Available online at www.sciencedirect.com

ScienceDirect

INTEGRATION, the VLSI journal 40 (2007) 315–325

INTEGRATION the VLSI journal

www.elsevier.com/locate/vlsi

On the identification of modular test requirements for low cost hierarchical test path construction

Yiorgos Makris[a,*], Alex Orailoglu[b]

[a]Electrical Engineering Department, Yale University, New Haven, CT 06520, USA
[b]Computer Science & Engineering Department, University of California, San Diego, La Jolla, CA 92093, USA

Received 4 September 2003; received in revised form 22 November 2005; accepted 11 January 2006

Abstract

We discuss a novel method for identifying test requirements of modules in a hierarchical design in order to facilitate the construction of cost-effective hierarchical test paths. Unlike current practices, which construct very general paths capable of justifying all vectors and propagating all responses to and from each module in the design, test requirements in our method are defined as a set of fine-grained input and output bit clusters and pertinent symbolic values. These test requirements reflect the inherent connectivity and regularity of each module and, when supported by corresponding hierarchical test paths, they guarantee complete testability of the module. Their key advantage is that they are not fully specified test vectors and, therefore, they do not require a computationally expensive search algorithm to satisfy from the primary inputs and outputs of the circuit. At the same time, they are also not arbitrarily general and, therefore, they do not impose overly strenuous transparency requirements on the surrounding modules, which could require excessive design-for-testability hardware. In essence, they combine the generality required for fast hierarchical test path construction with the precision necessary for minimizing the incurred cost, thus fostering cost-effective hierarchical test.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Hierarchical test; Test requirements; Transparency; Design for test

1. Introduction

Size and complexity considerations, along with the modular nature of the core-based design trend, have accentuated the importance of hierarchical test methodologies [1–5]. Hierarchical test employs a divide and conquer approach to reduce the size of the test generation problem and, thus, to improve fault coverage and test generation time. These benefits, however, come at the cost of necessitating access from the primary inputs and outputs of the circuit to the boundaries of each module. A simple and straightforward method for establishing such access is through the use of a dedicated test bus [6,7]. Yet the area and performance overhead of a test access bus is not always tolerable. Therefore, a lot of effort has been invested in utilizing the existing functionality of the design to establish module access through *hierarchical test paths*. As depicted in Fig. 1(a), these paths establish transparent access to the module under test (MUT), through the upstream and downstream logic. During hierarchical test path construction, the MUT is treated as a black box. Implicitly, it is assumed that all possible vectors and responses need to be justified and propagated, although in practice this is almost never the case. Furthermore, transparent accessibility to all modules is rarely inherently present in a design. As a result, significant design-for-test (DFT) overhead is incurred to establish these paths [8,9], limiting the cost-effectiveness and applicability of hierarchical test.

In an effort to reduce this DFT cost, two directions have been examined. Along the first direction, several research efforts [1,2,9–11] have been invested in defining, extracting, and utilizing inherent design transparency. Along the second direction [4], inherent functionality of the surrounding logic is used to constrain local test generation, rendering translatable test. Not much attention has been

*Corresponding author. Tel.: +1 2034321203; fax: +1 2034320593.
E-mail addresses: yiorgos.makris@yale.edu (Y. Makris), alex@cs.ucsd.edu (A. Orailoglu).

0167-9260/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2006.01.002

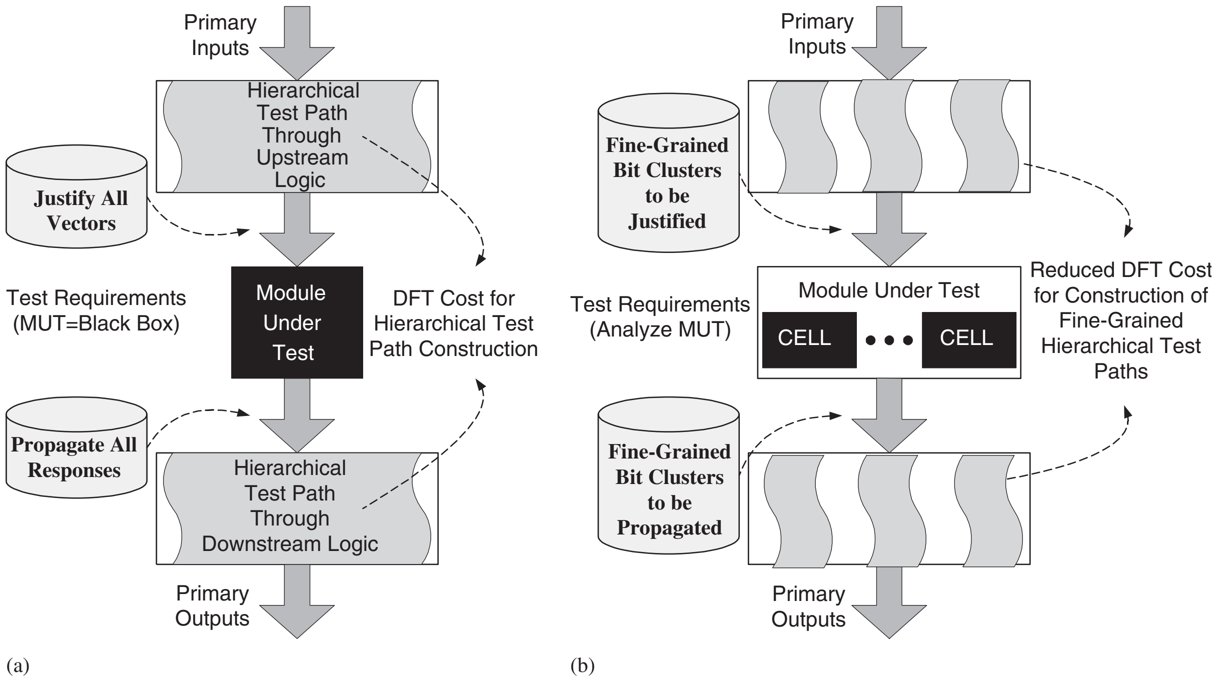

Primary Inputs
Hierarchical Test Path Through Upstream Logic
Justify All Vectors
Test Requirements (MUT=Black Box)
Module Under Test
DFT Cost for Hierarchical Test Path Construction
Propagate All Responses
Hierarchical Test Path Through Downstream Logic
Primary Outputs
(a)
Primary Inputs
Fine-Grained Bit Clusters to be Justified
Test Requirements (Analyze MUT)
Module Under Test
CELL
CELL
Reduced DFT Cost for Construction of Fine-Grained Hierarchical Test Paths
Fine-Grained Bit Clusters to be Propagated
Primary Outputs
(b)

Fig. 1. Granularity of test requirements.

paid, however, to a third option, namely moderating DFT cost through an informed definition of the test requirements of each MUT, rather than treating the MUT as a black box. This idea is depicted in Fig. 1(b), where the internal connectivity of the MUT is examined and its test requirements are defined as input and output bit clusters. This results in several narrow hierarchical test paths instead of a single coarse path, thus increasing the probability of their inherent existence in the design.

In this paper, we assess the impact of test requirement granularity on the cost of hierarchical test path construction. Subsequently, we propose a methodology for reducing this cost by fine-tuning the generality of test requirements and the number and the granularity of necessary hierarchical test paths. Furthermore, regular module connectivity is used to define compact and parametrized test requirements. *Cell-level analysis* and *symbolic path composition* result in the definition of test requirements as a set of input and output *bit clusters*. Cell-level analysis supports compactness, while bit clusters enhance accuracy and symbolic paths guarantee generality. Although we only consider combinational modules, the proposed method constitutes the first step towards low cost hierarchical test path construction based on test requirement analysis.

The remainder of this paper is organized as follows. Related work is discussed in Section 2. The proposed method is then presented in detail in Section 3. Specifically, the severity imposed by test requirements on hierarchical test path construction is first examined in Section 3.1. The identification of appropriate test requirements is introduced in Section 3.2 and the appropriate cell granularity is discussed in Section 3.3. Adjustment to cell connectivity is examined in Section 3.4. Examples are given in Section 4 and severity metrics along with results are provided in Section 5.

2. Related work

Hierarchical test path construction typically employs *transparency*. Transparency has been defined as *surjective* functions for justifying test vectors and *injective* functions for propagating test responses. Surjective and injective functions are referred to in the literature as *S-Paths* and *F-Paths* respectively [10], while bijective functions, satisfying both properties, are referred to as *I-Paths* and *T-Paths* [11]. Several variations of surjective, injective, and bijective functions, including *Ambiguity Sets* [1], *Transparency Modes* [2], and *Transparency Properties* [9], have also been used in order to improve efficiency and reduce cost.

Although test requirement identification has been examined in related fields, the objective for hierarchical test path construction is different than the traditional concept of C-Testability [12] commonly used in built-in self-test and iterative logic array test. The objective of test requirement identification for BIST [13–16], for example, is to derive a compact test set that can be easily generated on chip. The objective of test requirement identification for ILA test [17–19] is to exploit regularity and combine test application across cells. In contrast to these approaches, the objective of the proposed methodology is to identify test requirements that reduce the *severity* imposed on hierarchical test path construction and the corresponding DFT overhead.

3. Proposed method

3.1. Hierarchical test path severity

Hierarchical test methods employ symbolic paths for performing local to global test translation. Symbolic paths, however, impose strenuous functionality requirements on surrounding modules, directly impacting the incurred DFT overhead. As a result, hierarchical test methods are criticized for the unnecessary generalization of test requirements. But is it always the case that symbolic paths impose more strenuous requirements than a set of exact vectors?

Answering this question requires an understanding of the severity incurred by an exact test vector set on hierarchical test path construction. Given a set of k-bit test vectors and depending on the number and the distribution of values appearing on each subset of these k-bits, certain restrictions are imposed on the number of primary inputs required and the degrees of freedom necessary between them. Bits that are always equal or always inverse throughout the test set only require one free variable. The free variables required are not always equal to the vector width. But at a certain set density point, the full width is required; essentially, if more than half of the possible values are in the set, no bit can be inferred, necessitating k free variables.

Consider for example the 2-input module of Fig. 2(a) and the provided alternative test sets. Test requirements for each bit may be either a constant '0' or '1' value symbolically represented by 'V', or a free variable represented by 'A'. Bits that are not required in a test set are represented by 'X'. For TS_1, a hierarchical test path with one free variable, A_1, at its inputs is sufficient to satisfy the test requirements. This is also valid for test set TS_2. For TS_3 and TS_4, a hierarchical test path with one free variable, A_1, and a constant at its inputs is again sufficient. Once a test set with three vectors is reached, however, such as TS_5, a 2-bit hierarchical test path with no bit correlation is necessary. This is almost as severe as requiring two free variables, A_1 and A_2. A 2-bit path is what hierarchical test methods establish in this case.

Similarly, for the 3-input module of Fig. 2(b), TS_6 can be satisfied with two free variables, A_1 and A_2, at its inputs. However, TS_7 requires three free variables, A_1, A_2, and A_3. A careful observation reveals that unlike in TS_6, in TS_7 every 2-bit subset obtains more than half of the possible values, thus necessitating a 2-bit hierarchical test path with uncorrelated bits. Since this holds for every subset, the severity is equivalent to a full 3-bit path.

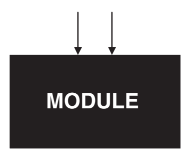
Evidently, there is a threshold for the number and distribution of vectors in a test set, over and above which the severity of the corresponding hierarchical test path is equal to that of the full symbolic path. Generalizing the above observations leads to the following condition:

- Hierarchical test path severity threshold: The hierarchical test path severity of a set of k-bit vectors is equivalent to a k-bit symbolic path if every subset of bits obtains more than half of the possible values.

This condition signifies when exact test vectors can be relaxed into symbolic test requirements, providing a starting point for the proposed test requirement identification.

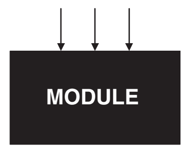
3.2. Test requirement identification

The proposed methodology targets each basic *cell* in a module and requires that free variables be justified to all the inputs and propagated from all the outputs. These symbolic requirements are translated into module inputs and outputs to be controlled and observed. As shown in Fig. 3(a), inputs or outputs of the cell that are also inputs or outputs of the module are directly assigned a free variable 'A'. However, there are also l cell inputs and m cell outputs that need to be justified and propagated through the surrounding cells. A transparency composition scheme [20] is employed, identifying a surjective path from k module inputs to the l cell inputs, where $k \geqslant l$, and an



TEST SET	PATH REQUIREMENT
TS1={00, 11}	(A1, A1) : 1 Free Variable
TS2={01, 10}	(A1, A1') :1 Free Variable
TS3={00, 01}	(V, A1) : 1 Free Variable
TS4={10, 11}	(V, A1) : 1 Free Variable
TS5={00, 01, 10}	(A1, A2) : 2 Free Variables

(a)



TEST SET	PATH REQUIREMENT
TS6= {001, 010, 101, 110}	(A1, A2, A2') : 2 Free Variables
TS7= {010, 100, 110, 001}	(A1, A2, A3) : 3 Free Variables

(b)

Fig. 2. Test Requirement severity examples.

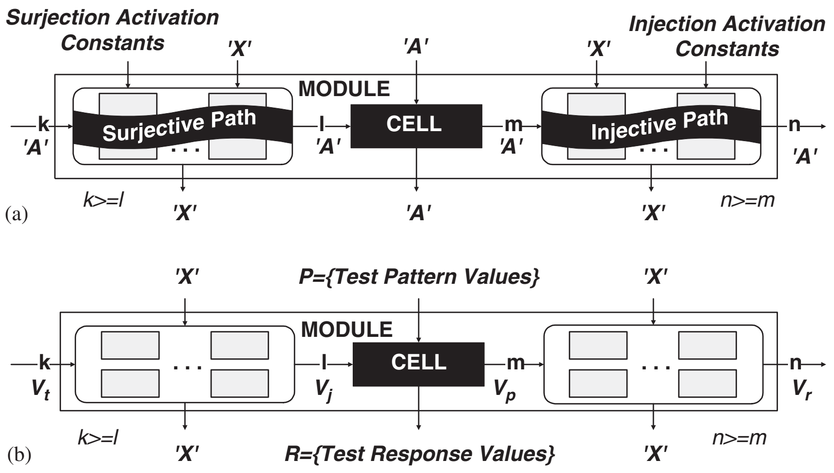


Fig. 3. Proposed vs. exhaustive methodology.

injective path from the m cell outputs to n module outputs, where $n \geqslant m$. The resulting justification requirements for the module inputs are either a constant '0' or '1', or a free variable 'A', while the propagation requirements for the module outputs are free variables 'A'. The remaining inputs and outputs are assigned to 'X'.

The transparency-based scheme is a relaxed test requirement analysis. In Fig. 3(b) for example, the cell inputs that are also module inputs should be assigned to P, the set of required vectors. Similarly, the cell outputs that are also module outputs should be assigned to R, the set of required responses. For the l cell inputs and m cell outputs that are justified and propagated through the surrounding cells, exact analysis is more complicated. Assume that V_j, $V_j \subseteq 2^l$, is the set of values that need to be justified to these l inputs of the cell. Similarly, assume that V_p, $V_p \subseteq 2^m$, is the set of values that needs to be distinguishably propagated from these m outputs of the cell. Essentially, a set V_t, $V_t \subseteq 2^k$, and a set V_r, $V_r \subseteq 2^m$, such that $|V_j| = |V_t|$ and $|V_p| = |V_r|$, are required, with the module implementing a function f from V_t to V_j and a function g from V_p to V_r. Then, V_t and V_r would be the remaining test requirements at the module inputs and outputs. Such a value-based reasoning for identifying the sets V_j, V_t, V_p, and V_r, providing exact test requirements, is overly time-consuming. Therefore, the described transparency-based scheme results in a simpler and faster identification of test requirements.

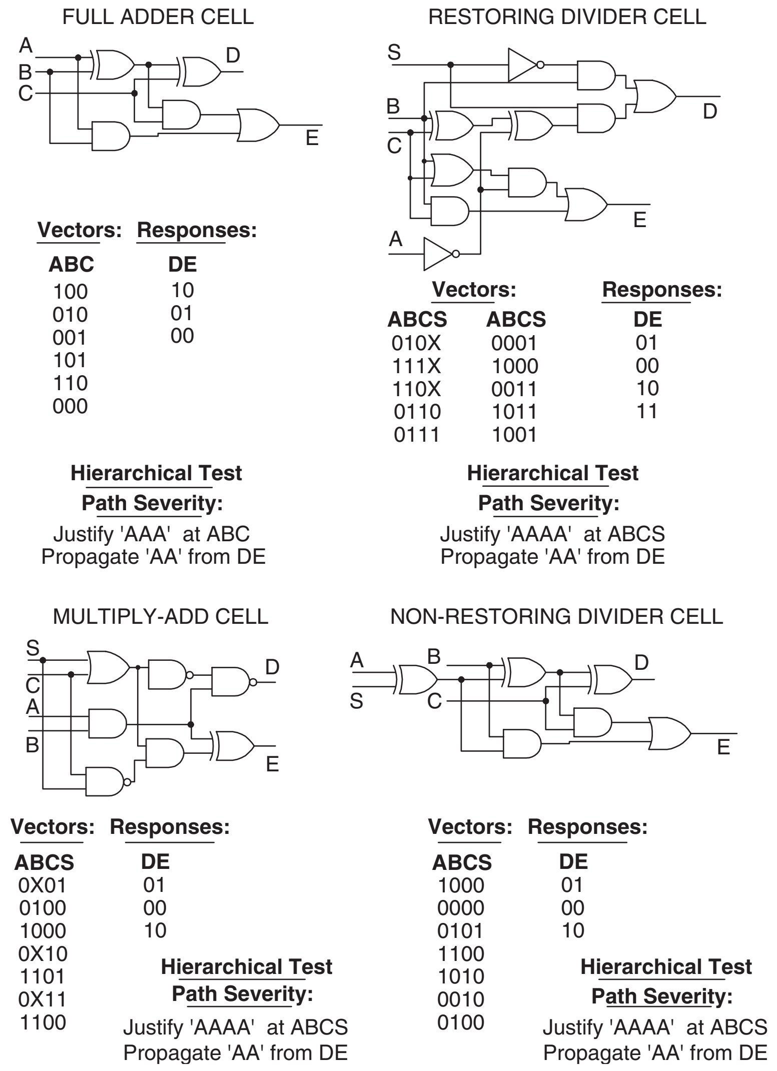
3.3. Cell granularity

Success of the proposed methodology depends on the choice of cell granularity, which is based on several factors. First of all, the selected cell should satisfy the severity threshold condition. Repetitive structures should also be considered, since they result in regular and parametrizable requirements. Finally, the size and the number of cells should be taken into account. An examination of several basic cells reveals that they constitute the appropriate granularity level for hierarchical test requirement identification. Due to the dense connectivity structure within such basic cells, as compared to the sparser inter-cell connectivity, gate-level test requirements satisfy the severity threshold condition of Section 3.1. Fig. 4 shows examples of four cells and the corresponding gate-level tests[1] which satisfy the severity threshold condition. Cells of this granularity are the basic components of arithmetic circuits [22]. Basic cells allow exploitation of regularity and reduction of analysis complexity. With the exception of boundary cells, only prototypical cells are analyzed and test requirements are defined in a parametrized way. Furthermore, regular requirements incur regular DFT, which can be combined across the requirements of several cells and be highly optimized.

3.4. Test requirement granularity adjustment

While test requirement identification at a finer granularity than the basic cell level does not provide hierarchical test path severity reduction, the derived test requirements across several cells may also satisfy the threshold condition. Therefore, granularity should be adjusted accordingly, resulting in a compact set of test requirements that are as symbolic as possible but do not increase hierarchical test path severity. The proposed methodology adjusts granularity to inter-cell connectivity through a structural analysis of the requirements and the paths. If the paths required for accessing and testing a cell fully incorporate additional cells, then the cells are combined into a larger block. For example, consider the connectivity structure shown in Fig. 5. The surjective and injective paths required for testing cell ♯3 fully incorporate cells ♯1, ♯2, ♯4, and ♯5. It is therefore wise to combine test application for all five cells, since no additional severity is imposed on the corresponding hierarchical test paths. Another way of explaining this is that the requirements for testing cells ♯1, ♯2, ♯4, and ♯5 are subsets of the requirements for testing

[1]Tests were generated using ATALANTA [21] with random fill off.



FULL ADDER CELL

Vectors: ABC	Responses: DE
100	10
010	01
001	00
101	
110	
000	

Hierarchical Test Path Severity:
Justify 'AAA' at ABC
Propagate 'AA' from DE

RESTORING DIVIDER CELL

Vectors: ABCS	ABCS	Responses: DE
010X	0001	01
111X	1000	00
110X	0011	10
0110	1011	11
0111	1001	

Hierarchical Test Path Severity:
Justify 'AAAA' at ABCS
Propagate 'AA' from DE

MULTIPLY-ADD CELL

Vectors: ABCS	Responses: DE
0X01	01
0100	00
1000	10
0X10	
1101	
0X11	
1100	

Hierarchical Test Path Severity:
Justify 'AAAA' at ABCS
Propagate 'AA' from DE

NON-RESTORING DIVIDER CELL

Vectors: ABCS	Responses: DE
1000	01
0000	00
0101	10
1100	
1010	
0010	
0100	

Hierarchical Test Path Severity:
Justify 'AAAA' at ABCS
Propagate 'AA' from DE

Fig. 4. Cells satisfying the severity threshold condition.

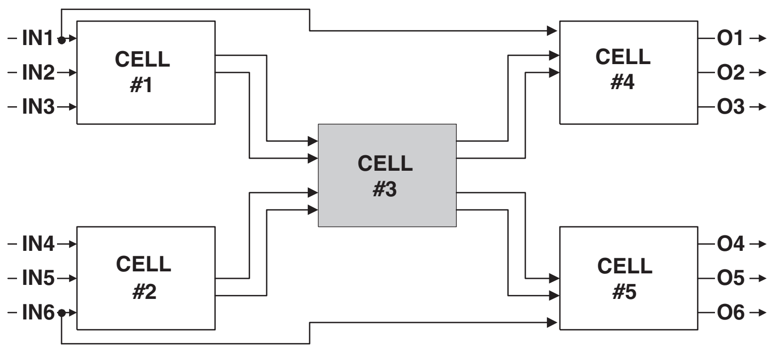
cell ♯3, and can therefore be discarded. Consequently, once the test requirements for each cell are identified at the module boundary, an additional granularity adjustment is made.

4. Examples

We demonstrate the proposed method on several example modules to validate its ability to identify symbolic, yet accurate test requirements, adjusting their granularity to the inter-cell connectivity of the module. The method exploits inherent cell regularity, in order to parametrize and compact the identified test requirements.

The first module, shown in Fig. 6, is a simple 4-bit carry-ripple adder [22], comprising 4 full-adder cells, such as the one shown in Fig. 4. Consider for example the test requirements for *FA*♯3. According to the proposed methodology of Section 3.2, '*A*'s are assigned to the inputs and outputs of the cell that are also inputs and outputs of the module, in this case A_2, B_2, and Z_2. The '*A*' requirement on C_2 is satisfied through a surjective path from A_1, B_1, while the '*A*' requirement on C_3 is satisfied through an injective path to Z_3, activated by any constant value '*V*' on A_3, B_3. The remaining inputs and outputs, C_{in}, A_0, B_0, Z_0, Z_1, and C_{out} are assigned '*X*'s. The identified test requirements are symbolic but also compact, thus close in precision to exact test. In addition, module regularity allows test requirement parametrization; therefore, the analysis is performed only for the prototypical cell and the boundary cases.

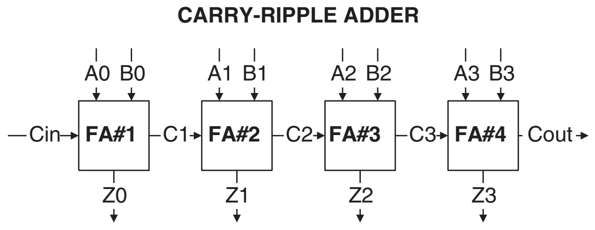
The second module is the 74181 ALU [17], which unlike the adder is neither homogeneous, nor regular. The connectivity and the test requirements for each cell are shown in Fig. 7. Based on the granularity adjustment methodology of Section 3.4, the propagation requirements for the cell pairs (♯1, ♯5), (♯2, ♯6), (♯3, ♯7), and (♯4, ♯8) are merged. Furthermore, establishing a surjective path to the 10 inputs of cell ♯9 requires a hierarchical test path to all 14 inputs of the ALU. Consequently, the justification requirements for cells ♯1 through ♯8 are all subsets of the



EXAMPLE TEST REQUIREMENTS:

	IN1	IN2	IN3	IN4	IN5	IN6	O1	O2	O3	O4	O5	O6
CELL #1:	A	A	A	V	V	X	A	A	X	X	X	X
CELL #2:	V	V	X	A	A	A	X	X	A	A	A	A
CELL #3:	A	A	A	A	A	A	A	A	A	A	A	A
CELL #4:	A	A	A	V	V	X	A	A	X	X	X	X
CELL #5:	V	V	X	A	A	A	X	X	A	A	A	A
Combined Requirement:	A	A	A	A	A	A	A	A	A	A	A	A

Fig. 5. Granularity adjustment example.



Test Justification Requirements:

	Cin	A0	B0	A1	B1	A2	B2	A3	B3
FA#1:	A	A	A	V	V	X	X	X	X
FA#2:	X	A	A	A	A	V	V	X	X
FA#3:	X	X	X	A	A	A	A	V	V
FA#4:	X	X	X	X	X	A	A	A	A

Test Propagation Requirements:

	Z0	Z1	Z2	Z3	Cout
FA#1:	A	A	X	X	X
FA#2:	X	A	A	X	X
FA#3:	X	X	A	A	X
FA#4:	X	X	X	A	A

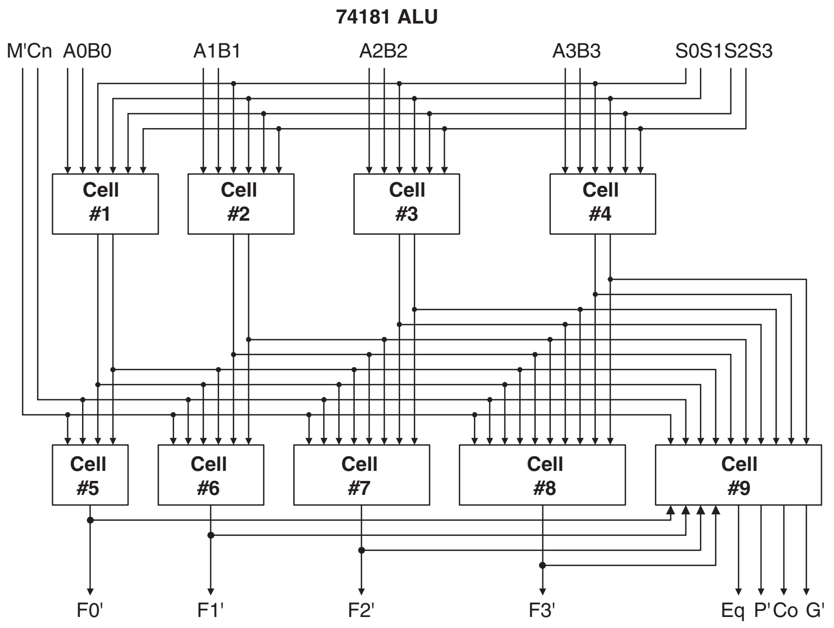
Fig. 6. Test requirements of carry-ripple adder.

justification requirements for cell ♯9 and are discarded. The final set of test requirements for the ALU is thus adjusted to the module connectivity and is shown in boldface.

The third module is a restoring array divider [22] composed of cells such as the one shown in Fig. 4. The circuit and the test requirements are shown in Fig. 8. The inherent module regularity allows parametrization of the test requirements. Consider, for example, the test requirements for cell ♯5. The four inputs of the cells are justified through two surjective paths, one from D_3, Z_4, and Z_5 and another one from D_2 and Z_2. The two outputs of the cell are propagated through injective paths to outputs Q_2, Q_3, and S_4. These inputs and outputs are consequently assigned a test requirement '*A*'. The remaining inputs all require constant values to establish the injective and surjective paths and are, therefore, assigned a test requirement '*V*', while the remaining outputs are assigned a test requirement '*X*'. The granularity is once again adjusted using the methodology of Section 3.4, and the final set is shown in boldface.

5. Severity metrics and experimental results

The following two metrics are introduced to reflect the severity imposed by the test requirements of a module on test path existence and test path identification. The underlying assumption is that the likelihood of path existence and the complexity of path identification decrease, as the generality of the path increases. Path generality increases with the width and with the values to be attained at each bit.



Test Justification Requirements:

	M'	Cn	A0	A1	A2	A3	B0	B1	B2	B3	S0	S1	S2	S3
Cell#1:	V	V	A	X	X	X	A	X	X	X	A	A	A	A
Cell#2:	V	V	V	A	X	X	V	A	X	X	A	A	A	A
Cell#3:	V	V	V	V	A	X	V	V	A	X	A	A	A	A
Cell#4:	V	V	V	V	V	A	V	V	V	A	A	A	A	A
Cell#5:	A	A	A	X	X	X	A	X	X	X	A	V	V	V
Cell#6:	A	A	A	A	X	X	A	A	X	X	A	A	V	V
Cell#7:	A	A	A	A	A	X	A	A	A	X	A	A	A	V
Cell#8:	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Cell#9:	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**

Test Propagation Requirements:

	G'	Co	P'	F3'	F2'	Eq	F1'	F0'
Cell#1:	**X**	**X**	**X**	**X**	**X**	**X**	**A**	**A**
Cell#2:	**X**	**X**	**X**	**X**	**A**	**X**	**A**	**X**
Cell#3:	**X**	**X**	**X**	**A**	**A**	**X**	**X**	**X**
Cell#4:	**X**	**A**	**X**	**A**	**X**	**X**	**X**	**X**
Cell#5:	X	X	X	X	X	X	X	A
Cell#6:	X	X	X	X	X	X	A	X
Cell#7:	X	X	X	X	X	A	X	X
Cell#8:	X	X	X	X	A	X	X	X
Cell#9:	**A**	**A**	**A**	**X**	**X**	**A**	**X**	**X**

Fig. 7. Test requirements of 74181 ALU.

Test Path Existence Severity, reflecting the possibility that hardware will be needed to establish transparency paths due to the generality of test requirements, is defined as

$$TPES(Module) = \sum_{\forall Paths} TPES(Path), \text{ where} \tag{1}$$

$$TPES(Path) = \prod_{\forall Bits} TPES(Bit), \text{ and} \tag{2}$$

$$TPES(Bit) = \left\{ \begin{array}{ll} 1 & \text{if} \quad `X' \\ 2 & \text{if} \quad `V' \\ 4 & \text{if} \quad `A' \end{array} \right\} \tag{3}$$

Test path identification severity, reflecting the possibility that testability hardware will be needed due to the translation complexity of exact test requirements, is defined as

$$TPIS(Module) = \sum_{\forall Paths} TPIS(Path) \tag{4}$$

where

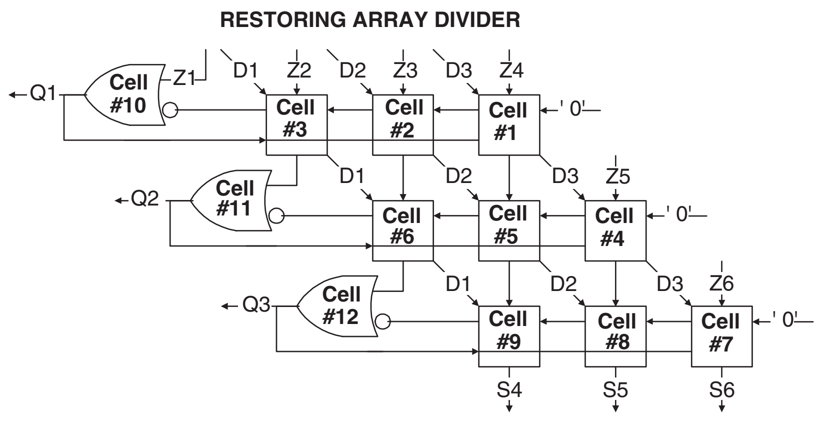
$$TPIS(Path) = \prod_{\forall Bits} TPIS(Bit) \tag{5}$$

and

$$TPIS(Bit) = \left\{ \begin{array}{ll} 1 & \text{if} \quad `X' \\ 2 & \text{if} \quad `A' \\ 4 & \text{if} \quad `V' \end{array} \right\} \tag{6}$$

As an example, Fig. 9 calculates the controllability and observability TPES and TPIS of a 4-bit carry-ripple adder for the test requirements imposed by symbolic paths and by compacted gate-level test. Fig. 10, calculates the metrics for the requirements imposed by non-compacted gate-level test and by the proposed method.

The controllability metrics C-TPES and C-TPIS for the four approaches are summarized in Tables 1 and 2, while the observability metrics O-TPES and O-TPIS are summarized in Tables 3 and 4. Results are also reported in these tables for the restoring divider and the ALU example circuits of the previous section, as well as an 8-bit pipelined



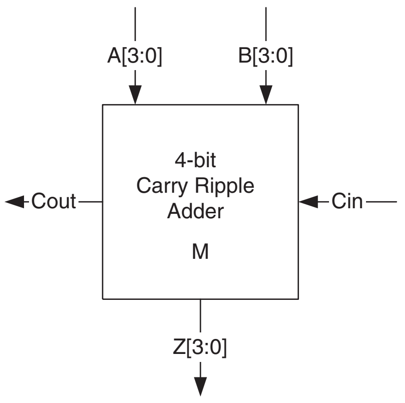
Test Justification Requirements:

	Z1	Z2	Z3	Z4	Z5	Z6	D1	D2	D3
Cell#1:	A	V	V	A	V	V	V	V	A
Cell#2:	A	V	A	A	V	V	V	A	A
Cell#3:	A	A	A	V	V	V	A	A	V
Cell#4:	V	A	V	V	A	V	V	V	A
Cell#5:	V	A	V	A	A	V	V	A	A
Cell#6:	V	A	A	A	V	V	A	A	V
Cell#7:	V	V	A	V	V	A	V	V	A
Cell#8:	V	V	A	V	A	A	V	A	A
Cell#9:	V	V	A	A	A	V	A	A	V
Cell#10:	A	A	X	X	X	X	A	X	X
Cell#11:	A	A	A	V	X	X	A	V	X
Cell#12:	X	A	A	A	V	X	A	V	X

Test Propagation Requirements:

	Q1	Q2	Q3	S4	S5	S6
Cell#1:	A	A	X	A	X	X
Cell#2:	A	A	A	X	X	X
Cell#3:	A	A	X	X	X	X
Cell#4:	X	A	A	X	A	X
Cell#5:	X	A	A	A	X	X
Cell#6:	X	A	A	X	X	X
Cell#7:	X	X	A	X	X	A
Cell#8:	X	X	A	X	A	X
Cell#9:	X	X	A	A	X	X
Cell#10:	A	X	X	X	X	X
Cell#11:	X	A	X	X	X	X
Cell#12:	X	X	A	X	X	X

Fig. 8. Test requirements of array divider.



TEST PATTERNS - TEST RESPONSES

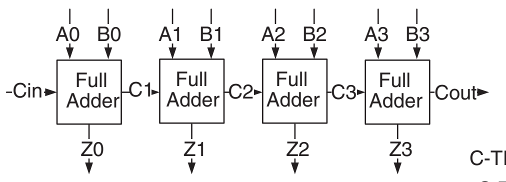
A3A2A1A0B3B2B1B0Cin	*CoutZ3Z2Z1Z0*
0 0 0 0 1 0 1 1 0	0 1 0 1 1
0 1 0 1 0 1 1 0 1	0 1 1 0 0
1 0 0 1 0 0 1 1 0	0 1 1 0 0
1 1 1 0 0 1 1 0 1	1 0 1 0 1
0 1 0 1 0 0 0 1 0	0 0 1 1 0
1 0 1 0 1 1 1 0 0	1 1 0 0 0
0 0 0 1 1 1 0 0 0	0 1 1 0 1
0 1 1 0 1 0 0 1 0	0 1 1 1 1

8 Distinct Vectors - 7 Distinct Responses

Symbolic Paths	**Compacted Test**
1 Justification Path: "AAAAAAAAA"	*8 Justification Paths: "VVVVVVVVV"*
C-TPES (M)=4^9=262144	C-TPES (M)=8*2^9=4096
C-TPIS (M)=2^9=512	C-TPIS (M)=8*4^9=20197152
1 Propagation Path: "AAAAA"	*7 Propagation Paths: "VVVVV"*
O-TPES=4^5=1024	O-TPES=7*2^5=224
O-TPIS=2^5=32	O-TPIS=7*4^5=7168

Fig. 9. Metric calculation example for symbolic paths and compacted test.

multiplier accumulator (MAC) [23]. As demonstrated, the coarseness of the symbolic paths results in very high TPES values, although their generality ensures low TPIS values. On the other hand, the accuracy of exact test ensures low TPES values, yet results in high TPIS values due to the complexity of exact translation. If the test is not compacted the problem is slightly alleviated but the TPIS values are still orders of magnitude higher than the TPES values. The



TEST PATTERNS - TEST RESPONSES
(RANDOM FILL TURNED OFF)

A3A2A1A0B3B2B1B0Cin CoutZ3Z2Z1Z0

```
0 1 1 X 0 0 1 X X    0 1 0 X X
X 0 1 1 X 0 0 1 X    X X 1 0 X
0 1 0 X 0 0 0 X X    0 0 1 X X
0 0 0 X 0 1 0 X X    0 0 1 X X
0 0 1 X 0 0 1 X X    0 0 1 X X
X 0 1 0 X 0 0 X 0    X X 0 1 X
X 0 0 0 X 0 1 X 0    X X 0 1 X
X 0 0 1 X 0 0 1 X    X X 0 1 X
X X 0 1 X X 0 0 1    X X X 1 0
X X 0 1 X X 0 0 0    X X X 0 1
X X 0 0 X X 0 1 0    X X X 0 1
X X 0 0 X X 0 0 1    X X X 0 1
0 1 X X 0 1 X X X    0 1 X X X
1 0 X X 1 0 X X X    1 0 X X X
1 0 X X 0 0 X X X    0 1 X X X
0 0 X X 1 0 X X X    0 1 X X X
1 1 X X 0 1 X X X    1 0 X X X
```

Proposed Methodology

4 Justification Paths:
"XXVAXXVAA"
"XVAAXVAAX"
"VAAXVAAXX"
"AAXXAAXXX"

$\text{C-TPES(M)}=2^2*4^3+2^2*4^4+2^2*4^4+4^4=2560$

$\text{C-TPIS(M)}=4^2*2^3+4^2*2^4+4^2*2^4+4^2=656$

4 Propagation Paths:
"XXXAA"
"XXAAX"
"XAAXX"
"AAXXX"

$\text{O-TPES}=4^2+4^2+4^2+4^2=64$

$\text{O-TPIS}=2^2+2^2+2^2+2^2=16$

Non-Compacted Test

17 Distinct Justification Paths:
(8 have 3 Xs, 4 have 4 Xs, 5 have 5 Xs)

$\text{C-TPES(M)}=8*2^6+4*2^5+5*2^4=848$

$\text{C-TPIS(M)}=8*4^6+4*4^5+5*4^{4}*2^4+4^2=38144$

7 Distinct Propagation Paths:
(3 have 2 Xs, 4 have 3 Xs)

$\text{O-TPES}=3*2^3+4*2^2=40$

$\text{O-TPIS}=3*4^3+4*4^2=256$

Fig. 10. Metric calculation example for non-compacted test and proposed method.

Table 1
Comparison of C-TPES metrics

C-TPES metric	Symbolic paths	Compacted test	Non-compacted test	Proposed method
Adder	262144	4096	848	2560
Divider	262144	9216	7360	49152
ALU	238435456	425984	65280	238435456
MAC	17179869184	4980736	1269728	318644

Table 2
Comparison of C-TPIS metrics

C-TPIS metric	Symbolic paths	Compacted test	Non-compacted test	Proposed method
Adder	512	20197152	38144	656
Divider	512	4718592	3662468	98304
ALU	16384	6979321856	230430714	16384
MAC	131072	652835028992	7356319724	102796

proposed methodology resolves the problem by combining the generality required for fast hierarchical test path construction with the accuracy necessary for ensuring translatability. As a result, the TPES and TPIS values are of the same order of magnitude and close to the minimal values, except for controlling the ALU where the full path is required. Thus, the identified test requirements significantly reduce the overall burden imposed on hierarchical test.

Table 3
Comparison of O-TPES metrics

O-TPES metric	Symbolic paths	Compacted test	Non-compacted test	Proposed method
Adder	1024	224	40	64
Divider	4096	1088	832	36
ALU	65536	5632	4064	32
MAC	262144	11264	8266	784

Table 4
Comparison of O-TPIS metrics

O-TPIS metric	Symbolic paths	Compacted test	Non-compacted test	Proposed method
Adder	32	7168	256	16
Divider	64	69632	47888	272
ALU	256	1441792	1013856	320
MAC	512	5767168	3964236	2840

6. Conclusions

Accurate modular test requirement identification is critical to the cost-effectiveness of hierarchical test, since the severity imposed on the corresponding hierarchical test paths is directly related to the anticipated testability hardware overhead. A thorough understanding of the severity imposed by exact test patterns as compared to symbolic test provides the basis for defining appropriate test requirements. The proposed methodology identifies a set of fine-grained, yet adequate input and output bit clusters to be justified and propagated respectively, through which symbolic test can be applied to each basic cell in the module. Through an efficient cell-based transparency extraction approach, the proposed method adjusts the granularity of the identified test requirements to the module connectivity. Furthermore, the identified test requirements are independent of particular test sets and can be parametrized to exploit inherent repetitive structures and regularity in the design, thus reducing the analysis time and the corresponding storage. Most importantly, the identified test requirements combine the generality required for fast hierarchical test path construction with the accuracy necessary for minimizing the corresponding hierarchical test path severity. Thus, the overhead incurred for constructing adequate hierarchical test paths is reduced, fostering cost-effective hierarchical test.

References

[1] B.T. Murray, J.P. Hayes, Hierarchical test generation using precomputed tests for modules, IEEE Trans. Comput. Aided Des. 9 (6) (1990) 594–603.

[2] P. Vishakantaiah, J.A. Abraham, D.G. Saab, CHEETA: Composition of hierarchical sequential tests using ATKET, in: International Test Conference, 1993, pp. 606–615.

[3] J. Lee, J.H. Patel, Hierarchical test generation under architectural level functional constraints, IEEE Trans. Computer-Aided Des. Integrated Circuits Syst. 15 (9) (1997) 1144–1151.

[4] R.S. Tupuri, A. Krishnamachary, J.A. Abraham, Test generation for gigahertz processors using an automatic functional constraint extractor, in: Design Automation Conference, 1999, pp. 647–652.

[5] Y. Makris, J. Collins, A. Orailoglu, P. Vishakantaiah, TRANSPARENT: A system for RTL testability analysis, DFT guidance and hierarchical test generation, in: Custom Integrated Circuits Conference, 1999, pp. 159–162.

[6] Y. Zorian, E.J. Marinissen, S. Dey, Testing embedded-core based system chips, in: International Test Conference, 1998, pp. 130–143.

[7] E.J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti, Y. Zorian, On IEEE P1500's standard for embedded core test, Journal of Electronic Testing: Theory and Applications, (Aug. 2002) 365–383.

[8] I. Ghosh, A. Ragunathan, N.K. Jha, A design for testability technique for RTL circuits using control/data flow extraction, IEEE Trans. Computer-Aided Des. Integrated Circuits Syst. (8) (1998) 706–723.

[9] Y. Makris, A. Orailoglu, RTL test justification and propagation analysis for modular designs, J. Electron. Testing: Theory Appl. 13 (2) (1998) 105–120.

[10] S. Freeman, Test generation for data-path logic: The F-path method, IEEE J. Solid 23 (2) (1988) 421–427.

[11] M.S. Abadir, M.A. Breuer, A knowledge-based system for designing testable VLSI chips, IEEE Des. Test Comput. 2 (4) (1985) 56–68.

[12] M. Abramovici, M.A. Breuer, A.D. Friedman, Digital Systems Testing and Testable Design, IEEE Press, New York, 1990.

[13] H. Al-Asaad, J.P. Hayes, B.T. Murray, Scalable test generators for high-speed datapath circuits, J. Electron. Testing: Theory Appl. 12 (1/2) (1998) 111–125.

[14] I. Voyiatzis, A. Paschalis, D. Nikolos, C. Halatsis, R-CBIST: An effective RAM-based input vector monitoring concurrent BIST technique, in: International Test Conference, 1998, pp. 918–925.

[15] K.K. Saluja, R. Sharma, C.R. Kime, A concurrent testing technique for digital circuits, IEEE Trans. Computer-Aided Des. Integrated Circuits Syst. 7 (12) (1988) 1250–1260.

[16] D. Gizopoulos, A. Paschalis, Y. Zorian, An effective built-in self-test scheme for parallel multipliers, IEEE Trans. Comput. 48 (9) (1999) 936–950.

[17] E.J. McCluskey, S. Bozorgui-Nesbat, Design for autonomous test, IEEE Trans. Comput. c-30 (11) (1981) 866–874.

[18] T. Sridhar, J.P. Hayes, Design of easily testable bit-sliced systems, IEEE Trans. Comput. c-30 (11) (1981) 842–854.

[19] H. Elhuni, A. Vergis, L. Kinney, C-Testability of two-dimensional iterative logic arrays, IEEE Trans. Computer-Aided Des. Integrated Circuits Syst. 5 (4) (1986) 573–581.

[20] Y. Makris, V. Patel, A. Orailoglu, Efficient transparency extraction and utilization in hierarchical test, in: VLSI Test Symposium, 2001, pp. 246–251.

[21] ATALANTA combinational test generation tool, Available from ⟨http://www.ee.vt.edu/ha/cadtools⟩.

[22] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford University Press, Oxford, 1999.

[23] P. Ashenden, The Designer's Guide to VHDL, Morgan-Kaufmann, Los Altos, CA, 1996.

Yiorgos Makris received his Diploma of Computer Engineering and Informatics from the University of Patras, Greece, and his M.S. and Ph.D. degrees in Computer Engineering from the University of California, San Diego. He is currently an Associate Professor of Electrical Engineering and Computer Science at Yale University. His research interests include test and reliability of digital and analog circuits and systems, testing of asynchronous circuits, and machine-learning applications in computer-aided design of nanodevices.

Alex Orailoglu received his S.B. Degree cum laude from Harvard University in Applied Mathematics and his M.S. and Ph.D. degrees in Computer Science from the University of Illinois, Urbana-Champaign. He is currently a Professor of Computer Science and Engineering at the University of California, San Diego. His research interests include embedded systems and processors, digital and analog test, fault tolerant computing, computer-aided design, and nanoelectronics.

Professor Orailoglu serves in the technical, organizing and/or steering committees of the major VLSI Test, Design Automation, Embedded Systems and Computer Architecture conferences and workshops. He is an associate editor of the IEEE Design and Test Magazine, of the Journal of Electronic Test: Theory and Applications, of the IEE Digital Systems and Design Journal, and of the Journal of Embedded Computing. He has served as the Technical Program Chair of the 1998 High Level Design Validation and Test (HLDVT) Workshop, as the General Chair of HLDVT'99, as the Technical Program Co-Chair of the 2003 CODES + ISSS (ACM/IEEE Hardware Software Codesign Symposium & ACM/IEEE International System Synthesis Symposium), as the General Co-Chair of CODES + ISSS '04, as the Program Co-Chair of the 18th Symposium on Integrated Circuits and Systems Design (SBCCI 2005), as the Program Chair of the IEEE International Workshop on Design and Test of Defect-Tolerant Nanoscale Architectures (NanoArch 2005), as the Program Chair of the Workshop on Application Specific Processors (WASP 2005), and as the Vice Program Co-Chair of the 2004 VLSI Test Symposium and of the 2005 VLSI Test Symposium. He currently serves as the Vice Program Co-Chair of the 2006 VLSI Test Symposium.

Professor Orailoglu is the founding chair of the Workshop on Application Specific Processors (WASP), and has also served as its General and Program Chair in 2002 and 2003. He is also the founding chair of the IEEE International Workshop on Design and Test of Defect-Tolerant Nanoscale Architectures (NanoArch). Dr. Orailoglu currently serves on the Steering Committees of CODES + ISSS, of WASP, of the IEEE Workshop on RTL and High Level Testing, and of HLDVT. Dr. Orailoglu currently serves on more than 20 Program Committees of technical meetings in the areas of VLSI Test, Embedded Systems, Computer Architectures, and Nanoelectronics.

Professor Orailoglu has served as a member of the IEEE Test Technology Technical Council (TTTC) Executive Committee, as the Vice Chair of TTTC, as the Chair of the Test Technology Education Program group, as the Technical Activities Committee Chair and Planning Co-Chair of TTTC. He currently serves as the Communities Chair of the IEEE Computer Society Technical Activities Board. He is the founding chair of the IEEE Computer Society Task Force on Embedded System Codesign and the founding Vice-Chair of the IEEE Computer Society Task Force on Nanoelectronics. Professor Orailoglu has published 200 research articles and is a Golden Core member of the IEEE Computer Society.