VLSI Design Verification and Testing

Functional Testing

Mohammad Tehranipoor
Electrical and Computer Engineering
University of Connecticut

Overview

- Motivation and introduction
- Structure independent approach
- Structure dependant approach
- Organization/architecture dependant approach
  - Microprocessor testing
  - Memory testing
- Summary

Motivation and Introduction

- Ref: Abramovici, et. Al – Reference book Section 18.2 of the text (for understanding the problem)
- Motivation
  - Structural information can facilitate testing – we show this for combinational and sequential circuits
  - Organization/Architecture information can make testing of microprocessors and memories practical
  - Develop fault models when we are to use this kind of information

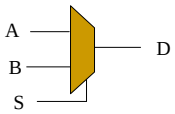
Structure independent approach

- Combinational circuit testing
  - Exhaustive testing of circuits that do not have very large number of inputs
  - Note: if a fault can cause the circuit to behave like a circuit with states (i.e. causes increase in the number of states – combinational circuit becomes sequential in nature) then exhaustive testing may not fully test the circuit. Example of such a fault is *stuck-open* fault in a CMOS implementation

Structure independent approach

- Combinational circuit testing (contd.)
  - A non-exhaustive functional test must be carefully designed otherwise it may not achieve the desired objective. Consider a 2-to-1 mux. A non-exhaustive functional test is given below:

S	A	B	D
0	0	x	0
0	1	x	1
1	x	0	0
1	x	1	1



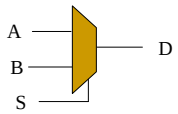
Exhaustive = 8 patterns

Non-exhaustive = 6 patterns

Structure independent approach

- Combinational circuit testing (contd.)
  - A fully specified test can potentially be (the one that repeats values in the test) as follows:

S	A	B	D
0	0	0	0
0	1	1	1
1	0	0	0
1	1	1	1



  - This test cannot test stuck-at fault on the control line S

Structure independent approach

- Sequential circuit testing
  - An example of this method and its limitations has been discussed in detail in the *checking experiment approach* to testing

7

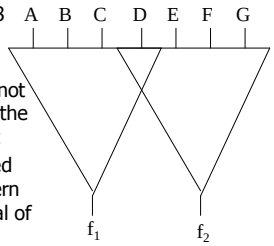
Structure dependent approach

- Combinational circuit testing
  - Structure can potentially provide the information about dependence of outputs on inputs. This leads to two methods of structure dependent functional testing
    - Pseudo-exhaustive testing
    - Sensitized partition testing

8

Structure dependent approach

- Pseudo-exhaustive testing
  - The circuit will require 128 patterns for exhaustive testing
  - Function description may not tell us the dependence of the output on partial input set
  - Both f_1 and f_2 can be tested exhaustively with 16 pattern each, thus requiring a total of 32 patterns



9

Structure dependent approach

- Sensitized partition testing
  - Testing an ALU – control signals are determined and applied in such a way that each smaller part (say 4-bit ALU) is exhaustively tested
  - An example from the book by Abramovici et. al.

10

Structure dependent approach

- Sequential circuit testing
  - Testing iterative logic arrays
  - Machine partitioning approach to testing
    - Substantial literature in this area but has not been applied to real application of today
  - An example – testing of Shift Register

    A simple test 0 1 1 0 0 x x x x x … can test the shift register completely. This test is often used in testing "scan chains"; to be discussed under DFT and BIST methods

11

Organization/architecture dependent approach

- In many cases architecture and/or organization information is available and such information can be used to facilitate testing. We will show two applications of this approach.

12

Microprocessor testing

- References
  - Reference book – Abramovici et. Al.
  - Thatte and Abraham, "Test generation of microprocessors", IEEE Transactions on Computers, June 1980, pp. 429-441

Microprocessor testing

- Basic concept
  - Need to develop test programs that can be executed on the processor
  - Need an open loop strategy to force instructions in the order we wish to execute – e.g. after a jump instruction we may wish to execute an instruction from an address different from the address provided by jump instruction
  - Develop a model (or models) for faults in different organizational sub-units of the microprocessor

Microprocessor testing

- What do we know?
  - Different sub-units
    - Register file, bus, ALU, memory (cache), …
  - How instructions are executed
    - How data moves from one sub-unit to other sub-units
    - Data movement from and to the external world
  - Sequencing and timing
    - Number of clocks, atomic and semi-atomic actions, e.g. PUSH – causes increment PC, send SP as address, send register as data, increment SP, etc.

Microprocessor testing

- Method
  - Test each instruction
  - Test each sub-unit such as ALU
  - Test busses
  - Test register file and decoders
  - Test sequencing of instructions
- Key Concept
  - Start small – test components and instructions that are small and easy to test and then use the tested parts to test other parts

Microprocessor testing

- Model development
  - Determine which instructions are "easy" to execute – such as those that use fewest resources, fewest cycles – easy to control and observe (graph model)
  - Use such instructions to read and write register file to test register file(s) and address decoding logic
  - Test busses by moving different types of data on busses
  - Test ALU by executing ALU related instructions such as ADD, SUB, …
  - …

Microprocessor testing

- Fault model development
  - **Busses**: stuck-at and bridging faults
  - **ALU**: stuck-at (assume that the structural information is available)
  - **Register file**: stuck-at, arbitrary decoder failure – this will use similar fault model and tests as used for testing memories
  - **Instruction decoder**:
    - No instruction is executed (I_j/ϕ)
    - Different instruction is executed (I_j/I_k)
    - An additional instruction is also executed (I_j/I_j+I_k)

Microprocessor testing

- Algorithm development
  - Develop simple sub-programs for each sub-unit testing
  - Put them together
  - Testing "jumps" and "call" will require intervention of tester – open loop strategy of testing microprocessor

19

Microprocessor testing

- Results (case study on an 8-bit HP processors)
  - Program size 1K – FC about 90%
  - Additional complexities introduced in the test program (8K program) raised the coverage by 6%
  - Other faults were associated with the power-up logic, initialization, interrupts, … (hard to test by functional tests)
- Limitations
  - Lack of good and practical model of modern microprocessors
  - **Automating** the program generation difficult and impractical
  - Structural methods with the use of DFT provide better coverage with fewer test vectors

20

Memory testing

- Basic reasoning
  - Logic design methods may not be applicable
    - Special design methods
    - Not designed using logic gates
  - Cutting edge technology
    - High density
    - Novel and non-traditional design methods and layout
  - Can be stand alone or embedded
- Need to develop
  - Fault model
  - Test algorithms

21

Summary

- Described structure independent and structure dependent methods of testing logic
- Microprocessor testing using organization information – model, fault model, test algorithms, results and limitations
- Memory testing – it needs
  - Model, fault model and test algorithms to be discussed next

22