Testing Asynchronous Circuits in a Synchronous Environment

F. te Beest, A. Peeters and H. Kerkhoff

MESA+ Research Institute, Testable Design & Test Group, University of Twente
In cooperation with Philips Research Labs
E-mail: F.J.teBeest@el.utwente.nl

Outline

♦ Synchronous test framework
♦ Tangram test
♦ Integrating Tangram test in test framework
♦ Results
♦ Conclusions
Test hierarchy overview (1)

Small logic blocks

Functional test:
- Long development time
- No measure of fault coverage

VLSI logic blocks

Full Scan:
- Large area overhead
- Fast, fault coverage metrics

Test hierarchy overview (2)

Embedded memories, different circuit styles

Macro test:
- Test every circuit style with a special dedicated test

Large reusable designs

Core based test:
- Test is made by core-provider and used by core-user
Non scannable design

Scannable design
Full Scan test

♦ Full scan reduces test complexity from sequential to combinational
♦ Suited for various fault models
♦ Most widely used fault model: Stuck At
♦ Precise fault coverage metrics

Macro test

♦ Disconnect test generation (what) and test executing (how)
♦ Generate independent tests for A and B
♦ Execute then at the interface of C
♦ Can work on multiple-hierarchy levels
♦ Test = Test Patterns (what) + Test Protocol (how)
Terminology

- Pattern
  - A vector with stimulus and response values
- Pattern List
  - The list of all patterns needed for a test of a macro
- Test Protocol
  - The prescription according to which a pattern should be applied
- Test
  - Repeated execution of a test protocol, where every time another pattern from the pattern list is filled in

Scan test protocol

Describes in detail how to apply stimuli and capture responses

Graphical representation: Scan-test timing diagram

Relative Time: -3 -2 -1 0 1 2 3
Tasks in Macro test

♦ Test pattern generation
  – Using various tools depending on type of block and test wanted
♦ Test pattern expansion
  – Transformation of protocols to a higher hierarchical level
♦ Test pattern scheduling
  – Exploit parallelism in design
♦ Test pattern assembly
  – Generate a final top level test, that can be simulated and executed on a tester

Core based test

♦ Reuse of large modules
♦ Intra-company and inter-company use
♦ Intellectual Property Rights (IPR) protection
♦ Reduced time to market
♦ Expertise import
♦ Core internal test often by Macro test
Core test

- Test pattern expansion is a NP-hard problem
- Use special access logic to expand test in core based design

This work:

- Develop a test for asynchronous circuits that is compatible with the Macro test method
- Use as much existing tools as possible
- First focus is the possibility of obtaining a high fault coverage (possible at high cost)
- Subsequently apply various optimizations
Tangram

Tangram program

Handshake circuit

Library

Handshake circuit feedback

Verilog netlist

Ambit (area)

Pearl (timing)

Test library

Scannable Netlist

Tangram circuit structure

Control

Data path

Logic

Logic
Modifications

To reuse synchronous test tools, the circuit (in testmode) has to behave synchronous

Therefore we need to:
♦ Add a global clock
♦ Add scan functionality to every state holding element (C-elements, Latches and Flip-Flops)

Scan C-elements
Scan C-elements
Split circuit

The split between control and data path is needed because of the interference caused by the latch-controller.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Async</th>
<th>Scan</th>
<th>Control normal</th>
<th>Data normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clk 1</td>
<td>1</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>Clk 2</td>
<td>0</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>Tm</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Se</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Full scan with Tangram

♦ Split circuit in Control and Data path
♦ Generate test patterns with standard test pattern generation
♦ Expand tests with Macro test tools
  – This can be done together with the expansion of the test for other blocks, like
    • Various memory styles
    • Synchronous logic
Required tools

♦ New tool to modify Tangram net list, so that it includes scan elements
♦ Reuse tools for:
  – Test pattern generation
  – Test protocol expansion and scheduling
  – Test assembly
  – Simulation

Test flow

![Test flow diagram]

- **TgScan**
- **ATPG**
- **Expansion**
- **Assembly**
- **Verilog / Verifault**
- **Test vectors**
Results

♦ DCC error decoder (DDD) example:
  – Stuck at fault coverage: 99.9%
    • Not including: delay elements and mutual exclusion elements
  – Area overhead

<table>
<thead>
<tr>
<th>DDD</th>
<th>Original</th>
<th>Scannable</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latch</td>
<td>4665 eqv</td>
<td>10141 eqv</td>
<td>117 %</td>
</tr>
<tr>
<td>Flip flop</td>
<td>5066 eqv</td>
<td>9346 eqv</td>
<td>84 %</td>
</tr>
</tbody>
</table>

Results

♦ Additional benefit:

The test pattern generation tool is able to find remaining redundancies in the circuit. This can be used to improve the peephole optimisations in the Tangram compiler.
Conclusion & Future work

♦ Tangram circuit can be full-scan tested with a high fault coverage
♦ High area overhead
  – Large implementation of Scan C-element
    • Partial scan
    • Dedicated scan cells
  – Use of dummy latches
    • L1 L2 scan configurations
    • Dedicated scan cells

Conclusion & Future work

♦ Interface (ports) and Synchroniser test:

  – Octavian Petre, MESA+ Research Institute
    Testable Design & Test Group