IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Enhancing SoC Testability with Hierarchical DFT and Core-Wrapping Approaches

*[1]V. Rajitha Rani, [2]Mamatha Samson

*[1]Research Scholar, [2] Professor

Department of Electronics and Communication Engineering
JNTU, Hyderabad,India

Abstract: Design for Test (DFT) has become a critical component of modern System-on-Chip (SoC) design due to increasing chip complexity. Addressing the challenges of generating test patterns for large-scale devices requires efficient solutions, such as Hierarchical DFT. This approach enables core-level DFT insertion and pattern generation, which is mapped to the top level, enhancing core reusability and reducing development time for SoCs with numerous IPs. This work focuses on developing a Hierarchical DFT flow using core-wrapping methodologies. Key elements include implementing dedicated and shared wrapper chains, integrating test compression logic, and performing Automatic Test Pattern Generation (ATPG) on DFT-inserted netlists. Internal and external mode patterns are generated and stored in PATDB (pattern database) format, with fault data and core descriptions saved in .tcd (tessent core description) files for retargeting. At the SoC level, a top-level wrapper consolidates multiple core instantiations with minimal top-level logic, enabling test pattern generation in Standard Test Interface Language (STIL) format for verification and silicon production tests. Considering the inherent complexity of Hierarchical DFT flows, optimizing key metrics such as test coverage, runtime, test duration, design area, development effort, and verification workload is essential. Achieving the right balance across these factors ensures an efficient and scalable solution for modern SoC designs. This work systematically evaluates different configurations and strategies to identify the most effective hierarchical DFT implementation, addressing the unique challenges posed by increasing design complexity and scalability demands in contemporary semiconductor technologies.

Key Words **-** Design for Test,ATPG, Coverage analysis, Wrapper, Intest, Extest.

I.INTRODUCTION

A major design challenge for modern SoCs is handling the impact of large-scale designs on EDA tools and workflows. To address this, design flows utilize hierarchical boundaries, enabling the partitioning of designs into manageable physical cores [1],[2],[3]. These cores are then integrated at the SoC level. This core-based methodology enhances EDA tools' efficiency. It facilitates the parallel development of multiple design blocks, significantly improving overall design throughput [4],[5] and productivity in front-end and back-end flows.
The same hierarchical principles can be effectively applied to the design-for-test (DFT) [6] process as shown in Fig 1. By incorporating essential design infrastructure, each core can become “DFT-complete” before SoC-level integration [7],[8],[9]. During integration, individual cores or groups of cores can be tested in their *Internal Test Mode*, while *External Test Mode* targets the logic outside the cores. This segmentation simplifies the overall SoC testing process, significantly reducing pattern generation runtime and minimizing the memory footprint of DFT tools, leading to enhanced efficiency and scalability in pattern generation and test execution.

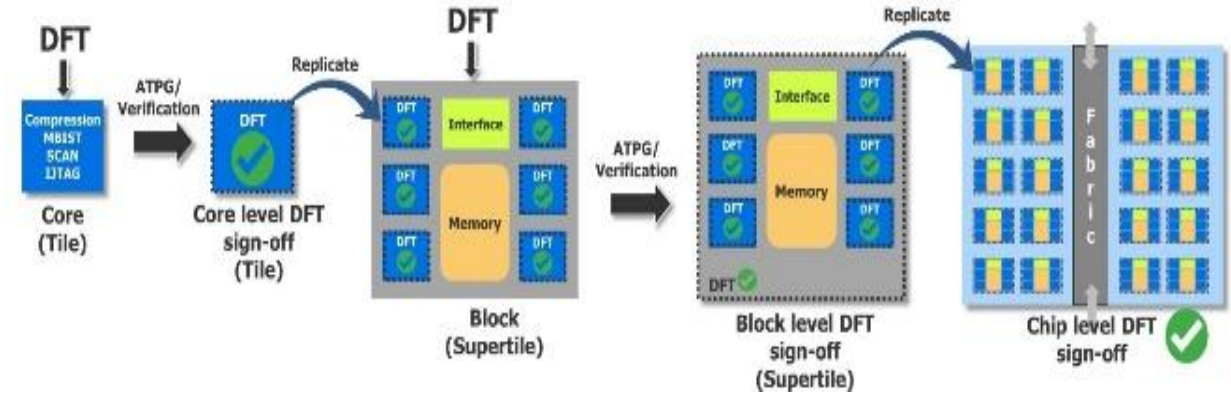


Fig 1: Tessent hierarchical DFT allows for complete DFT sign-off at different levels of design hierarchy.

A core, also known as an IP core [10],[11], is a fundamental building block in SoC design provided by design companies, along with corresponding test solutions. SoC designers enable test access to these cores, embedding them with wrapper chains [12],[13],[14] and test logic to address hierarchical testing challenges. This minimizes core-level test complexity [15],[16],[17],[18] and reduces the top-level pin count of the SoC. This paper highlights the significance of Hierarchical DFT techniques [19],[20], utilizing wrapper chains to tackle large SoC testing challenges. These techniques improve ATPG runtime, memory efficiency, and pin count, accelerating time-to-market.

II.RELATED WORK

2.1.Purpose of Wrappers

- Design isolation: It isolates the boundary between the core & the external logic.
- X-Blocking: It blocks the unnecessary X propagation into the core from the surroundings.
- Test data reuse: This makes Pattern Retargeting possible means we can apply the previously generated test patterns to the cores from the top level.
- Observability & Controllability: Used to control the IO ports of the core (Controllability) & used to observe the IO ports of the core (Observability) [21].

The hierarchical approach offers significant advantages when the ATPG tool supports pattern retargeting. This capability allows patterns to be generated at the core level and seamlessly mapped to SoC-level pins. As a result, only the core netlist needs to be loaded, minimizing the memory footprint. Additionally, the same pattern set can be reused across multiple identical core instances in the SoC without increasing the pattern count. Advanced retargeting can also merge patterns from different cores, creating a compact and efficient pattern set optimized for SoC-level testing.

With foundational hierarchical DFT infrastructure—such as wrapper chains, gray-box netlists, and pattern retargeting—the efficiencies seen in front-end and back-end design flows can also be realized in DFT processes. The benefits are substantial: a large SoC that traditionally requires over 100GB of memory for ATPG can see this reduced to less than 10GB using a hierarchical approach. Similarly, ATPG runtimes, which could previously span days or weeks, can be shortened to mere hours, drastically improving the scalability and practicality of testing complex SoCs.

III.PROPOSED METHOD

Figure 2 illustrates a hierarchical DFT methodology where top-level pins are shared between individual core-level compressor logic and top-level compressor logic, reducing the overall chip pin count required for testing. This configuration allows for testing individual cores separately or in parallel, significantly decreasing test time. Scan insertion is performed at the core level, and when these cores are assembled at the top level, scan chains can be connected in one of two ways: concatenated or direct to I/O. In the concatenated approach, scan chains from one block are sequentially linked with chains from another, streamlining connectivity.

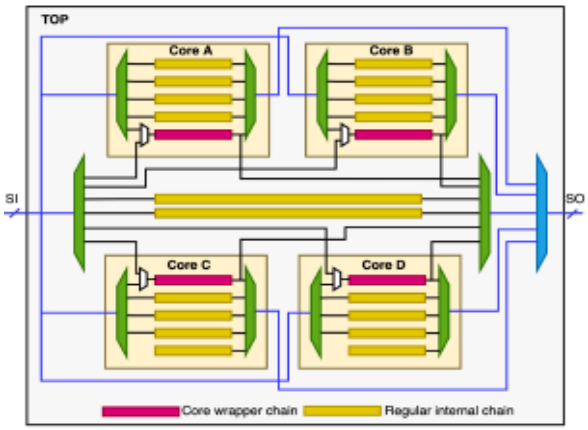


Fig 2: Hierarchical DFT Flow

To implement a hierarchical (bottom-up approach) DFT methodology, one or more wrapper chains must be added to each core. These wrapper chains, conceptually similar to an IEEE 1500 [10] boundary scan chain, define the boundary between internal and external test modes (analogous to INTEST and EXTEST). In Internal mode, the wrapper chains isolate the core from external activity, enabling controllability at core inputs and observability at core outputs. In External mode, the wrapper chains reverse their function, allowing observation of core inputs and control from core outputs to test logic outside the core.

Wrapper chains can include two types of wrapper cells: **shared** and **dedicated**. A shared wrapper cell is an existing functional flip-flop in the design that also serves as a wrapper cell, requiring no additional logic—just identification and integration into the wrapper chains. A dedicated wrapper cell, however, is an additional component, typically consisting of a multiplexer and flip-flop, which adds an area and potentially introduces delay to the functional path.

Additionally, wrapper chains enable the creation of a simplified core netlist, often referred to as a scan shell, interface logic model, or gray box. This netlist includes only the wrapper chains and the logic between them and the core's I/O and is all that is needed for external test mode. By excluding most of the core logic, the resulting SoC-level netlist is significantly smaller in memory footprint compared to the full-chip netlist.

3.1.Test wrapper modes

Inward-facing or INTEST mode

In INTEST mode, the inputs to the partition are driven by the input wrapper cells, and the outputs are captured through the output wrapper cells as shown in Fig 3. To isolate the testing of the partitioned core, the scan chain outside the core is disabled. This setup allows efficient testing of the partitioned core using ATPG (Automatic Test Pattern Generation). During the capture phase, the input wrapper cells are shifted using a dedicated input wrapper scan enable signal. This mechanism prevents the capture of unknown values (X-states) originating from outside the partition. Simultaneously, the output wrapper cells record the internal state of the partition, ensuring accurate test coverage.

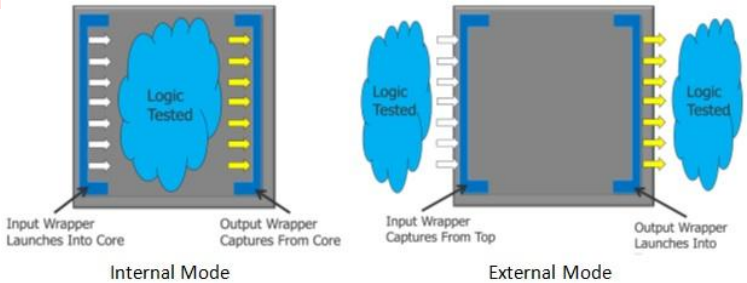


Fig 3: Inward Facing (INTEST) Mode and Outward Facing (EXTEST) Mode

Outward-facing or EXTEST mode

In EXTEST mode, the wrapper cells are enabled and configured to drive and capture data external to the design as shown in Fig 3. In this mode, internal scan chains are effectively bypassed, disabling their operation. This bypassing contributes to a reduction in ATPG test time. EXTEST mode is particularly useful for testing the top-level logic that interfaces between the partitioned design and the unwrapped logic. During the capture phase, the input wrapper cells outside the partition capture values, while the output wrapper cells are shifted using a controlled mechanism. This prevents the capture of unknown values (**X-states**) from the partition's un-driven internal scan chains, ensuring clean and accurate testing.

3.2. Wrapper Cell Structure

Shared wrapper cells

These are preferred for designs with area constraints and less stringent testability requirements. They are often existing functional flops in the design that are also used as wrapper cells. This can reduce test costs, time, and memory/data. However, shared wrapper cells can result in long wrapper chains. The same cell can act as both input shared wrapper cell and output shared wrapper cell as shown in Fig 4.

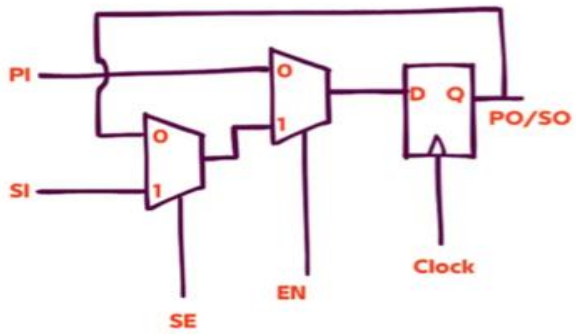


Fig 4: Shared wrapper cell

Input Shared wrapper cell Operation

1. If SE=1 Shift Mode
2. If SE=0 Capture Mode

a. INTEST when EN=1 then retains the previous q value (blocking X)
b. EXTEST when EN=0 then captures value from input (previous port)

Output Shared wrapper cell Operation

1. If SE=1 Shift Mode
2. If SE=0 Capture Mode

a. INTEST when EN=0 then captures value to output (next port)
b. EXTEST when EN=1 then retains the previous q value (blocking X)

Dedicated wrapper cells

These are ideal for critical signals, high-fanout scenarios, and complex scan chain configurations. They are new cells that are added to the design. Dedicated wrapper cells can result in shorter wrapper chains. However, they can increase seller risk and cost. The same cell can act as both input dedicated wrapper cell and output dedicated wrapper cell as shown in Fig 5

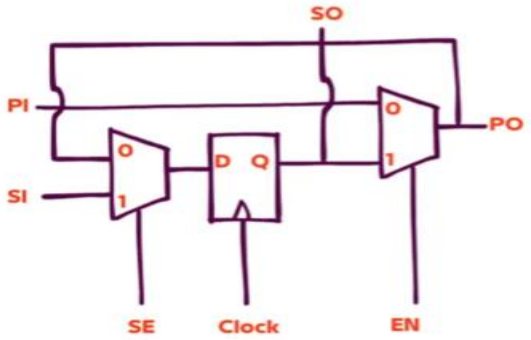


Fig 5: Dedicated wrapper cell

Input dedicated wrapper cell Operation

1. If SE=1 Shift Mode
2. If SE=0 Capture Mode

a. INTEST when EN=1 then retains the previous q value (blocking X)
b. EXTEST when EN=0 then captures value from input (core logic)

Output dedicated wrapper cell Operation

1. If SE=1 Shift Mode
2. If SE=0 Capture Mode

a. INTEST when EN=0 then captures value to output (next port)
b. EXTEST when EN=1 then retains the previous q value (blocking X)

There are certain components which cannot be included in wrapper chains:

1. Input ports which do not drive any sequential cell
2. Clock ports/Bidirectional Ports
3. DFT signals
4. Floating outputs

3.3. Hierarchical DFT advantages

1. Automated tools streamline the assembly of core-level scan chains at the SoC level, simplifying the design process.
2. Balanced core-level scan chains aid tools in efficiently balancing SoC-level chains, enhancing the overall test architecture.
3. Optimizing the use of a limited number of scan chain pins becomes more practical, improving pin utilization.
4. The methodology increases core-level channels, boosting test coverage and accessibility.
5. ATPG runtime and memory demands are significantly reduced, enabling shorter test times and more efficient testing of large designs.

Hierarchical DFT offers advantages but poses challenges, including scan chain disruptions in designs with multiple clock edges, the need for lock-up latches to handle clock domain crossings and timing issues, and the risk of timing problems propagating across interconnected cores. Addressing these challenges requires meticulous design planning and robust timing management to ensure seamless integration and operation of the hierarchical DFT methodology.

This paper examines the crucial role of wrapper cells in hierarchical DFT, detailing their types, functionalities, and implementation in managing access across design blocks. It highlights the hierarchical core wrapper approach's effectiveness through practical results, showcasing its broad applicability in industrial designs.

IV. RESULT ANALYSIS

This section presents the results of applying the proposed methods from Section III to two cores, Core-A and Core-B, within a multi-gate SoC. Core-A is a flat model, while Core-B is a hierarchical design that instantiates Core-A four times. The design employs a hierarchical DFT strategy, with each core encapsulated using an IEEE 1500-based core wrapper.

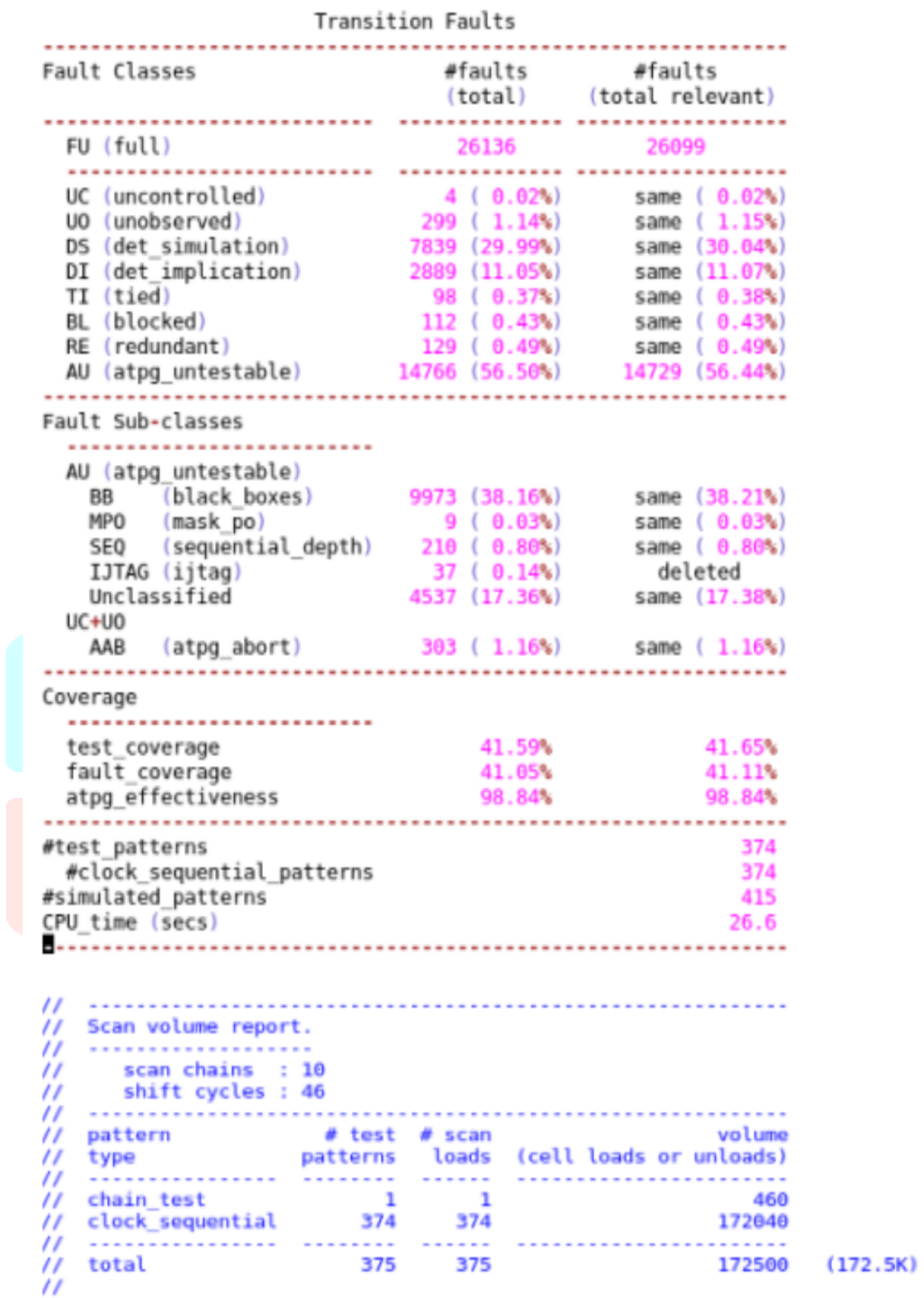
The Netlist/RTL of the mentioned circuits is first converted to DFT inserted netlist using Mentor Graphics Tessent DFT Advisor, Compression using Tessent TestKompress, Synthesis using Synopsys Design Compiler, patterns generated using Tessent Fast Scan, and QuestaSim is used for simulation. This tool takes a test procedure file and DFT inserted netlist as input and produces test patterns in “STIL” format and “Verilog” format. Only Static faults are targeted in this experiment. Figures 6 and 7 show the sample experimental results for the core-A stuck-at fault with wrapper and core-B circuit with wrapper for transition fault respectively.

Tables I and II compare the methodologies based on various parameters, including the number of scan cells per chain, number of chains, pattern count, shift cycles, detected faults, simulation time, cell instances, gate count, test data volume, and test coverage percentages for stuck-at and transition faults for core-A and core-B.

```
// command: report statistics -det
                                Statistics Report
                                Stuck-at Faults
```

Fault Classes	#faults (total)	#faults (total relevant)
FU (full)	44970	44404
UO (unobserved)	13 (0.03%)	same (0.03%)
DS (det_simulation)	35072 (77.99%)	same (78.98%)
DI (det_implication)	5633 (12.53%)	same (12.69%)
PT (posdet_testable)	3 (0.01%)	same (0.01%)
UU (unused)	1150 (2.56%)	same (2.59%)
TI (tied)	294 (0.65%)	same (0.66%)
BL (blocked)	205 (0.46%)	same (0.46%)
RE (redundant)	331 (0.74%)	same (0.75%)
AU (atpg_untestable)	2269 (5.05%)	1703 (3.84%)
Fault Sub-classes		
DI (det_implication)		
SCAN (scan_path)	3328 (7.40%)	same (7.49%)
SEN (scan_enable)	797 (1.77%)	same (1.79%)
CLK (clock)	1450 (3.22%)	same (3.27%)
SR (set_reset)	4 (0.01%)	same (0.01%)
DIN (data_input)	6 (0.01%)	same (0.01%)
MBIST	48 (0.11%)	same (0.11%)
AU (atpg_untestable)		
BB (black_boxes)	56 (0.12%)	same (0.13%)
PC (pin_constraints)	379 (0.84%)	same (0.85%)
rsta_reg/QB_pport C0	168 (0.37%)	
rstb_reg/QB_pport C0	168 (0.37%)	
(Individual pin constraints below threshold)	42 (0.09%)	
(Combined pin constraints)	1 (0.00%)	
TC (tied_cells)	762 (1.69%)	same (1.72%)
/corea_first_insertion_tessent_tdr_sri_ctrl_inst/ltest_en_latch_reg (9970) T1	90 (0.20%)	
(Individual tied cells below threshold)	16 (0.04%)	
(Combined tied cells)	656 (1.46%)	
SEQ (sequential_depth)	505 (1.12%)	same (1.14%)
IJTAG (ijtag)	566 (1.26%)	deleted
Unclassified	1 (0.00%)	same (0.00%)
UC+UO		
AAB (atpg_abort)	13 (0.03%)	same (0.03%)
Coverage		
test_coverage	94.69%	95.95%
fault_coverage	90.52%	91.67%
atpg_effectiveness	99.97%	99.97%

Fig.6: Coverage of core-A circuit with the wrapper for stuck-at fault.



Transition Faults

Fault Classes	#faults (total)	#faults (total relevant)
FU (full)	26136	26099
UC (uncontrolled)	4 (0.02%)	same (0.02%)
UO (unobserved)	299 (1.14%)	same (1.15%)
DS (det_simulation)	7839 (29.99%)	same (30.04%)
DI (det_implication)	2889 (11.05%)	same (11.07%)
TI (tied)	98 (0.37%)	same (0.38%)
BL (blocked)	112 (0.43%)	same (0.43%)
RE (redundant)	129 (0.49%)	same (0.49%)
AU (atpg_untestable)	14766 (56.50%)	14729 (56.44%)
Fault Sub-classes		
AU (atpg_untestable)		
BB (black_boxes)	9973 (38.16%)	same (38.21%)
MPO (mask_po)	9 (0.03%)	same (0.03%)
SEQ (sequential_depth)	210 (0.80%)	same (0.80%)
IJTAG (ijtag)	37 (0.14%)	deleted
Unclassified	4537 (17.36%)	same (17.38%)
UC+UO		
AAB (atpg_abort)	303 (1.16%)	same (1.16%)
Coverage		
test_coverage	41.59%	41.65%
fault_coverage	41.05%	41.11%
atpg_effectiveness	98.84%	98.84%
#test_patterns		374
#clock_sequential_patterns		374
#simulated_patterns		415
CPU_time (secs)		26.6

```
//  ---------------------------------------------------------
//  Scan volume report.
//  -------------------
//     scan chains  : 10
//     shift cycles : 46
//  ---------------------------------------------------------
//  pattern            # test  # scan                volume
//  type             patterns   loads  (cell loads or unloads)
//  ---------------  --------  ------  ----------------------
//  chain_test              1       1                     460
//  clock_sequential      374     374                  172040
//  ---------------  --------  ------  ----------------------
//  total                 375     375                  172500   (172.5K)
//
```

Fig.7: Coverage of core-B circuit without the wrapper for transition faults.

Table 4.1: Comparison table for Stuck-at fault with and without the wrapper

	Benchmark circuit	core-A		core-B	
	Technique	without_wrapper	with_wrapper	without_wrapper	with_wrapper
	Parameter				
Stuck-at faults	Scan cells per Chain	49 or 50	26 or 27	46 0r 45	28 or 29
	Number of chains	13	5	10	5
	Test Coverage	95.91%	95.95%	48.28%	94.20%
	Test Data Volume(bits)	186550	175686	103040	35685
	Test time	4619339	4858748	6009587	6491792
	Shift Cycles	50	27	46	29
	No of Faults Detected	43476	44970	33698	81446
	Cell Instances	6005	6190	4892	11311
	Gates	10159	10439	8823	21623

Table 2: Comparison table for Transition fault with and without the wrapper

	Benchmark circuit	core-A		core-B	
	Technique	without_wrapper	with_wrapper	without_wrapper	with_wrapper
	Parameter				
Transition faults	Scan cells per Chain	49 or 50	26 or 27	46 0r 45	28 or 29
	Number of chains	13	5	10	5
	Test Coverage	91.85%	92.14%	41.65%	92.61%
	Test Data Volume(bits)	465400	457310	172500	54288
	Test time	3958903	6963329	5407377	6613562
	Shift Cycles	50	27	46	29
	No of Faults Detected	35990	36892	26136	81446
	Cell Instances	6005	6190	4892	11311
	Gates	10159	10439	8823	21629

V. CONCLUSION

The results emphasize the benefits of using wrappers in hierarchical DFT, including reduced scan cells per chain, fewer scan chains, and enhanced test coverage for both Core-A and Core-B. For Core-A, the stuck-at and transition fault test coverage improved slightly by approximately 1%, as its flat model limits the scope for significant change. In contrast, Core-B experienced a substantial improvement in test coverage, increasing by around 45%, showcasing the effectiveness of the wrapper approach.

Additionally, Core-B achieved a threefold reduction in test data volume, indicating more efficient testing, while Core-A saw minimal changes in this parameter. Test time also decreased by 5% to 10%, depending on the complexity of the glue logic, contributing to overall efficiency. Importantly, the introduction of wrappers resulted in only a negligible area overhead, ensuring the benefits were achieved without significantly impacting the design's physical footprint.

REFERENCES

[1] Wilson Pradeep, Muniswara Vorugu, Vevekanenda Gonugunta," Enhancing At-Speed Testability of Complex Inter-Core IO Interfaces", Proceedings IEEE International Test Conference (ITC), October 2022.
[2] Paolo Bernardi,et all. "Built-In Self-Test Architecture Enabling Diagnosis for Massive Embedded Memory Banks in Large SoCs", Electronics 2024. https://doi.org/10.3390/electronics13020303.
[3] E Renold Sam Vethamuthu, S Sivanantham, R Sakthivel. "Implementation of Hierarchical DFT Approach for Better Testability", 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR), 2018, pp. 1-4,doi: 10.1109/ICETIETR.2018.8529130.
[4] Y. Higami, T. Inamoto, S. Wang, H. Takahashi and K. K. Saluja, "Improving of Fault Diagnosis Ability by Test Point Insertion and Output Compaction," 2023 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC), Jeju, Korea, Republic of, 2023, pp. 1-5, doi: 10.1109/ITC-CSCC58803.2023.10212844.
[5] Dan Trock, Rick Fisette, "Recursive Hierarchical DFT Methodology with Multi-Level Clock Control and Scan Pattern Retargeting", IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.1128-1131, 2016.
[6] V. Rajitha Rani, Mamatha Samson, "Enhancing DFT Performance: The Power of Scan Compression and Lockup Latches", IEEE 4th International Conference on Ubiquitous Computing and Intelligent Information Systems, Dec-2024.
[7] Brion Keller et al., "Efficient testing of hierarchical core-based SOCs", Proceedings IEEE International Test Conference (ITC), pp. 1-10, October 2014.
[8] "Maximized Reuse Core Wrapper Cells", TestMAX DFT User Guide, version T-2022.03.
[9] "Wrapper Cells Analysis", Tessent™ Scan and ATPG User's Manual, Version 2022.1, Revision 24.
[10] IEEE 1500 - http://grouper.ieee.org/groups/1500
[14] V.R. Devanathan, C.P. Ravikumar, V. Kamakoti, "Reducing SoC Test Time and Test Power in Hierarchical Scan Test: Scan Architecture and Algorithms", 20th International Conference on VLSI Design (VLSID'07) IEEE, 2007.
[11] https://blogs.sw.siemens.com/tessent/2016/03/15/hierarchical-dft-how-to-do-more-more-quickly-with-fewer-resources
[12] https://www.edn.com/complex-soc-testing-with-a-core-based-dft-strategy
[13] https://www.electronicdesign.com/home/article/21206502/divide-and-conquer-hierarchical-dft-for-soc-designs
[15] J. Janicki, et ai, "EDT bandwidth management -Practical scenarios for large SoC designs," Proc.International Test Conference, 2013.
[16] c. Papameletis, et aI, "Automated Dff insertion and test generation for 3D- S ICs with embedded cores and multiple towers," Proc. European Test Symposium, 2013.
[17] C. Allsup and K. Chung, "Product How To: DFT strategy for ARM processor-based designs," EDN Magazine, January 22, 2013.
[18] J. Aerts and E. J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs", in Proc. Intl. Test Conf., 1998, pp. 48-457
[19] K. M. Butler, et. al.,"Minimizing Power Consumption in Scan Test-ing: Pattern Generation and DFT Techniques," in Proc. Intl. Test Conf., 2004, pp. 355.364.
[20] Jeff Remmers, Moe Villalba, Richard Fisette, "Hierarchical DFT approach- A case study", IEEE Int. test conference, paper 30.2, pp 856, 2004
[21] V.R. Devanathan, C.P. Ravikumar, V. Kamakoti, "Reducing SoC Test Time and Test Power in Hierarchical Scan Test: Scan Architecture and Algorithms", 20th International Conference on VLSI Design (VLSID'07), IEEE, 2007.