Reducing Hierarchical Test Path Cost via Modular Test Requirement Analysis

Yiorgos Makris
EE Department - Yale University
New Haven, CT 06520
yiorgos.makris@yale.edu

Alex Orailoglu
CSE Department - U.C. San Diego
La Jolla, CA 92093
alex@cs.ucsd.edu

Abstract

We propose a methodology that examines design modules and identifies appropriate vector justification and response propagation requirements for hierarchical test. Based on a cell-level analysis and transparency composition methodology, test requirements for a module are defined as a set of fine-grained input and output bit clusters and pertinent justification and propagation values. The identified test requirements are independent of the actual test set and are adjusted to the cell-level connectivity and inherent regularity of the module. As a result, they combine the generality required for fast hierarchical test path construction with the accuracy necessary for minimizing the incurred DFT hardware overhead, thus fostering cost-effective hierarchical test. Experimental results verify the ability of the proposed methodology to moderate the cost of hierarchical test path construction through accurate, compact, and highly parametrizable test requirement definition.

1. Summary

Hierarchical methods leverage on the ability to individually target each module in a design and generate highly efficient local test. This benefit, however, comes at the cost of necessitating hierarchical test paths through the upstream and downstream logic, which establish transparent access to the module under test (MUT), as depicted in Figure 1(a). During hierarchical test path construction, however, the module under test is treated as a black box and no information pertaining to its inherent test requirements is utilized. Consequently, excessive DFT hardware is employed in order to construct hierarchical test paths. In an effort to reduce the DFT cost of hierarchical test path construction, two directions have been examined. Along the first direction, several research efforts have been invested in efficiently defining, extracting, and utilizing inherent design transparency. Along the second direction, inherent functionality of the upstream and downstream logic is utilized to constrain local test generation in an effort to render highly translatable test. Not much attention has been paid, however, to a third alternative, namely the possibility of moderating the cost of hierarchical test paths through an examination and informed definition of test requirements for each module. This idea is depicted in Figure 1(b), where the internal cell-level connectivity of the MUT is examined and its test requirements are defined in terms of several input and output bit clusters. This, in turn, necessitates several narrow hierarchical test paths instead of a single coarse path, thus increasing the probability of their inherent existence in the design and reducing the expected DFT cost for their construction. In this work, we examine the impact of test requirement granularity on the severity of hierarchical test path construction and we propose a methodology for reducing the cost of hierarchical test path construction by adjusting the generality of test requirements. A cell-level analysis and a symbolic path composition result in the definition of fine-grained test requirements on sets of input and output bit clusters.

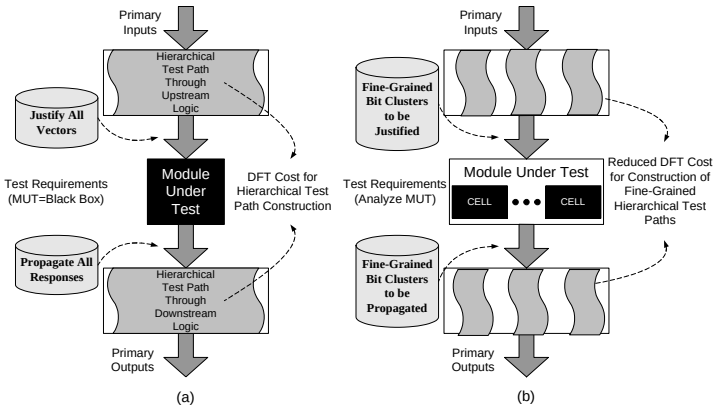


Figure 1. Granularity of Test Requirements

Test Requirement Analysis for Low Cost Hierarchical Test Path Construction

Yiorgos Makris

ELECTRICAL ENGINEERING DEPARTMENT

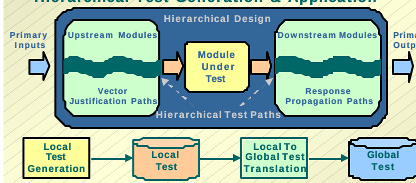
YALE University

Alex Orailoglu

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

Motivation

Hierarchical Test Generation & Application



- Divide: Complexity Reduction – Customization
- Conquer: Need Access to the Boundary of Each Module
- **Limitation:** DFT Cost for Hierarchical Test Path Construction

Research Direction

Problem Definition
- Lack of Identifiable Transparency Incurs DFT Overhead

Current Research Efforts
- Reduce the Cost of Hierarchical Test Paths through:
  - Improved Transparency Extraction Methods
  - Low-Cost DFT for Establishing Transparency

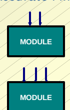
Proposed Method
- Reduce the Cost of Hierarchical Test Paths through:
  - Accurate Test Requirement Identification
  - Low-Cost DFT for Reducing Test Requirements

Idea
- Identify Sufficient Test Requirements Expressed in Many Narrow Paths Instead of One Wide Path

Justification
- Full Transparency Rarely Needed for Testing Each Module
- Narrow Paths Have Higher Probability of Inherent Existence

Hierarchical Test Path Severity

Test Requirements of Modules
- Test Requirement Severity Critical to Path Complexity
- Current Approaches Assume Full Symbolic Path Needed
- Accurate Fine-Grained Paths Important to Reduce DFT



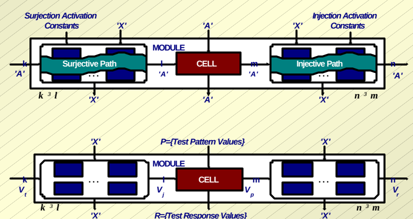
TEST SET	PATH REQUIREMENT
TS1={00, 11}	(A1, A1) : 1 Free Variable
TS2={01, 10}	(A1, A1') :1 Free Variable
TS3={00, 01}	(V, A1) : 1 Free Variable
TS4={10, 11}	(A1, V) : 1 Free Variable
TS5={00, 01, 10}	(A1, A2) : 2 Free Variables

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

Severity Threshold Condition
- *The hierarchical test path construction 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*

Proposed Methodology

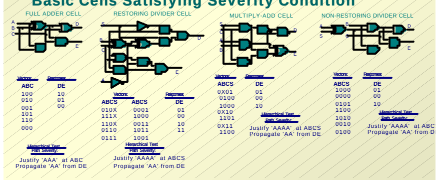
Cell-Based vs. Exhaustive Requirement Analysis



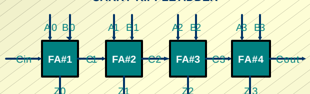
Cell Granularity

Selection Factors
- Satisfaction of Severity Threshold Condition
- Repetitive Structures – Regularity - Homogeneity
- Number and Size of Cells

Basic Cells Satisfying Severity Condition



Test Requirements Example #1



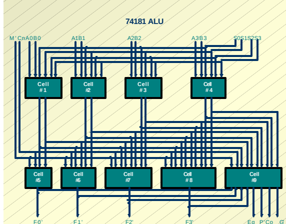
Test Justification Requirements:

	CinA0B0A1B1A2B2A3B3
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:

	Z0Z1Z2Z3Cout
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

Test Requirements Example #2



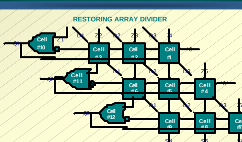
Test Justification Requirements:

	M'CnA0A1A2A3B0B1B2B3S0S1S2S3
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'CoP'F3'F2'EqF1'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

Test Requirements Example #3



Test Justification Requirements:

	Z1Z2Z3Z4Z5Z6D1D2D3
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:

	Q1Q2Q3S4S5S6
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

Metrics and Experimental Results

Severity Metrics

$$TPES(Module) = \sum_{\forall Paths} TPES(Path) \qquad TPIS(Module) = \sum_{\forall Paths} TPIS(Path)$$

$$TPES(Path) = \prod_{\forall Bits} TPES(Bit) \qquad TPIS(Path) = \prod_{\forall Bits} TPIS(Bit)$$

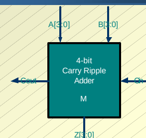
$$TPES(Bit) = \begin{cases} 1 & if \ 'X' \\ 2 & if \ 'V' \\ 4 & if \ 'A' \end{cases} \qquad TPIS(Bit) = \begin{cases} 1 & if \ 'X' \\ 2 & if \ 'A' \\ 4 & if \ 'V' \end{cases}$$

Experimental Results

	Symbolic Paths		Compacted Test		Non-Compacted Test		Proposed Method	
	G-TPES	C-TPIS	C-TPES	C-TPIS	G-TPES	C-TPIS	C-TPES	G-TPIS
Adder	262144	512	4096	20197152	848	38144	2560	656
Divider	262144	512	9216	4718592	7360	3662468	49152	98304
ALU	238435456	16384	425984	6979321856	65280	230430714	238435456	16384

	Symbolic Paths		Compacted Test		Non-Compacted Test		Proposed Method	
	? -TPES	? - TPIS	? - TPES	? -TPIS	? -TPES	? - TPIS	? -TPES	? -TPIS
Adder	1024	32	224	7168	40	256	64	16
Divider	4096	64	1088	69632	832	47888	36	272
ALU	65536	256	5632	1441792	4064	1013856	32	320

Metric Calculation Example



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

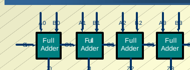
8 Justification Paths: "VVVVVVVVV"

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

Metric Calculation Example (cont'd)



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

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

Proposed Methodology

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

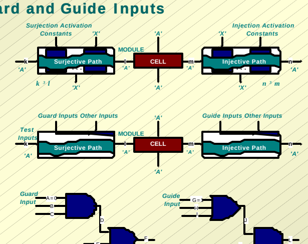
C-TPES(M)=2^2*4^3 +2^2 *4^4+2^2*4^4 +4^4 =2560
C-TPIS(M)=4^2* 2^3+4^2*2^4+4^2* 2^4+4^2=656

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

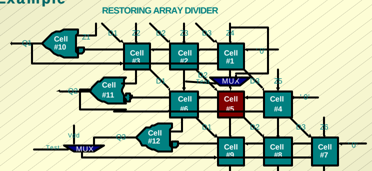
DFT for Test Requirement Reduction

Guard and Guide Inputs



DFT for Test Requirement Reduction

Example



Original Test Justification Requirements:

	Z1Z2Z3Z4Z5Z6D1D2D3
Cell#5:	V A V A A V V A A
Cell#9:	V V A A A V A A V

Modified Test Justification Requirements:

	Z1Z2Z3Z4Z5Z6D1D2D3
Cell#5:	V A V A A X V A V
Cell#9:	V X A A A V X X V

Conclusions

What ?
- Fine-Grained Test Requirement Identification

Why ?
- Reduce the Cost of Hierarchical Test Path Construction

How ?
- Analysis of Severity Imposed on Test Paths
- Efficient Cell-Level Transparency Extraction
- Test Requirement Granularity Identification & Adjustment

Results ?
- Identified Test Requirements Combine:
  - Generality (for Fast Hierarchical Test Path Construction)
  - Accuracy (for Low Hierarchical Test Path Severity & Cost)

Future Work ?
- Extension for Sequential Logic
- DFT For Test Requirement Reduction