Timing Coverage: An Approach to Analyzing Performance Holes

Surbhi Kalia, Shubhadeep Karmakar, Vikas Makhija, Apoorva Mathur
Synopsys India Pvt. Ltd.
kalia@synopsys.com, shubhade@synopsys.com, vikasm@synopsys.com, apoorvam@synopsys.com

I. Introduction

Designs are becoming more complex and require advanced verification to cover each aspect of the specification. While trying to achieve comprehensive verification of a complex design, repetitive testing of the same functionality can waste a lot of time. This repetition can be avoided using coverage to achieve comprehensive verification. Functional coverage is a method used to find holes in verification with respect to the functional specification of a design; however, due to the increasing complexity of designs, it may not be sufficient for plugging all the verification holes. It is also equally as important to check if performance critical aspects of a design have been verified. This paper proposes a timing coverage model for measuring performance verification closure. We will use LPDDR (Low Power Double Data Rate) Verification IP (VIP) as an example to demonstrate timing coverage driven verification for finding performance verification holes.

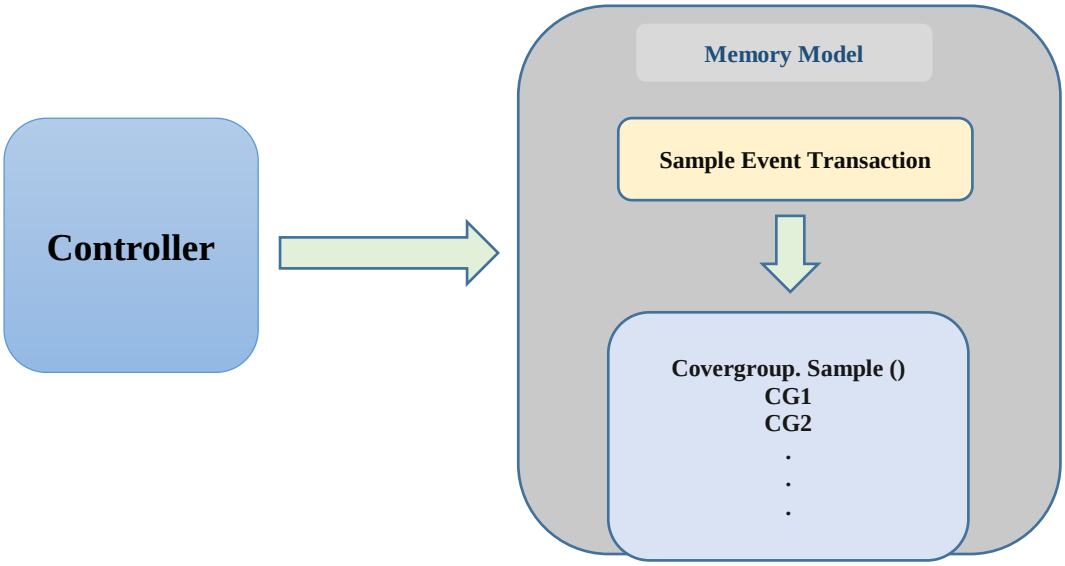


Fig. 1: A functional coverage model

The most common coverage metrics in a verification flow are code coverage and functional coverage. Code coverage helps with identifying the values, sub-expressions in conditional statements, and evaluation of decision statements. It also helps to ascertain whether or not a line or block has been executed. It cannot however, provide detailed information about the functional correctness of design logic. To cover correctness of features in any design, functional coverage metrics are implemented as part of verification cycles and the result predicts the functional completeness of verification.

The functional coverage can determine the verification completeness, but does not guarantee whether performance bottlenecks have been tested or covered in the design. Fig. 1, demonstrates a generic functional coverage model where different functional features of a design have been implemented as covergroups/coverpoints. An event transaction defines the point in the design where the corresponding covergroup is sampled. In this example, we have proposed a new area, or subdomain, in functional coverage, referred to as timing coverage, which helps in testing the design against different timing variations and ensures all the performance critical testing has been completed for the design under test (DUT).

II. UVM ARCHITECTURE WITH FUNCTIONAL COVERAGE

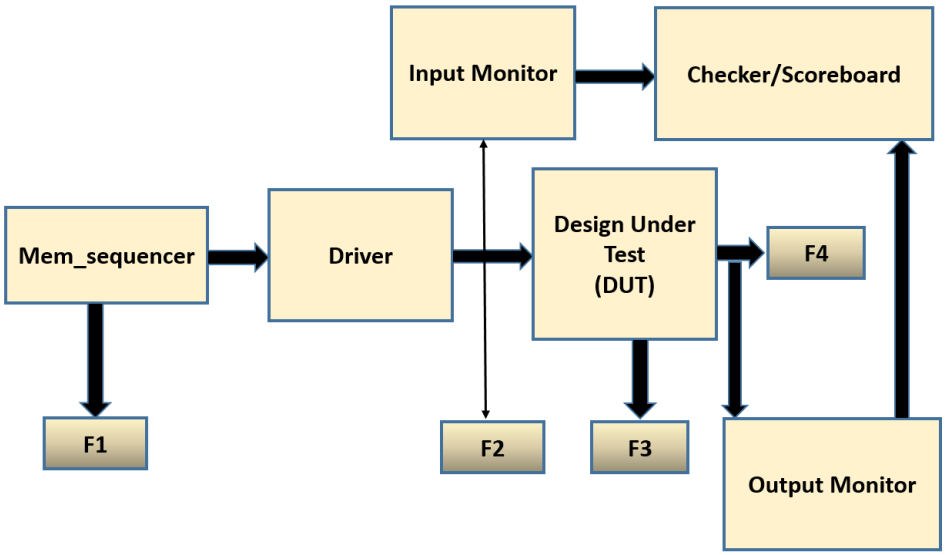


Fig. 2: UVM VIP architecture with functional coverage points

The LPDDR VIP model in Fig. 2 is implemented with different functional coverage models, with various coverage points at different components.

- **F1**: Functional coverage points are very near the randomization.
- **F2**: Functional coverage points are sampled at input interface of DUT.
- **F3**: Functional coverage points which sample internal DUT states.
- **F4**: Functional coverage points which sample output interface of DUT.

III. MODELING OF TIMINGS WITH TIMING COVERAGE

As previously discussed, the timing coverage model is a subset of the functional coverage model, which is designed to capture the performance centric holes. In the timing coverage model, the covergroups and coverpoints have been configured in order of delay values, ranging from minimum supported timing to maximum supported timings. If the verification test flow of a DUT have the scenarios which cover minimum and maximum timings, the particular coverbin is treated as a hit in the coverage report. Fig. 3 shows a typical verification model where timing coverage ensures the performance completeness of a system.

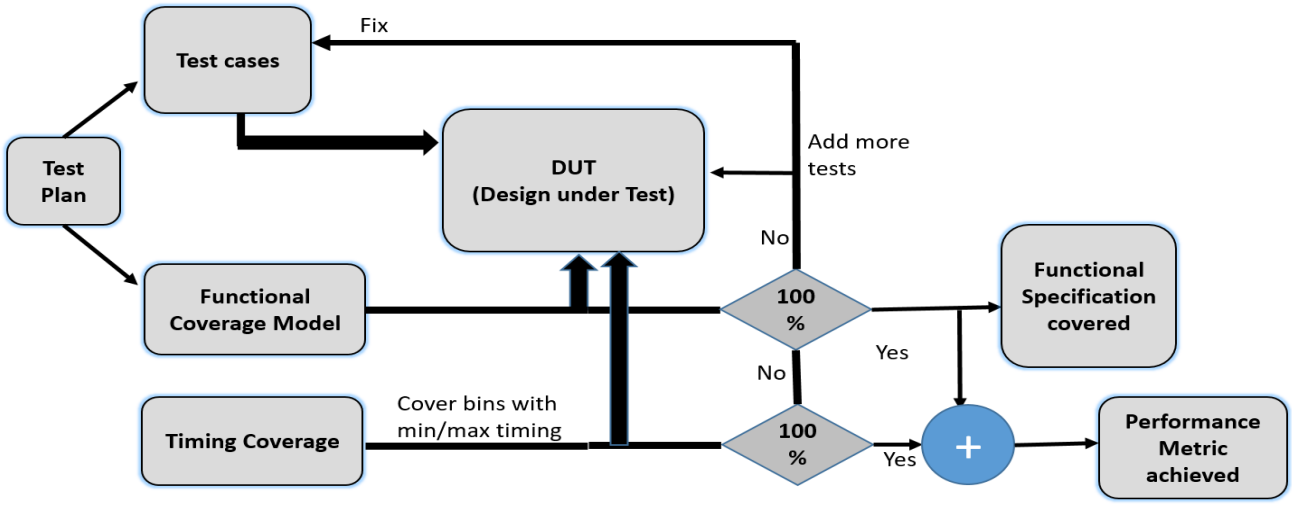


Fig. 3: Functional and timing coverage flow

All the functionalities defined in the test plan to be tested are mapped to coverpoints in the functional coverage model. Whenever the functionality is hit during simulation, the functional coverage point is automatically updated – i.e.. 100% functional coverage ensures functional specification is covered. On the other hand, with less than 100% functional coverage, more test scenarios can be added to ensure verification is complete. Thus, the functional coverage model discovers the uncovered cases from the implemented coverpoints and crosspoints which covers the functional specification.

Similarly, the timing coverage model will ensure all minimum and maximum supported timings are implemented with the help of coverpoints and crosspoints. Hence, 100% timing coverage means that DUT has been tested for every possible timing. If 100% is not met, more test scenarios covering the uncovered delays in timing coverage model should be added to achieve performance completeness.

The purpose of performance driven verification is to capture the performance bottlenecks of DUTs in addition to capturing functional holes. When 100% functional coverage and timing coverage are not achieved, more test cases can be added to ensure functional specification and all maximum and minimum timings have been covered. Ultimately, performance completeness of the DUT is ensured with 100% functional coverage and 100% timing coverage.

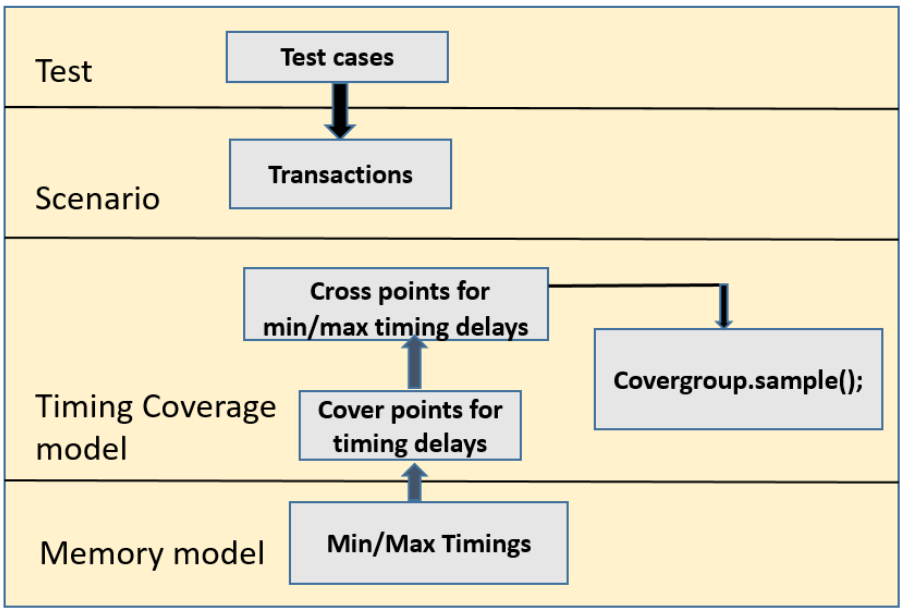


Fig. 4: Timing Coverage Model Environment architecture

The whole timing coverage model Environment consists of a test module, transaction generator, timing coverage model, and memory model. The supported minimum and maximum timings are calculated from the memory model and mapped into coverpoints for the minimum or maximum supported timings. The timing coverage model consists of covergroup definition, consisting of coverpoints and crosspoints for timing delays for all the valid configurations possible and ignore bins for the invalid values of timing delays for different configurations. A test module consists of all the combinations of scenarios to hit all the minimum and maximum supported timings coverage bins. Fig. 4 shows the typical timing coverage model environment where the timing coverage model interacts with a test module and memory module.

Bins for ranges	**Represents the ranges for Minimum and Maximum Delays.**
Default bins	**Represents specific values of delays.**
Illegal bins	**Represents illegal values of timing delays for particular configurations.**
Cross bins	**Represents cross of timing delays with all the configurations.**
Ignore bins	**Represents the ignored values for the invalid combinations.**

Fig. 5: Representing different bins in timing coverage

The covergroup consists of coverpoints and crosspoints which contains parameters called bins as summarized in Fig. 5.

IV. METHODOLOGY

In this paper, we have used a typical case of LPDDR VIP where different timing metrics, as shown in Fig. 6, are tested as a cover point for the timing coverage model. The timing delays used as cover point can be categorized as follows:

- **Command to command delay** – In a memory architecture the bus utilization is an important aspect for a performance calculation. Analyzing the delays, like delay between different read commands, delay between a read to write command, active to read delay, etc., can help in finding bottlenecks.
- **Data Hit/Miss Rates** - The efficiency of DRAM based architecture is based on no of read/writes without opening a page. The timing model calculates the Read/Write without precharge command (data hit) and precharge and activate combination (data miss Scenarios).
- **Power down time and Self refresh time** – This time can be configured to check the time during which the memory was in power down or self-refresh state.
- **Refresh time** – Refresh is an important parameter of a DRAM. Timings between different Refresh per bank and Refresh all banks are also captured in the timing coverage model.
- **Setup, hold and pulse width time** - As the memory design is a synchronous design, so the timings like setup/hold/pulse width are part of the timing metrics. The timing model covering different setup hold and pulse width time for the signals with respect to clock.

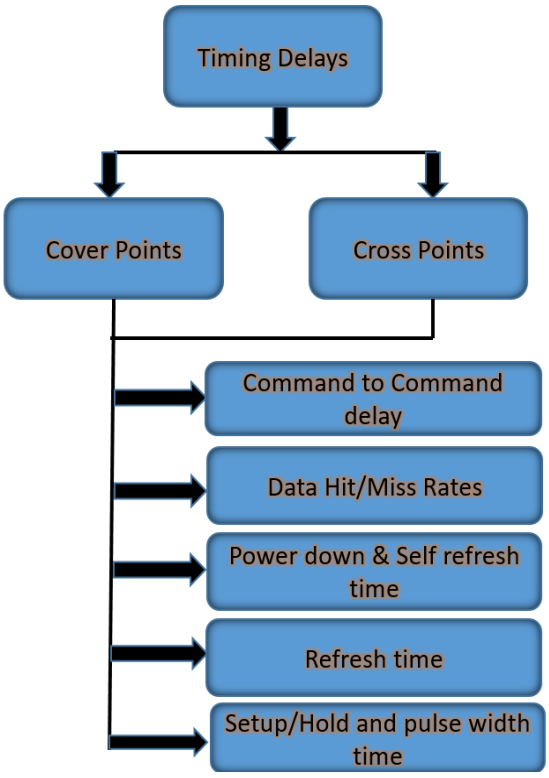


Fig. 6: Timing coverage implementation

For LPDDR VIP, cross coverage is specified between different timing metrics and functional cover bins like burst length, data width, latencies, etc. This helps to streamline the result with a single coverage report for both functional and performance aspects. This report can be analyzed to achieve better protocol mapping and capture performance holes in all the applicable configurations in the following ways:

1. **Command to Command Delays**: Delays are required between different memory commands to limit the efficiency of pipelined accesses. Delays are calculated from one command to another command like write to read, read to read, write to write, etc., which are mapped into the coverpoints of covergroup as shown in Fig. 7. Then, cross coverages are implemented for maximum/minimum delays with all the valid configurations (Fig. 8). The covergroup is sampled based on the triggering of events as shown in Fig. 9.

```
If (current_cmd == Write)
begin
if (next_cmd == CMD1)
  delay_1=clock_count
{ Execution syntaxes };

else if(next_cmd == CMD2)
  delay_2=clock_count
{ Execution syntaxes } ;

else if(next_cmd == CMD3)
  delay_3=clock_count
{ Execution syntaxes };
end
```

Fig. 7: Calculating Write command to different commands delays

```
covergroup lpddr5_wr_to_wr_in_same_bg ;

    option.per_instance = 1;
    type_option.comment = "Covering WR command to WR command timing in same bg in BG Mode";

lpddr5_wr_to_wr_min_delay_in_same_bg_without_cas_sync : coverpoint status.tccd_l
    {
      option.weight = 0;
      option.comment = "Covering WR_16/WR_32 to WR_16/WR_32 command Minimum delay in same bg in BG mode";
       //Bins for minimum write command to write command delay
    }
lpddr5_wr_to_wr_max_delay_in_same_bg_without_cas_sync : coverpoint status.tccd_l
    {
      option.weight = 0;
      option.comment = "Covering WR_16/WR_32 to WR_16/WR_32 command Maximum delay in same bg in BG mode";
      //Bins for maximum write command to write command delay
    }
cross_lpddr5_wr_to_wr_min_delay_in_same_bg_with_bl_ckr : cross
lpddr5_wr_to_wr_min_delay_in_same_bg_without_cas_sync,burst_length_cp,ckr_cp
    {
      option.weight = 1;
      type_option.weight =1;
      option.comment = "Covering cross of write to write min delay,ckr and burst_length in same bg in BG
mode";
      //Ignore bins for illegal values of min delay for ratio and burst length
    }
cross_lpddr5_wr_to_wr_max_delay_in_same_bg_with_bl_ckr_wl_wl_select : cross
lpddr5_wr_to_wr_max_delay_in_same_bg_without_cas_sync,write_latency_value_cp,mode_register_wl,wl_select_cp,b
urst_length_cp,ckr_cp,mode_register_wck_pst
    {
      option.weight = 1;
      type_option.weight =1;
      option.comment = "Covering cross of write to write max delay,write_latency_value,wckpst,ckr and
burst_length in same bg in BG mode";
      //Ignore bins for illegal values of max delay for write latency, burst length, ratio, wckpst value
    }
```

Fig. 8: Cross coverage of write to write command delay with valid configurations

```
fork
//Sampled when write to write event is triggered
  begin
   Forever@(write_to_write_command_event)begin
     Covergroup.sample();
    end
   end
```

Fig. 9: Sampling logic for covergroups

2. **Data Hit/Miss Rates**: A data hit is defined as any read or write operations to any open bank. A typical performance model of a memory ensures increased data hit transactions or reduced number of data miss transactions as shown in Fig. 10.

```
covergroup data_hit_data_miss ;
option.per_instance = 1;
type_option.comment = "Covering Data hit and data miss rates in LPDDR";

data_hit_data_miss : coverpoint data_hit_miss_rate ;
{
 option.weight = 0;
 option.comment = "Covering data hit and miss rates "
 bins data_hit = 1'b1;
 bins data_miss = 1'b0;
}
endgroup
```

Fig. 10: Covergroup for data hit/miss rates

3. **Power down and self-refresh time**: Power down and self-refresh time can be included in timing coverage to ensure minimum timing is followed when memory is in power down or self-refresh state to meet the performance metric.

4. **Refresh timings**: Refresh delays are important parameters in DRAM because the capacitor is leaking and must be periodically refreshed in order to not lose its data. Per bank refresh to per bank refresh different bank delay is configured in timing coverage model as shown in Fig. 11.

```
covergroup refpb_to_refpb_delay;
option.per_instance = 1;
type_option.comment = "Covering refpb to refpb time in
LPDDR";

refpb_to_refpb_delay : coverpoint refpb_to_refpb_time ;
{
 option.weight = 0;
 option.comment = "Covering refpb to refpb command delay
in lpddr "
 bins ref_per_bank_to_ref_per_bank_delay= {90};
}
endgroup
```

Fig. 11: Covergroup for refresh timings

5. **Setup, hold and pulse width time**: Setup, hold and pulse width time ensures reliable sampling of data, hence it forms an important aspect of the timing coverage model. For example, tdss (DQS falling edge to CK setup time) and tdsh (DQS falling edge hold time from CK) in LPDDR memory can be covered in the timing model as shown in Fig. 12.

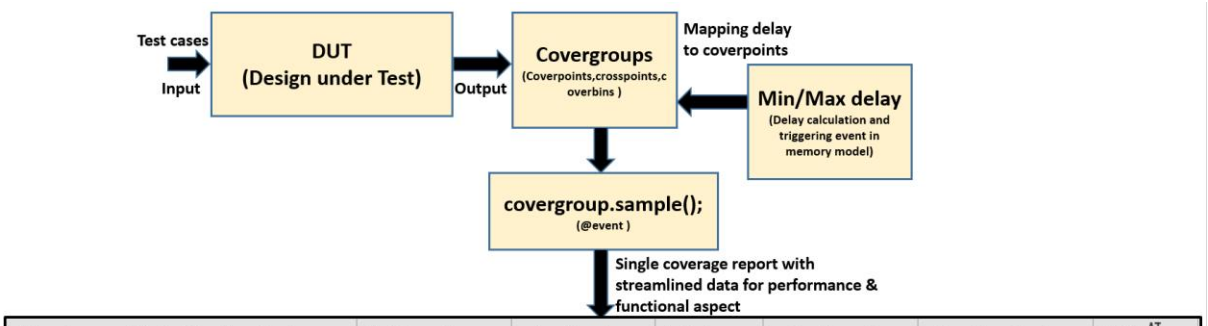
```
covergroup lpddr_min_tdss;
option.per_instance = 1;
type_option.comment = "Covering DQS falling edge to
CK setup time in LPDDR";

lpddr_min_tdss: coverpoint tdss_ps ;
{
 option.weight = 1;
 option.comment = "Timing : tdss_ps "
 bins tdss_ps= {tdss_min_ps};
}

endgroup
```

Fig. 12.: Covergroup for setup timing in LPDDR

Better protocol mapping is achieved when cross coverages are implemented for all the timing delays with the applicable configurations. Moreover, a streamlined data is achieved for better performance. This performance data can be utilized to find holes in functional specification and performance aspect as shown in Fig. 13.



lpddr5_wr_to_wr_max_delay_in_different_bg_without_cas_sync	write_latency_value_cp	mode_register_wl	wl_select_cp	burst_length_cp	ckr_cp	mode_register_wck_pst	COUNT	AT LEAST
wr_to_wr_max_delay_different_bg_25	write_latency_value_16	mode_register_wl_11	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_2	6	1
wr_to_wr_max_delay_different_bg_25	write_latency_value_16	mode_register_wl_11	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_1	6	1
wr_to_wr_max_delay_different_bg_24	write_latency_value_16	mode_register_wl_11	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_0	12	1
wr_to_wr_max_delay_different_bg_24	write_latency_value_15	mode_register_wl_10	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_2	6	1
wr_to_wr_max_delay_different_bg_24	write_latency_value_15	mode_register_wl_10	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_1	6	1
wr_to_wr_max_delay_different_bg_23	write_latency_value_15	mode_register_wl_10	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_0	12	1
wr_to_wr_max_delay_different_bg_23	write_latency_value_14	mode_register_wl_9	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_2	6	1
wr_to_wr_max_delay_different_bg_23	write_latency_value_14	mode_register_wl_9	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_1	6	1
wr_to_wr_max_delay_different_bg_22	write_latency_value_14	mode_register_wl_9	wl_select_set_b	bl_32_bins	ckr_4_1	mode_register_wck_pst_0	12	1
wr_to_wr_max_delay_different_bg_21	write_latency_value_16	mode_register_wl_11	wl_select_set_b	bl_16_bins	ckr_4_1	mode_register_wck_pst_2	6	1

Fig. 13.: Timing coverage report generation framework

V. RESULTS

Overall, this paper focused on the functional timing coverage based on the various mode register configurations. The coverage reports were collected using the functional coverage model previously mentioned.

Delays for different commands get changed based on the Write/Read latencies, and these delays are re-configurable. We provided different delays by reconfiguring these for each of the transactions as shown in Fig. 13, and monitored each of the corresponding delays independently in the cross coverage report.

Comparisons between functional and timing coverage were also done. Using functional coverage only, some scenarios were missed due to limited test cases to verify each of the timings; however, with timing coverage and appropriate directed test cases, the covered scenarios increased by 200%. Overall, a good performance improvement.

Testing	**Coverage Statistics**	
	With functional Coverage	***With Timing Coverage + Direct Test Cases***
Coverage Percentage	**86.57**	**100**
Covered scenarios	**187**	**560**
Uncovered scenarios	**29**	**0**
Missing holes	**29**	**0**

Fig. 14: Coverage statistics result

VI. SUMMARY

The timing coverage model helps to analyze the holes with respect to performance verification. An example of LPDDR DRAM was used to explain the different possible

covergroups and then a coverage report was generated using high-dimension cross coverage (cross coverage of more than two variables) which can be used as a feedback for both verification test plan and DUT limitations.

We implemented the timing coverage model consisting of static timing delays, however this approach can be extended to dynamic delay values also. Moreover, this timing coverage report can be utilized for better protocol mapping using the spec linking technique to ensure that both performance and functional specification aspects are covered.

REFERENCES

[1] El-Ashry, Sameh, and Khaled Salah. "A functional coverage approach for direct testing: An industrial IP as a case study." EUROCON 2015-International Conference on Computer as a Tool (EUROCON), IEEE. IEEE, 2015.

[2] Chai, Lingling, Zheng Xie, and Xin'an Wang. "A verification methodology for reusable test cases and coverage based on system verilog." Electron Devices and Solid-State Circuits (EDSSC), 2014 IEEE International Conference on. IEEE, 2014.

[3] Yao, Aihong, Jian Wu, and Zhijun Zhang. "Functional coverage driven verification for TAU-MVBC." Internet Computing for Science and Engineering (ICICSE), 2010 Fifth International Conference on. IEEE, 2010.

[4] http://www.es.ele.tue.nl/premadona/files/akesson01.pdf

[5] https://homepage.cs.uiowa.edu/~ghosh/4-1-10.pdf

[6] http://www.asic-world.com/systemverilog/coverage1.html

[7] http://www.testbench.in/TS_11_TYPES_OF_CODE_COVERAGE.html

[8] Cheng, An-Che, Chia-Chih Yen, and Jing-Yang Jou. "A formal method to improve SystemVerilog functional coverage." High Level Design Validation and Test Workshop (HLDVT), 2012 IEEE International. IEEE, 2012.