Begin point resulted in transition of End point. Figure 7b shows the Cover property registering the number of times, the path got exercised.

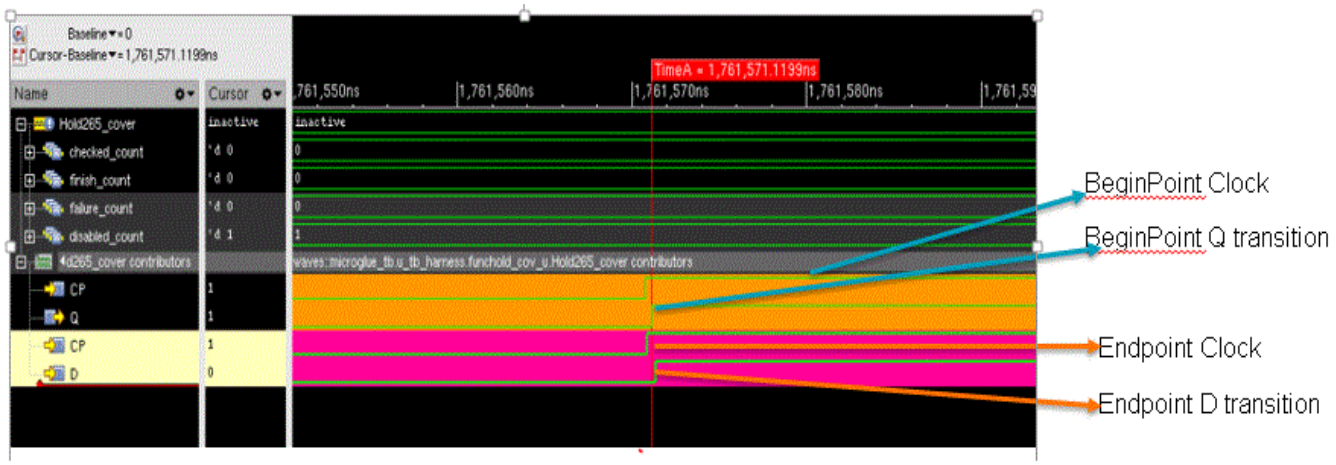

Figure 7a: Change of the Begin-point of a path resulting in the change of the End-point of a path

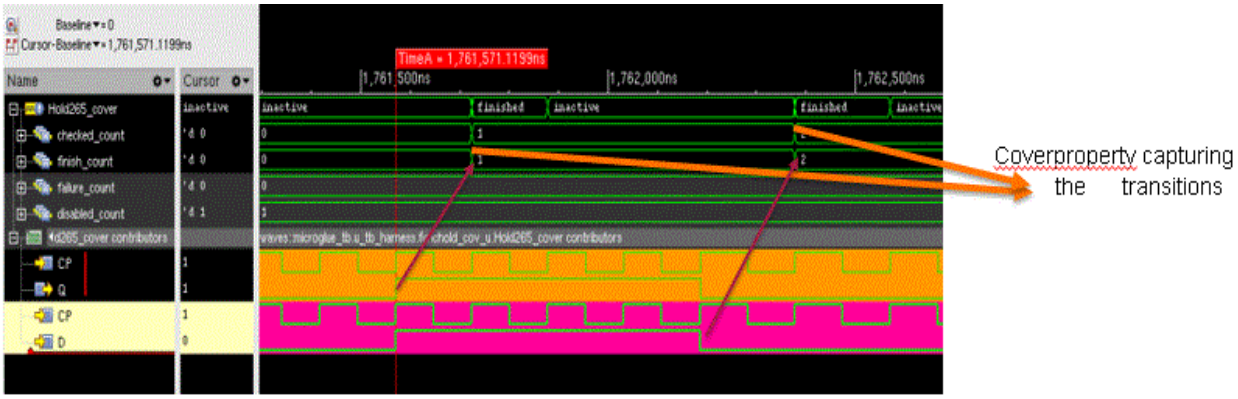

Figure 7b: Number of times a path is exercised in a test, is captured by the cover property

B. Functional Cover property of a clock-gated multi cycle path with an example

Functional cover property for multi cycle paths, monitors if a change in the Begin-point with respect to Begin-point clock resulted in a change at the Endpoint with respect to End-point clock after more than a single cycle.

The property can be explained in the below mentioned steps in detail.

1) Begin & End points of a timing path operates at different clocks.
2) Wait for a change in the transition of the begin point with respect to Begin-point clock & ignore any transition from "x".
3) Capture the End point value at the time, and the time at which Begin point is changed.
4) Wait for the next posedge of the Begin Point clock and make sure that the source Data is stable.
5) Wait for a change in the transition at the end-point at the nearest edge of the End-point clock & ignore any transition from "x". Also check, if the current value of end point, is not the same as that of captured in the point 3.
6) Capture the End point value at the time, and the time at which the End point gets transitioned.
7) The time captured in point 3 & 6 is used to determine the timing relation of the path

The sample construct of the Functional cover property, which was described above is shown in figure 8. It also has the markings corresponding to the description mentioned above.

```
property Hold_Sample_MCP;
  bit ep_val;
  time lead_time;
  time trail_time;
  bit ep_val_capture;
  @(posedge BeginPoint.CP)   disable iff !start_chk || (BeginPoint.Q === 1'bx) || (BeginPoint.CP ===
  1'bx)) ( $changed(BeginPoint.Q) && ($past(BeginPoint.Q) !== 1'bx)) ##0 (1,ep_val =
  EndPoint.D,lead_time = $time )  |-> @(posedge BeginPoint.CP) ($stableBeginPoint.D)  |=> @(posedge
  EndPoint.CP )  ($changed(EndPoint.D) && EP.D !== ep_val) && ($past(EndPoint.D) !== 1'bx )) ##0
  (1,trail_time = $time, report_check(lead_time, trail_time,1),ep_val_capture = EndPoint.D,
  report_capture(ep_val,ep_val_capture,1));

endproperty

  Hold_sample_mcp_cover : cover property(Hold_sample_MCP) $display(" Hold_sample  : BP is %d, EP
  is %d, check_x is %d test is %s " ,BP,EP,check_x(BP,EP),getenv("TESTRUN"));
```

1 6 2 4 5 3

Figure 8. Sample Functional cover property for a Multi cycle path

The Begin & End point list for the multi cycle paths given by the implementation team, is passed onto the script to generate the Functional cover property as shown in Figure 9 below.

```
`define Launch_Path
proj_tb.u_top.u_projDigital_top.u_projDigital.u_digital_top.aon..edgeDet_clock_dead_edgeDet_syncQ1_reg
`define Capture_Path
proj_tb.u_top.u_projDigital_top.u_projDigital.u_digital_top.aon..edgeDet_clock_dead_edgeDet_syncQ2_reg


property Hold169_MCP_CG;
        bit ep_val; time lead_time; time trail_time;   bit ep_val_capture;
        @(posedge Launch_Path.CP)   disable iff (!start_chk || (Launch_Path.Q === 1'bx) || (Launch_Path.CP === 1'bx) ) (
$changed(Launch_Path.Q ) && ($past(Launch_Path.Q) !== 1'bx)) ##0 (1,ep_val =Capture_Path.D,lead_time = $time)  |-> @(posedge
Launch_Path.CP) $stable(Launch_Path.D)  ##1  @(posedge Capture_Path.CP) ##0  ($changed(Capture_Path.D) &&
($past(Capture_Path.D) !== 1'bx ));
endproperty
Hold169_MCP_CG_cover : cover property(Hold169_MCP_CG) $display(" Hold169_MCP_CG  : BP is %d, EP is %d, check_x is %d
test is %s", Launch_Path.Q,Capture_Path.D,samp_fun::check_x(Launch_Path.Q,Capture_Path.D),getenv("TESTRUN"));
```

Figure 9. Functional cover property for a Multi cycle path

This functional cover property mentioned in the figure 9, is exercised in GLS and the waveform snippet for the same is shown as below in figure 10.

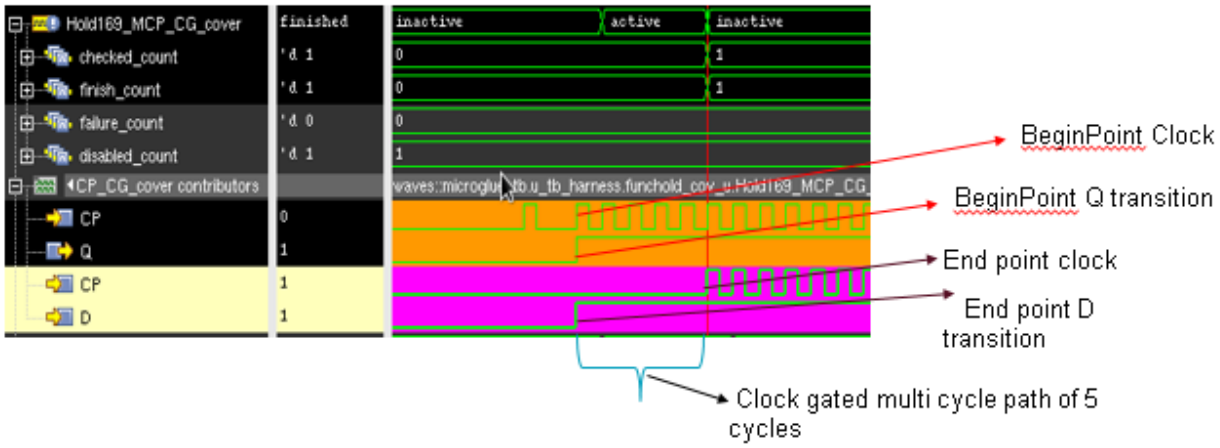

Figure 10: Multi cycle path exercised in a test, is captured by the cover property

C. Functional Cover & Assert property of a static false path :

Functional cover property of a false path monitors if a change in the Begin-point with respect to the Begin-point clock resulted in a change in the End-point with respect to the End-point clock, and the corresponding assert property gives out an uvm error.

The property can be explained in the below mentioned steps in detail.

1) Begin & End points of a timing path operates at different clocks.
2) Wait for a change in the transition of the begin point with respect to Begin-point clock & ignore any transition from "x".
3) Capture the End point value at the time, and the time at which Begin point is changed.
4) Wait for a change in the transition at the end-point at the nearest edge of the End-point clock & ignore any transition from "x". Also check, if the current value of end point, is not the same as that of captured in the point 3.
5) Capture the End point value at the time, and the time at which the End point gets transitioned.

The sample construct of the Functional cover/assert property, which was described above is shown in figure 11. It also has the markings corresponding to the description mentioned above.

```
property FalsePath_Sample;
   bit ep_val;                      1              2        5                 4          3
   time lead_time;
   time trail_time;
    bit ep_val_capture;
   @(posedge BeginPoint.CP)   disable iff (!start_chk || (BeginPoint.Q === 1'bx) || (BeginPoint.CP
   === 1'bx)) ( $changed(BeginPoint.Q) && ($past(BeginPoint.Q) !== 1'bx)) ##0 (1,ep_val =
   EndPoint.D,lead_time = $time )  |=> @(posedge EndPoint.CP) (($changed(EndPoint.D) && EP.D
   !== ep_val) && ($past(EndPoint.D) !== 1'bx )) ##0 (1,trail_time = $time, report_check(lead_time,
   trail_time,1),ep_val_capture = EndPoint.D, report_capture(ep_val,ep_val_capture,1));

endproperty

   FalsePath_sample_cover : cover property(FalsePath_sample) $display(" FalsePath_sample  : BP is
   %d, EP is %d, check_x is %d test is %s " ,BP,EP,check_x(BP,EP),getenv("TESTRUN"));

   FalsePath_sample_assert : assert property(FalsePath_sample) )  `uvm_error(" This is a False Path,
but is covered . FalsePath sample: BP is %d, EP is %d, check_x is %d test is %s "
,BP,EP,check_x(BP,EP),getenv("TESTRUN"));
```

Figure 11. Sample Functional cover & assert property for a False path

The functional cover/assert property is generated using a script, for the paths given by implementation team. Figure 12. is the sample snippet of a false path from gate level simulation waveform, where a change in the source signal doesn't trigger the destination signal.

Figure12. False path, where the coverage in cover property is nil.

IV. CONCLUSION & FUTURE WORK

Coding the functional cover properties for all the multi cycle & timing critical paths is a very tedious and time-consuming task, but the in-house script developed for Functional cover property generation for timing paths made the task easier.

The verification of multicycle, false, and critical timing paths can thus be ensured by writing the functional cover properties. The original GLS test-list covered only 53% of the multicycle & timing critical paths in the design. With the help of functional cover properties, we enhanced our GLS Stimuli by adding few tests and achieved 100% coverage on timing critical & multi cycle paths; thus boosting our tape-out confidence. The implementation team has been appreciative of this approach; and sees this as a way to complement the STA effort. Going forward, we have decided to adopt this approach to ensure the quality of GLS Stimuli in all our future projects.

We are exploring the various ways a multi-cycle path can be implemented, and develop the cover properties for the same.