Test Requirement Analysis for Low Cost Hierarchical Test Path Construction

Yiorgos Makris
EE Department - Yale University
yiorgos.makris@yale.edu

Alex Orailoglu
CSE Department - U.C. San Diego
alex@cs.ucsd.edu

Abstract

We propose a methodology that examines design modules and identifies appropriate vector justification and response propagation requirements for reducing the cost of hierarchical test path construction. Test requirements are defined as a set of fine-grained input and output bit clusters and pertinent symbolic values. They are independent of actual test sets and are adjusted to the inherent module connectivity and regularity. As a result, 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.

1. Introduction

Hierarchical methods leverage on the ability to target each module and generate highly efficient local test. This benefit, however, comes at the cost of module accessibility from the primary inputs and outputs, which is typically provided by hierarchical test paths [1, 2, 3, 4]. As depicted in Figure 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 module under test is treated as a black box. Implicitly, it is assumed that all possible vectors and responses need to be justified and propagated, although this is almost never the case. Furthermore, transparent accessibility to all modules is rarely inherent in a design. Consequently, DFT hardware is employed in the construction of hierarchical test paths.

In an effort to reduce this DFT cost, two directions have been examined. Along the first direction, several research efforts [1, 2, 5, 6, 7] 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 paid, however, to a third option, namely moderating DFT cost through an informed definition of modular test requirements. This is depicted in Figure 1(b), where the internal connectivity of the MUT is examined and its test requirements are defined as input and output bit clusters. This necessitates several narrow hierarchical test paths instead of a single coarse path, increasing the probability of their inherent existence.

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 by fine-tuning the generality of test requirements and thus the number and the granularity of necessary hierarchical test paths. Furthermore, regular module connectivity is exploited 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.

Related work is discussed in Section 2. The severity imposed by test requirements on hierarchical test path construction is examined in Section 3. The proposed test requirement identification methodology is introduced in Section 4 and the appropriate cell granularity is discussed in Section 5. Adjustment to cell connectivity is examined in Section 6. Examples are given in Section 7 and severity metrics along with results are provided in Section 8.

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 [5], while bijective functions, satisfying both properties, are referred to as *I-Paths* and *T-Paths* [6]. Several variations of surjective, injective, and bijective functions, including *Ambiguity Sets* [1], *Transparency Modes* [2], and *Transparency Properties* [7], have also been used in order to improve efficiency and reduce cost.

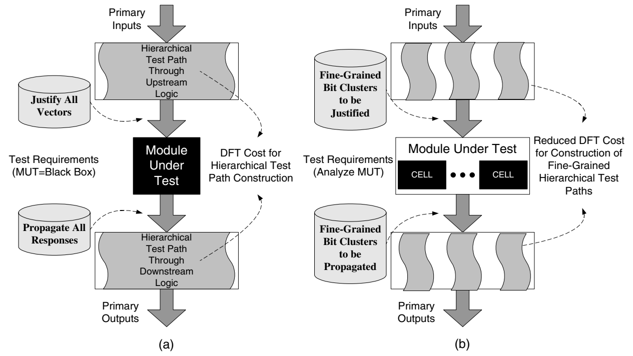
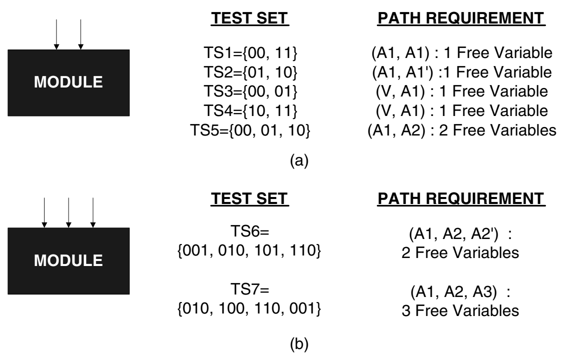

Primary Inputs
Hierarchical Test Path Through Upstream Logic
Justify All Vectors
Test Requirements (MUT=Black Box)
Module Under Test
DFT Cost for Hierarchical Test Path Construction
Propagate All Responses
Hierarchical Test Path Through Downstream Logic
Primary Outputs
(a)
Primary Inputs
Fine-Grained 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)

Figure 1. Granularity of Test Requirements


MODULE
MODULE
(a)
(b)

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

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

Figure 2. Test Requirement Severity Examples

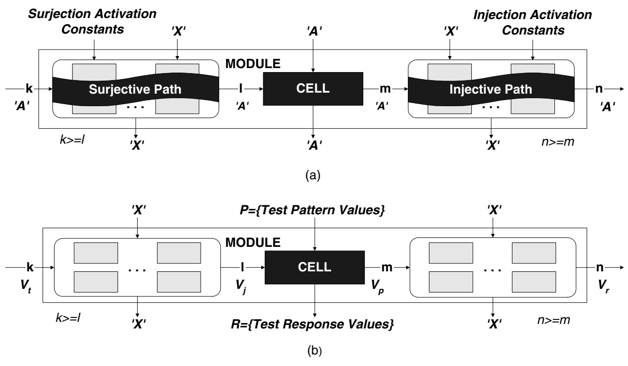


Figure 3. Proposed vs. Exhaustive Methodology

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 [8] commonly used in Built-In Self-Test and Iterative Logic Array Test. The objective of test requirement identification for BIST [9, 10, 11, 12], 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 [13, 14, 15] 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. 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 is not always equal to the vector width. But at a certain set density point, the full width is required; essentially, once 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 Figure 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 3 vectors is reached, however, such as TS_5, a 2-bit hierarchical test path with no but 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 Figure 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.

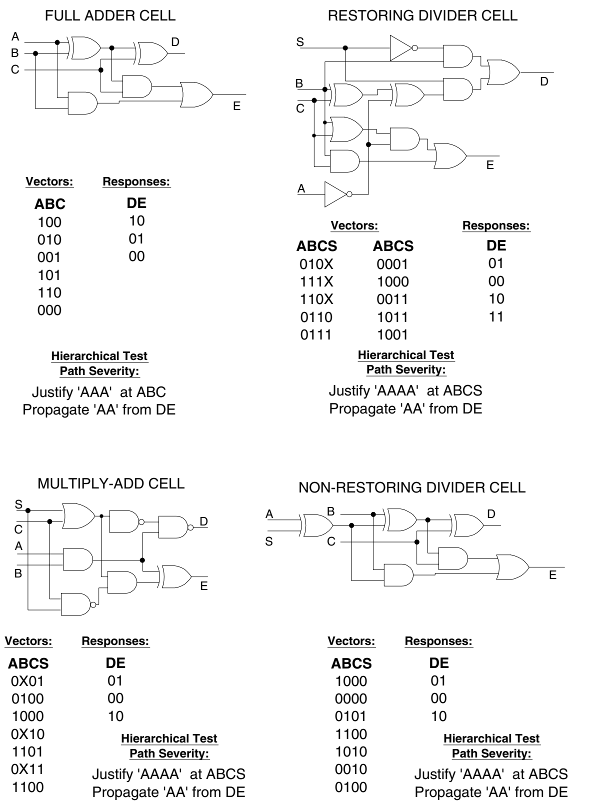
Evidently, there is a threshold for the number and distribution of vectors in a test set, over and above which the severity of the corresponding hierarchical test path is equal to that of the full symbolic path. Generalizing the above observations leads to the following condition:

- **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.

4. Proposed Methodology

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



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	Vectors: 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

Figure 4. Cells Satisfying the Severity Threshold

requirements are translated into module inputs and outputs to be controlled and observed. As shown in Figure 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 [16] is employed, identifying a surjective path from k module inputs to the l cell inputs, where $k \geq l$, and an injective path from the m cell outputs to n module outputs, where $n \geq m$. 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 Figure 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_j to V_t 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 transparency-based scheme described above is employed, resulting in a simpler and faster identification of test requirements

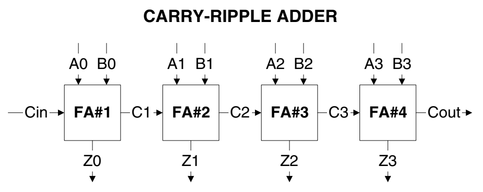
5. Cell Granularity

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. Figure 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 [18]. 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.

6. 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.

[1]Tests were generated using ATALANTA [17] with random fill off.



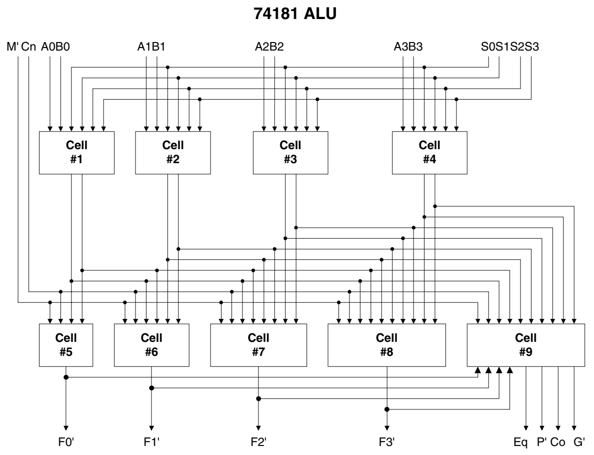
Test Justification Requirements:

	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

Figure 5. Test Requirements of Carry-Ripple Adder



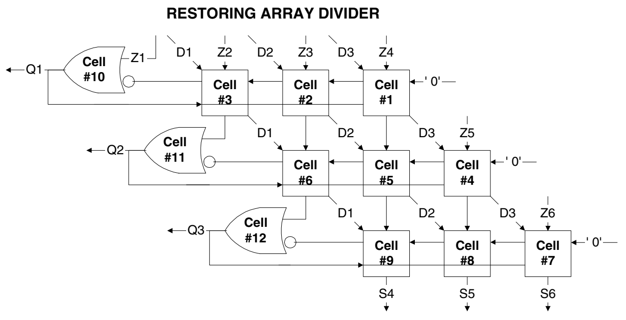
Test Justification Requirements:

	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**

Figure 6. Test Requirements of 74181 ALU



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

Figure 7. Test Requirements of Array Divider

7. Examples

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

The first module, shown in Figure 5, is a simple 4-bit carry-ripple adder [18], comprising 4 full-adder cells, such as the one shown in Figure 4. Consider for example the test requirements for $FA\sharp 3$. According to the proposed methodology of Section 4, $'A'$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 [13], which unlike the adder is neither homogeneous, nor regular. The connectivity and the test requirements for each cell are shown in Figure 6. Based on the granularity adjustment methodology of Section 6, the propagation requirements for the cell pairs $(\sharp 1, \sharp 5)$, $(\sharp 2, \sharp 6)$, $(\sharp 3, \sharp 7)$, and $(\sharp 4, \sharp 8)$ are merged. Furthermore, establishing a surjective path to the 10 inputs of cell $\sharp 9$ requires a hierarchical test path to all 14 inputs of the ALU. Consequently, the justification requirements for cells $\sharp 1$ through $\sharp 8$ are all subsets of the justification requirements for cell $\sharp 9$ 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 [18] composed of cells such as the one shown in Figure 4. The circuit and the test requirements are shown in Figure 7. The inherent module regularity allows parametrization of the test requirements. Consider, for example, the test requirements for cell $\sharp 5$. The four inputs of the cells are justified through a surjective path from D_3, Z_4, and Z_5 and a surjective path 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 6, and the final set is shown in boldface.

8. 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 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

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

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

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

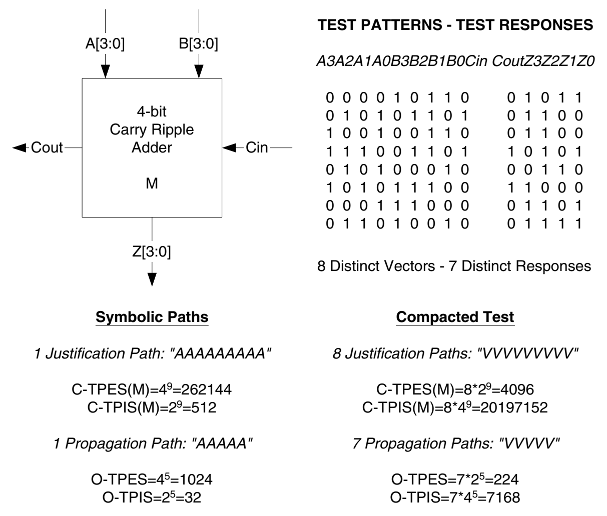
Test Path Identification Severity, reflecting the possibility that testability hardware will be needed due to the translation complexity of exact test requirements, is defined as

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

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

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

As an example, Figure 8 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. Figure 9, further calculates the metrics for the requirements imposed by non-compacted gate-level test and by the proposed methodology. 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 demonstrated, the coarseness of the symbolic paths results in very high TPES values,



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

1 Justification Path: "AAAAAAAAA"

C-TPES(M)=4^9=262144
C-TPIS(M)=2^9=512

1 Propagation Path: "AAAAA"

O-TPES=4^5=1024
O-TPIS=2^5=32

Compacted Test

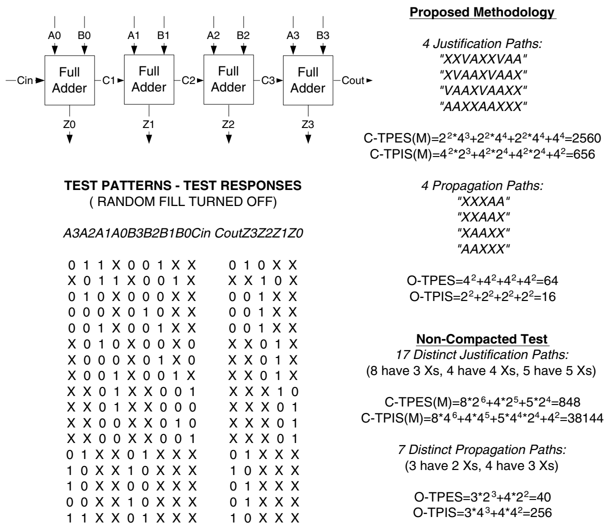
8 Justification Paths: "VVVVVVVVV"

C-TPES(M)=8*2^9=4096
C-TPIS(M)=8*4^9=20197152

7 Propagation Paths: "VVVVV"

O-TPES=7*2^5=224
O-TPIS=7*4^5=7168

Figure 8. Symbolic Paths and Compacted Test



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"

C-TPES(M)=2^2*4^3+2^2*4^4+2^2*4^4+4^4=2560
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"

O-TPES=4^2+4^2+4^2+4^2=64
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)

C-TPES(M)=8*2^6+4*2^5+5*2^4=848
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)

O-TPES=3*2^3+4*2^2=40
O-TPIS=3*4^3+4*4^2=256

Figure 9. Non-Compacted Test and Our Method

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 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[2] and close to the minimal values. Thus, the identified test requirements significantly reduce the overall burden imposed on hierarchical test.

[2]Except for controlling the ALU, where the full path is required.

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

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

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

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

9. 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 competitive hierarchical test.

References

[1] B. T. Murray and J. P. Hayes, "Hierarchical test generation using precomputed tests for modules," *IEEE Transactions on Computer Aided Design*, vol. 9, no. 6, pp. 594–603, 1990.

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

[3] J. Lee and 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.

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

[5] 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.

[6] M. S. Abadir and 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.

[7] Y. Makris and A. Orailoglu, "RTL test justification and propagation analysis for modular designs," *Journal of Electronic Testing: Theory and Applications*, vol. 13, no. 2, pp. 105–120, 1998.

[8] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.

[9] H. Al-Asaad, J. P. Hayes, and B. T. Murray, "Scalable test generators for high-speed datapath circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 12, no. 1/2, pp. 111–125, 1998.

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

[11] K. K. Saluja, R. Sharma, and C. R. Kime, "A concurrent testing technique for digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 12, pp. 1250–1260, 1988.

[12] D. Gizopoulos, A. Paschalis, and Y. Zorian, "An effective built-in self-test scheme for parallel multipliers," *IEEE Transactions on Computers*, vol. 48, no. 9, pp. 936–950, 1999.

[13] E. J. McCluskey and S. Bozorgui-Nesbat, "Design for autonomous test," *IEEE Transactions on Computers*, vol. c-30, no. 11, pp. 866–874, 1981.

[14] T. Sridhar and J. P. Hayes, "Design of easily testable bit-sliced systems," *IEEE Transactions on Computers*, vol. c-30, no. 11, pp. 842–854, 1981.

[15] H. Elhuni, A. Vergis, and L. Kinney, "C-Testability of two-dimensional iterative logic arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 5, no. 4, pp. 573–581, 1986.

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

[17] "ATALANTA combinational test generation tool," Available from `http://www.ee.vt.edu/ha/cadtools`.

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