At-speed Coverage Improvement for Complex AI SoC

Vishwa Deepika Sripada PG Scholar, Department of Electronics and Communication Engineering, CMR Institute of Technology, Hyderabad, Telangana, (Affiliated to JNTU Hyderabadand Accredited by NBA New Delhi)

Niranjan Reddy Kallem Professor & HoD, Department of Electronics and Communication Engineering, CMR Institute of Technology, Hyderabad, Telangana, (Affiliated to JNTU Hyderabadand Accredited by NBA New Delhi)

Abstract
For Complex SoC's ATPG pattern generations are done in hierarchical approach to reduce run-times and quick iterations. Pattern generations are done at block level and are re-targeted from Chip Top level. During this process, block internal logic will be covered using "InTest" testing technique and interface logic between blocks will be covered using "Extest" testing technique. This paper explains more about how the At-speed Coverage is improved in both Intest and Extest testing. Challenges faced in improving coverage in complex AI SoC's is achieved by adding an additional logic around wrapper cells and making sure without impacting functionality and timing requirements. Here shared wrapper cell approach is followed which work with functional clock frequency. To improve at-speed coverage as well as PPA

Keywords: System On Chip, Automatic Test Pattern Generation, Power Performance Area, Defective Parts Per Million

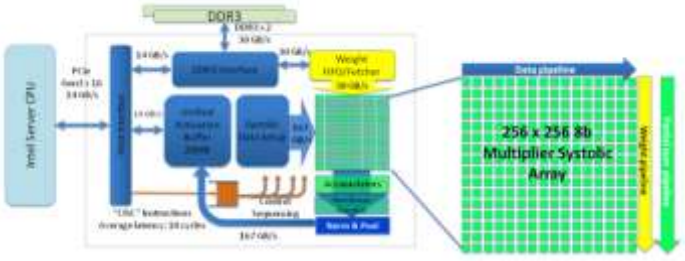
Introduction
As the technology was shrinking from one node to another, more and more logic was added to the design which increased its complexity, reducing controllability and observability of the design. This made it hard to test the manufactured device manually by test engineers. The product proved to be defective even though it passed the 'quality test'. This made designers to introduce "Design for testability", a design phase early in the design cycle of a chip, in which logic is added to improve controllability, observability and thereby increasing testability of the design. This includes many methodologies like scan methodology, built in self-test methodology, JTAG, IJTAG etc. Testability includes both functional and delay testing which is abstracted by many fault models like stuck-at fault model, transition delay fault model, path delay fault model, analog fault model etc. The faults represented by these models are covered with the help of above-mentioned methodologies.

The two prominent fault models for at-speed scan testing are the transition fault models and path-delay. The transition fault model represents a gross delay at every gate terminal. Transition fault tests target each gate terminal for a slow-to-rise or slow-to-fall delay fault. Path delay patterns check the combined delay through a predefined list of gates.The two at-speed fault models most widely used today include the path delay model and the transition delay model. Compared to static testing with the stuck-at fault model, testing logic at-speed requires a test pattern with two parts.The first part launches a logic transition value along a path, and the second part captures the response at a specified time determined by the system clock speed. At-speed coverage at SoC level for complex chips is usually done by inserting wrapper cells around IO's of blocks. In general, most of them use dedicated wrapper cell insertion which works with test clock and not covering the exact functional path in fact we are getting an extra new path to meet the timing. We may lose coverage on the real functional path in this case.

In this paper it explains more about how the at-speed Coverage is improved in both Intest and Extest testing. Challenges faced in improving coverage in complex AI SoC's is achieved by adding

an additional logic around wrapper cells and making sure without impacting functionality and timing requirements. For AI kind of complex chips we go for hierarchical implementation. Here we used shared wrapper cells which work with functional clock frequency. By allowing shared wrapper cells we are reducing the dedicated wrapper cell overhead. Below diagram shows the complexity of AI SoC's.

Fig. 1. AI SoC Architecture



Generally, in the AI architectures we see repetitive Cores along with Interface IPs like DDR, PCIE, Ethernet. So, improving coverage in single core will lead to overall SoC Coverage improvement.

Wrapper Cells

In hierarchical implementation, wrapper cells will be added at core level to get the controllability and observability of primary inputs and outputs of core. Generally, two types of wrapper cells exist, Dedicated wrapper cell and Shared wrapper cell. As in the complex chips we go with hierarchical implementation these wrapper cells will allow us to get the interface covered using Intest and Extest

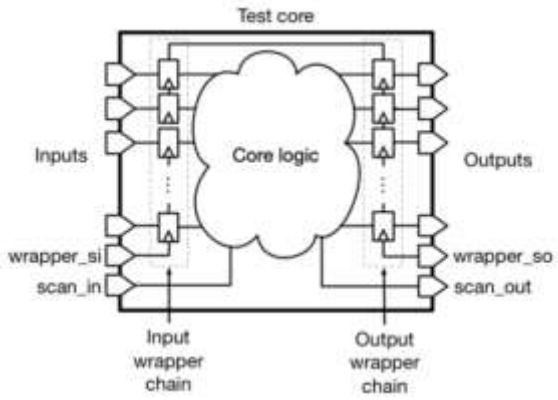


Fig. 2. Wrapper Cell Placement

A. Dedicated Wrapper Cell: Inserting dedicated wrapper cell will break the real functional path and allows to test only stuck at faults in the interface. New timing paths arises which are not the real functional paths.

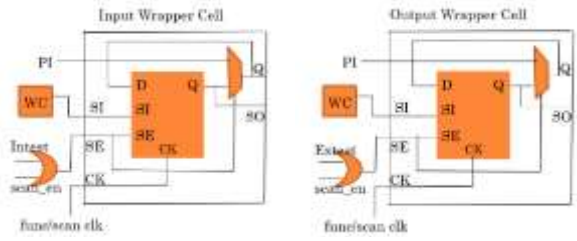


Fig. 3. Dedicated Wrapper Cell

B. Shared Wrapper Cell: Inserting shared wrapper cells will allow us to check the real functional path.

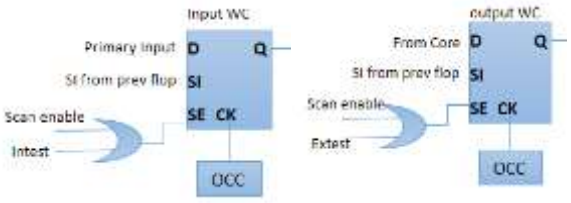


Fig. 4. Shared Wrapper Cell

Analysis of Atspeed Coverage Loss

Coverage loss is observed in many ways like CDC paths, False paths and Multi cycle paths. This is expected as per the design. There are few methodologies to get the coverage on CDC, MCP paths too. In addition, while going with hierarchical implementation of complex chips coverage loss seen due to wrapper cells will also have impact on overall coverage. As inserting dedicated wrapper cells will not give feasibility to check real functional path. Mainly transition faults on interface logic are not getting covered. If we go with shared wrapper cells we will have feasibility to get the coverage on interface paths as they are real functional paths

In shared wrapper cells, as shown in Fig.5. the input wrapper cell will always see SI path in Intest mode, as its SE pin is driven by intest mode control signal.(viz high in intest mode).To get the at-speed coverage on the combinational logic which is sitting in the fanout of input wrapper cell we need transition in this input wrapper cell at functional frequency. With the existing logic, transition in the flop is seen on SI pin but timing the SI pin from previous scan flop Q pin with functional frequency is not possible. (It is an timing critical path). To gain the transition by avoiding this timing path we have added some extra logic on each wrapper cell SI pin and making sure this is not affecting the functionality and new timing path is easy to close the timing.

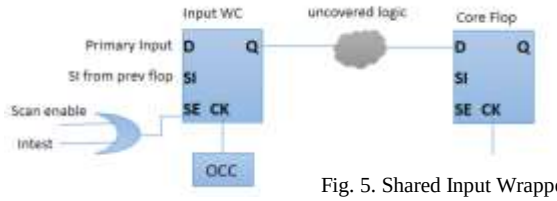


Fig. 5. Shared Input Wrapper Cell

Similarly same issue is seen on output wrapper cell in extest mode at-speed testing. We are adding same logic on output wrapper cell too. Below is the figure showing output wrapper cell structure.

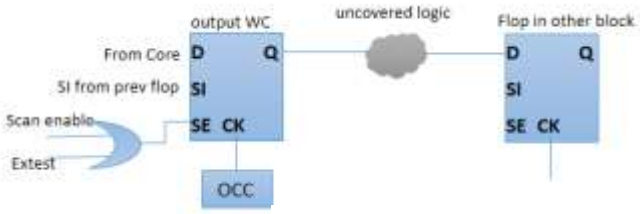


Fig. 6. Shared Output Wrapper Cell

This paper will propose a methodology to cover transition faults in interface logic with the help of shared wrapper cells.

Proposed Methodology

With the additional logic on top of shared wrapper cells the uncovered transition faults are getting covered. In the input wrapper cell as we always see SI pin, with the help of same pin we are enabling transitions by adding an inverter and multiplexer.

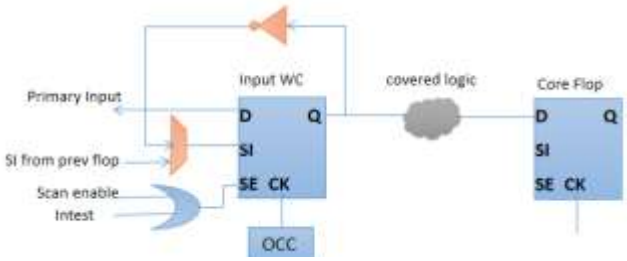


Fig. 7. Shared Input Wrapper Cell

As the new path is a feedback path there will not be any complexity to meet the timing with functional clock frequency.

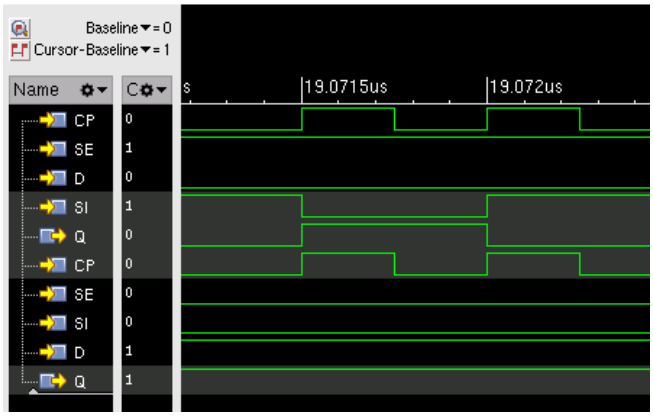


Fig. 8. Transition in Input Wrapper Cell

As shown in the above waveform, in capture window SI pin is toggling twice i.e rise to fall, fall to rise as per our requirement. So, with this transition we are able to get the coverage on the fanout of this input wrapper cell. Similar logic is added for the output wrapper cell to get the coverage on the interface logic between the blocks in Extest mode.

Fig. 9. Shared Output Wrapper Cell

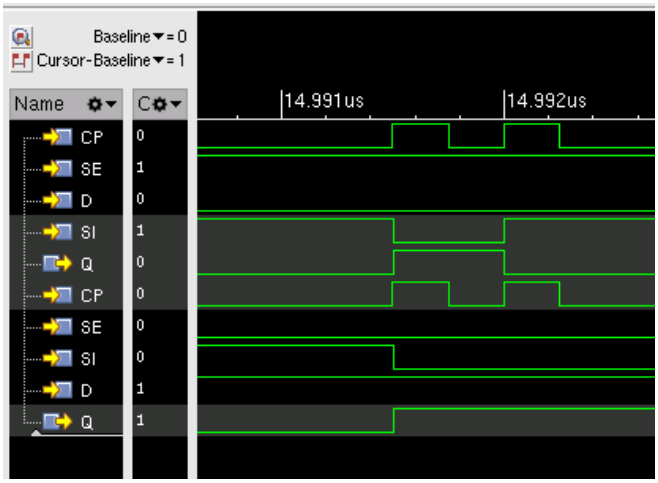


Fig. 10. Transition in Output Wrapper Cell

As shown in the above waveform, in capture window SI pin is toggling twice i.e rise to fall, fall to rise as per our requirement. So, with this transition we are able to get the interface coverage on the fanout of this output wrapper cell.

Experimental Results

Design Details:

TABLE I

Block A Design Details

Block A	Area (sq um)	Inputs	Outputs	Area Overhead %
Initial Design	253718.803	4932	4820	-
Design after additional Logic	254367.399	4932	4820	0.25

Above table shows the comparison for design before and after adding additional logic on wrapper cells. The area overhead seen here is due to adding inverter and multiplexer on each wrapper cell. With the minimal area overhead, we could improve at-speed coverage.

At-speed Coverage Details:

In the below table II , fault count got increased on newly added logic which gives more controllability for input wrapper cells to do transition. Which in turn increased the test coverage with minimum increase in pattern count.

TABLE II

Block A Intest Coverage Results

Intest Mode				
Block A	**Fault count**	**Test Coverage**	**Fault Coverage**	**Pattern Count**
Initial Design	8219733	87.59%	86.84%	16384
Design after additional Logic	8417521	88.76%	88.03%	17088

TABLE III

Block A Incremental Extest Coverage Results

Incremental Extest Mode			
Block A	**Test Coverage**	**Fault Coverage**	**Pattern Count**
Initial Design	87.59%	86.85%	4
Design after additional Logic	89.37%	88.63%	44

As lack of controllability on output wrapper cells in Initial design, we didn't see any coverage improvement in Extest mode. Once after adding the additional logic enabled controllability on output wrapper cells to do transition and this gave coverage on Register to Output path.

SoC Level Design Details:

TABLE IV

SoC Design Details

AI SoC	No. of Blocks	Total IP level		Area (sq um)	Area Overhead %
		Inputs	Outputs		
Initial Design	196	966672	944720	49728885.39	-
Design after additional Logic	196	966672	944720	49856010.2	0.255635764

we are adding similar logic for shared wrapper cells of all the Blocks in SoC, above are the Total IP IO details and SoC level area overhead.For this kind of complex SoC with less no of unique blocks as most of the blocks are repetitive,so improvement in single core will definitely have impact on the SoC level cumulative coverage.

SoC Cumulative At-speed Coverage Details:

TABLE V : SoC Level Cumulative Coverage

Cumulative Coverage of SoC			
AI SoC	Fault count	Test Coverage	Fault Coverage
Initial Design	1610503972	88.75%	88.02%
Design after additional Logic	1649834116	89.75%	88.99%

With this methodology we could achieve 1% improvement in cumulative at-speed coverage.

Conclusion

The proposed methodology to improve at-speed coverage by adding two gates on each wrapper cell will leads to a new timing path. As the new timing path is a feedback path there will not be any complexity to meet the timing with functional clock frequency. The main advantage of this proposal is to improve the coverage in both the Intest and Extest Modes by increasing the controllability.

Above experimental results shown are for a single block. By adding the similar logic to all the blocks in an SoC will improve the cumulative coverage. The maximum possible coverage will reduce the DPPM(Defective parts per Million) count.

References

1. A. Bosio, P. Girard, S. Pravossoudovich, P. Bernardi and M. S. Reorda, "*An efficient fault simulation technique for transition faults in non-scan sequential circuits*," *2009 12th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, 2009
2. V. Vorisek, T. Koch and H. Fischer, "*At-speed testing of SOC ICs*," *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, 2004
3. P. Kumar Datla Jagannadha *et al.*, "*Advanced test methodology for complex SoCs*," *2016 IEEE International Test Conference (ITC)*, 2016

4. J. Zhang, X. Huang, W. Cai and H. Weng, "*Improved delay fault coverage in SoC using controllable multi-Scan-Enable*," *2010 10th IEEE International Conference on Solid-State and Integrated Circuit Technology*, 2010
5. J. Chauhan, C. Panchal and H. Suthar, "*Scan methodology and ATPG DFT techniques at lower technology node*," *2017 International Conference on Computing Methodologies and Communication (ICCMC), 2017*

