ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

TRANSPARENCY-BASED HIERARCHICAL TEST GENERATION FOR MODULAR RTL DESIGNS

Y. Makris[1], J. Collins[1], A. Orailoğlu[1], P. Vishakantaiah[2]

[1] Reliable Systems Synthesis Lab, MC-0114
CSE Dept., University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA

[2] Intel Corporation
2501, 229th AVE, MS:RA2-401
Hillsboro, OR 97124, USA

ABSTRACT

We discuss a novel hierarchical test generation methodology for RTL designs, based on the concept of modular transparency. We introduce the *channel* notion, a powerful mechanism that captures modular transparency in terms of bijection functions defined on variable bitwidth signal entities. Through a recursive search algorithm, transparency channels are further combined into reachability paths suitable for translating local test vectors for each module into global design test. A *divide & conquer* hierarchical test generation methodology is described, resulting in significant test generation time speed-up and comparable fault coverage and vector count to complete circuit gate-level ATPG.

1. INTRODUCTION

System-On-Chip (SOC) design trends exploit the continuous technology improvements in order to realize large and complex circuits in a modular fashion. Such circuits however, have by far outpaced the capacity of the EDA tools to handle them as monolithic entities. Furthermore, the time-to-market constraints of SOC designs necessitate tools that operate early in the design cycle, on levels higher than the gate-level. Significant efforts are consequently invested in devising hierarchical RTL approaches for accommodating current and future design and test needs.

Hierarchical test generation, as depicted in Figure (1), constitutes a promising alternative to the slow and complex global design gate-level test generation. Local test is quickly generated for each module and translated to global test, applicable at the complete design boundary. While local test generation is highly efficient, the success of such schemes depends heavily on the efficacy of test translation. Exhaustive reasoning on the complete functional space of the surrounding modules to perform vector-by-vector test translation is doomed by the same complexity barrier as global test generation. Transparency behavior is alternatively employed in order to perform fast, symbolic test translation.

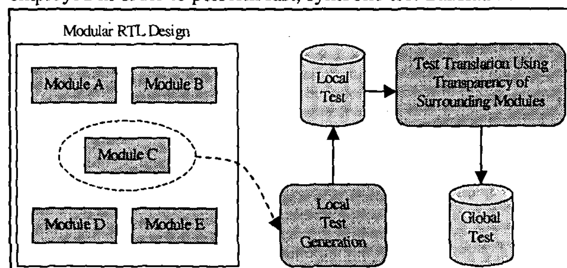

Figure 1. Hierarchical Test Generation.

Section 2 reviews previous work in the area of hierarchical test generation and test translation transparency definition. Section 3 introduces a novel transparency definition through the concept of *channels*, based on which a hierarchical test generation methodology is devised in section 4. Experimental data in support of the proposed scheme is provided in section 5.

2. PREVIOUS WORK

Several hierarchical test generation approaches and distinct transparency definitions have been proposed in the literature. The concepts of *I-Path* and *T-path* for capturing module transparency in terms of identity and transformation functions respectively were introduced in [1]. These *one-to-one* and *onto* functions are limited to equal bitwidth signal entities and only the *I-Path* has been pursued for hierarchical test generation. The path notion was extended in [3] through arbitrary *onto* functions *(S-paths)* for providing stimuli to a module and arbitrary *one-to-one* functions *(F-paths)* for propagating fault responses. A hierarchical test generation scheme and a test result propagation methodology, based on *ambiguity sets,* were introduced in [9,10]. Complexity issues limit the applicability of these methods to small datapath circuits. Test knowledge extraction for hierarchical designs is captured in terms of *modes* in [14], and further combined into test translation paths in [15]. Highly translatable local test is generated in [13] by incorporating global design *constraints* in the test generation process and high-level *architectural information* is used in [6] for similar purposes.

3. TRANSPARENCY CHANNELS

A novel mechanism for expressing transparency behavior of RTL modules in terms of *channels* is introduced in this section and its applicability for constructing test translation paths is discussed.

3.1 Transparency Channel Definition

Transparency channels capture modular transparency in terms of bijection functions between input and output *signal entities.* Channel functions may be *bit-decomposable* (e.g. left rotation) or not (e.g. addition MOD k). Channels are instantiated upon compliance of *conditions* that require a specific *potential* on signal entities. The potential may be either controllability or observability of the signal entity to a set of values. Signal entities may be defined either on the full word bitwidth or on sub-word bitwidths. A succinct definition of the transparency channels is given in Figure (2) along with a few examples of simple RTL modules and associated channels.

0-7803-5482-6/99/$10.00 ©2000 IEEE

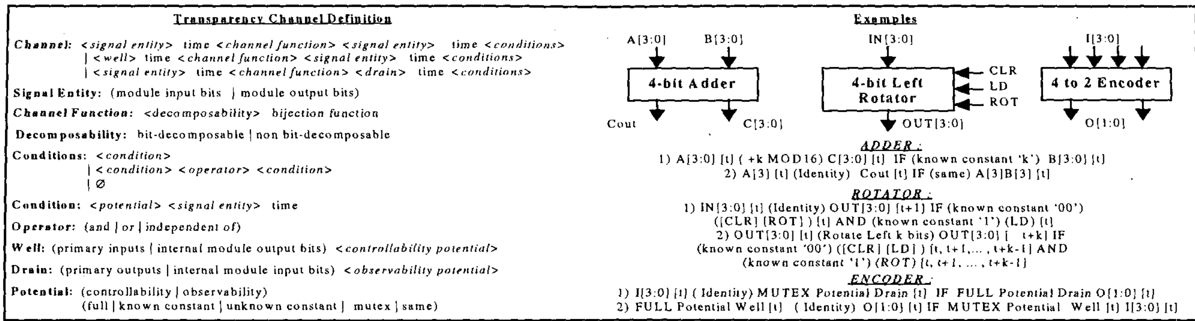

Figure 2. Transparency Channel Definition and Examples.

3.2 From Channels To Paths

Combining channels into test translation paths requires that the rules of Figure (3) be observed. Channels can be combined in series as in Figure (3)(a), provided that the conditions for realizing them do not conflict in space, time or potential. Condition checking is performed in a centralized fashion for all modules and paths in the design. Channels can also be composed in parallel either in the same module or in different modules, as in Figure (3)(b), provided that the conditions do not conflict. In addition, the signal entities of the channels need to remain independent of each other for the composition to be valid. Splitting a channel, in order to accommodate more than one path, is more complex. If the channel is *decomposable,* as in Figure (3)(c), then we can simply follow the bit decomposition and keep the paths apart. Condition conflicts are not an issue here, since only one channel is utilized. However, if the channel is not decomposable, as in Figure (3)(d), we cannot preserve the separation of the paths. The only solution in this case is to create a new path, possibly wider than the sum of the bitwidths of the individual paths, that accommodates both by utilizing the complete channel. These rules guarantee that the *full potential* capability will be preserved after merging or splitting channels.

4. HIERARCHICAL TEST GENERATION

Transparency channels facilitate a powerful hierarchical test generation scheme, resulting in significant test generation time reduction and highly efficient test for modular designs. Local test is generated for each module and subsequently translated through paths of channels into global design test. The proposed hierarchical test generation scheme, shown in Figure (4), is independent of the actual method employed for local test generation for each module. In addition, channel-based translation paths are identified regardless of the local test vectors. As a result, local tests can be modified and enhanced in order to provide higher fault coverage, without invalidating the translation paths. The fault coverage attained by the local vectors is an upper bound to the fault coverage of the globally translated vectors, for each particular module. Therefore, the local test generation process should maximize the number of translatable patterns, possibly by employing approaches that guide the local test generation with global design knowledge, such as [6,13].

A recursive design traversal algorithm is applied for each module in the design, employing transparency channels in order to identify test justification and propagation paths. For each module under test, internal module connectivity is examined and test justification and propagation requirements are defined.

The algorithm traverses the design, backtracking as necessary, in order to satisfy the requirements. While traversing an upstream or downstream module, available channels are probed as to their suitability for providing the required potential. Channels may be combined into wider channels or broken into smaller ones, according to the rules of section 3.2. Reconvergence and feedback loops are considered in order to prioritize the probing of channels, accelerating algorithm convergence. The search ends when *wells* or *drains* of the required *potential* are encountered, satisfying the requirements. The transparency channels and conditions on the test translation paths are reported and subsequently combined into test translation templates for each module. If the requirements of a module cannot be completely satisfied, a set of best-match paths, along with a set of test translation bottlenecks for testability enhancement are provided, as discussed in [8]. An extensive description of the test requirements, the recursive path identification algorithm and its applicability on testability analysis may be found in [7].

The transparency channels on the identified vector justification and response propagation paths are combined into test translation templates that apply the reverse effect of the channel functions on the translation path, performing rapid test translation.

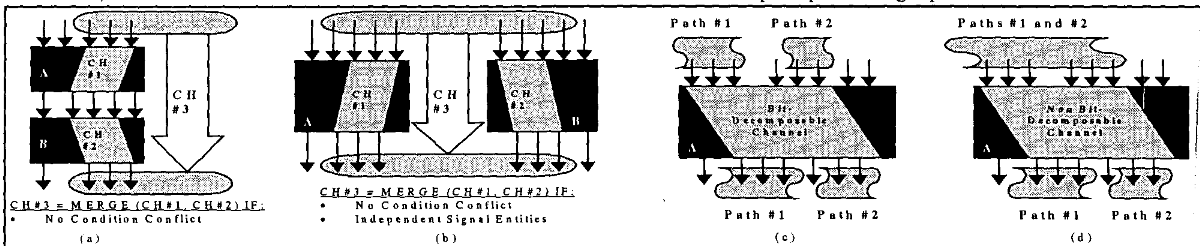

Figure 3. Rules for Combining Channels into Paths.

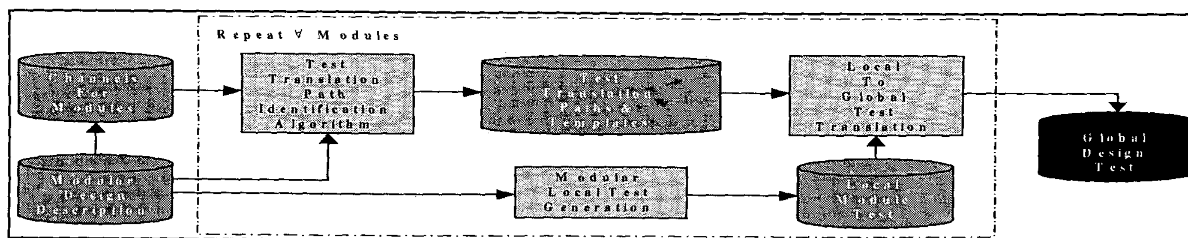

Figure 4. Channel-Based Hierarchical Test Generation.

5. EXPERIMENTAL RESULTS

The three-phase experimental setup of Figure (5), is employed to validate the adequacy of transparency channels for performing test translation and the efficiency of the proposed scheme:

PHASE #1: The RTL description of the complete design is synthesized and full-circuit ATPG is applied on the obtained gate-level view. Global test, along with the test generation time T_F, fault coverage C_F, and vector count V_F, is thus obtained.

PHASE #2: The proposed hierarchical test generation methodology is applied and the test translation paths for each module in the design are obtained in time T_P. The first module is synthesized and gate-level ATPG is applied on it, providing the local test vectors and the test generation time T_1, fault coverage C_1, and vector count V_1. These vectors are translated through the identified translation paths and fault-simulated on the complete circuit gate-level view for all design faults. The test translation time TR_1 and the fault simulation time F_1 are also noted. The process is repeated for each remaining module, targeting only faults that have not been covered by previous global vectors. The results are accumulated and the corresponding time T_S, fault coverage C_S and vector count V_S are obtained. The objective of this phase is to compare this methodology to full circuit ATPG, based on test generation time, fault coverage and vector count.

PHASE #3: $|V_S|$ random patterns are fault-simulated on the original design and the corresponding coverage C_R is obtained. The objective of this phase is to demonstrate that the global patterns generated by the proposed methodology provide higher fault coverage than randomly generated patterns.

HITEC [11], PROOFS [12] and HOPE [5] are used for test generation, fault simulation, and random pattern test generation. A prototype tool called TRANSPARENT (**TRANS**lation **P**ath **A**nalysis **REN**dering **T**est) was employed for identifying test translation paths and performing the actual translation. The experimental setup was applied on three benchmark designs. The first design (MTC100), is a 3-module circuit introduced in [14], with interesting feedback loop behavior. The second design is an 8-bit *shift-&-add* binary multiplier (MUL) comprising 10 modules as described in [4]. The third circuit is a pipelined multiplier accumulator (MAC) for complex numbers, comprising 23 modules, as described in [2]. The test generation time, vector count and fault coverage, along with the number of aborted, redundant and total faults is provided for the full circuit ATPG in Table (1). In Table (2), we provide the time spent by TRANSPARENT on identifying the translation paths and performing the actual test translation, the total local test generation time and the total fault simulation time. The results are accumulated and the total test generation time, vector count and fault coverage are shown and highlighted for comparison purposes. In Table (3), the fault coverage obtained by fault simulating $|V_S|$ random patterns is provided. In all three circuits, the channel-based hierarchical test generation scheme outperformed full circuit ATPG in terms of total test generation time. As shown in Table (4), in the first circuit the reduction was almost at an order of magnitude, while for the other two circuits it was approximately 65%. Table (5) shows that fault coverage slightly increased in the first two circuits, while in the third there was a 2.5% drop approximately. Random vectors achieved significantly lower coverage in all cases. In terms of vector count, a slight increase in the order of 10-15% was observed, as

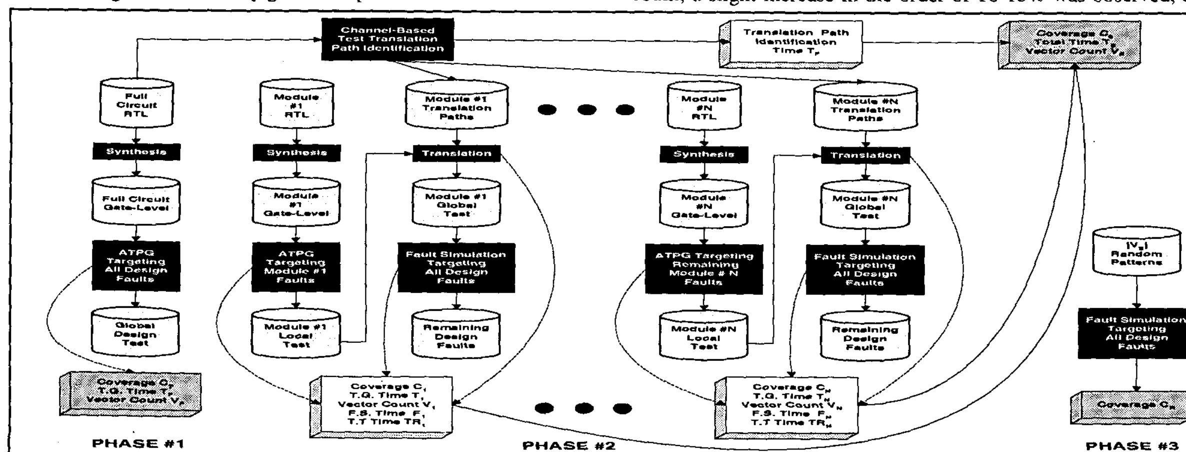

Figure 5. Experimental Setup.

Benchmark Circuit	Number of Modules	T_F (sec)	V_F (vectors)	C_F (faults)	Aborted (faults)	Redundant (faults)	Total (faults)
MTC100	3	3.383	146	1034	21	0	1055
MUL	10	13.58	191	760	64	10	834
MAC	23	21.30	627	27896	238	480	28614

Table 1. Complete Design Gate-Level ATPG Results.

Benchmark Circuit	Number of Modules	$T_P + \Sigma(TR_i)$ (sec)	$\Sigma(T_{Ni})$ (sec)	$\Sigma(F_{Ni})$ (sec)	T_S (sec)	V_S (vectors)	C_S (faults)
MTC100	3	0.30	0.117	0.02	0.437	178	1038
MUL	10	2.80	1.670	0.16	4.630	198	790
MAC	23	5.32	1.250	2.30	8.870	703	27245

Table 2. Channel-Based Hierarchical Test Generation Results.

Benchmark Circuit	Number of Modules	$\|V_S\|$ (vectors)	C_R (faults)
MTC100	3	178	748
MUL	10	198	569
MAC	23	703	22454

Table 3. Random Test Generation Results.

Benchmark Circuit	Gate-Level ATPG	Hierarchical Test Generation
MTC100	3.383 sec	0.44 sec
MUL	13.58 sec	4.63 sec
MAC	21.30 sec	8.87 sec

Table 4. Total Test Generation Time Comparison.

Benchmark Circuit	Gate-Level ATPG	Hierarchical Test Generation	Random Test Generation
MTC100	1034 faults	1038 faults	748 faults
MUL	760 faults	790 faults	569 faults
MAC	27896 faults	27245 faults	22454 faults

Table 5. Fault Coverage Comparison.

Benchmark Circuit	Gate-Level ATPG	Hierarchical Test Generation
MTC100	146 vectors	178 vectors
MUL	191 vectors	198 vectors
MAC	627 vectors	703 vectors

Table 6. Vector Count Comparison.

shown in Table (6). According to these results, transparency channels facilitate a powerful hierarchical test generation scheme that is a superior alternative to full circuit gate-level ATPG.

6. CONCLUSIONS

Size and complexity considerations impede test generation tools that attempt to address modern designs as monolithic entities, revealing the imperative need for hierarchical test generation approaches. However, such approaches require fast and effective mechanisms for translating locally generated vectors into global design applicable test. Towards this end, we introduce a definition that captures test translation capabilities of RTL modules in the form of transparency channels. Channels are further combined into test justification and propagation paths, supporting symbolic test translation without exhaustive reasoning on the complete functional space of the design. A hierarchical test generation methodology is consequently derived, wherein locally generated vectors are translated into global design test utilizing solely transparency behavior. A theoretical analysis along with a set of experimental results reveal that the proposed hierarchical test generation scheme provides significant test generation speed-up, while preserving fault coverage and vector count comparable to complete design gate-level ATPG.

7. REFERENCES

[1] M. S. Abadir, M. A. Breuer, "A Knowledge-Based System for Designing Testable VLSI Chips", *IEEE Design and Test of Computers*, vol. 2, no. 4, pp. 56-68, 1985.

[2] P. Ashenden, *The Designer's Guide to VHDL*, Morgan-Kaufmann Publishers Inc., 1996.

[3] S. Freeman, "Test Generation for Data-Path Logic: The F-Path Method", *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 421-427, 1988.

[4] J.P. Hayes, *Computer Architecture and Organization*, McGraw-Hill, 3rd Edition, 1998.

[5] H. K. Lee, D. S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1048-1058, 1996.

[6] J. Lee, J. H. Patel, "Hierarchical Test Generation under Architectural Level Functional Constraints", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1144-1151, 1997.

[7] Y. Makris, A. Orailoğlu, "RTL Test Justification and Propagation Analysis for Modular Designs", *Journal of Electronic Testing: Theory & Applications*, Kluwer Academic Publishers, vol. 13, no. 2, pp. 105-120, 1998.

[8] Y. Makris, A. Orailoğlu, "DFT Guidance Through RTL Test Justification and Propagation Analysis", *Proceedings of the International Test Conference*, pp. 668-677, 1998.

[9] B. T. Murray, J. P. Hayes, "Hierarchical Test Generation Using Precomputed Tests for Modules", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 6, pp. 594-603, 1990.

[10] B. T. Murray, J. P. Hayes, "Test Propagation through Modules and Circuits", *Proceedings of the International Test Conference*, pp. 748-757, 1991.

[11] T. Niermann, J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits", *Proceedings of the European Conference on Design Automation*, pp. 214-218, 1992.

[12] T. Niermann, W. T. Cheng, J. H. Patel, "PROOFS: A Fast, Memory Efficient Sequential Circuit Fault Simulator", *Proceedings of the the 29th ACM/IEEE Design Automation Conference*, pp. 535-540, 1990.

[13] R. S. Tupuri, J. A. Abraham, "A Novel Test Generation Method for Processors using Commercial ATPG", *Proceedings of the International Test Conference*, pp. 743-752, 1997.

[14] P. Vishakantaiah, J. A. Abraham, M. S. Abadir, "Automatic Test Knowledge Extraction From VHDL", *Proceedings of the the 29th ACM/IEEE Design Automation Conference*, pp. 273-278, 1992.

[15] P. Vishakantaiah, J. A. Abraham, D. G. Saab, "CHEETA: Composition of Hierarchical Sequential Tests using ATKET", *Proceedings of the International Test Conference*, pp. 606-615, 1993.