FPGA Design
EECE 277

Verification and Testing

Dr. William H. Robinson
March 18, 2005

http://eecs.vanderbilt.edu/courses/eece277/

Topics
“"I'm not here to talk about the past.””

– Mark McGwire

at the Congressional Hearing on Steroids

• Administrative stuff
  – Read Chapter 11 in your textbook
  – Laboratory Assignment #3 (due Monday, March 28)
  – Initial guidelines for final project

• Fault models
• Test vector generation
• Built-In Self-Test (BIST)

Laboratory Assignment #3

• Quartus II licenses
  – Renew your license if it expires

• Flex 10K70
  – Contains the equivalent of 70,000 gates

• There is no spoon...
  – Nor is there a space between clock and 'EVENT

• Timing analysis tool
  – Can use the longest delay path to calculate

• Area/speed optimization
  – Device technology limits speed

Laboratory Assignment #3

• State machine
  – Sample code on the Rapid Prototyping CD

• ALU
  – Support for shift operators not in std_logic_1164

• Compiling the Simple μP
  – Change address control in memory module to “registered”

• In-class lab session
  – Wednesday, March 23, 9am – 11am
Final Project

General requirements

- Consists of designing and implementing a microelectronic system application that can be demonstrated using the Altera UP-2 prototyping board
- Do not modify your UP2 prototyping board
- Application should involve interfacing with inputs (push-buttons, toggle switches, connectors) and outputs (LEDs, 7-segment displays, connectors)
- It should include some arithmetic computation (datapath) and a state machine (control) as well as a built-in self-test module

Ideas for Final Project

http://microsys6.engr.utk.edu/ece/bouldin_courses/551/candidates.html

- 01. Differential Scoreboard
- 02. Elevator Controller
- 03. Encryption/Compression
- 04. Decompression/Decryption
- 05. Hamming Code Transceiver
- 06. Digital Frequency Meter
- 07. Blackjack Game
- 08. Music Generator with 4-Part Harmony
- 09. Race Track Timer
- 10. Tic-Tac-Toe Game
- 11. String Comparator
- 12. Simon Game
- 13. Other (submit proposal for approval)
### VLSI Realization Process

1. **Customer’s need**
2. **Determine requirements**
3. **Write specifications**
4. **Design synthesis and verification**
5. **Test development**
6. **Fabrication**
7. **Manufacturing test**
8. **Chips to customer**

### Definitions

- **Design synthesis**
  - Given an I/O function, develop a procedure to manufacture a device using known materials and processes

- **Verification**
  - Predictive analysis to ensure that the synthesized design, when manufactured, will perform the given I/O function

- **Test**
  - A manufacturing step that ensures that the physical device, manufactured from the synthesized design, has no manufacturing defect

### Verification vs. Test

**Verification**
- Verifies correctness of design
- Performed by simulation, hardware emulation, or formal methods
- Performed once prior to manufacturing
- Responsible for quality of design

**Test**
- Verifies correctness of manufactured hardware
- Two-part process:
  - 1. Test generation: software process executed once during design
  - 2. Test application: electrical tests applied to hardware
- Test application performed on every manufactured device
- Responsible for quality of devices

### Testing as Filter Process

- **Good chips**
  - Prob(good) = y
  - Prob(pass test) = high
  - Mostly good chips
- **Defective chips**
  - Prob(bad) = 1 - y
  - Prob(fail test) = low
  - Mostly bad chips
Real Tests

• Based on analyzable fault models, which may not map on real defects

• Incomplete coverage of modeled faults due to high complexity

• Some good chips are rejected
  – The fraction (or percentage) of such chips is called the yield loss

• Some bad chips pass tests
  – The fraction (or percentage) of bad chips among all passing chips is called the defect level

The Concept of a Fault

• Testing centers around detection of faults in a circuit

• The digital world is made up of interconnected gates
  – Thus only two things can fail, gates and their interconnections

• A faulty gate fails to perform its function correctly

• A faulty interconnection produces
  – “stuck at 1,” “stuck at 0,” or permanently tri-stated input

• Problem: can you set up experiments that produce one result if the gate/interconnection is good and another if it is bad?
  – This leads to two closely related concepts

Observability and Controllability

• Observability – the ability to observe a gate output directly at external pins

• Controllability – the ability to apply any and all desired inputs to a gate via external pins

• To test a gate completely, all combinations of inputs must be applied and the output corresponding to each input must be observed

Example: Controllability

• How do you control the inputs to gate 2?
  – Set input A to ‘0’ (Sensitize gate 1 to changes on B)
  – Apply four possible input combinations to inputs B and C

• How do you control the inputs to gate 3?
  – Set input A or B to ‘1’ (Sensitize gate 2 to changes on C)
  – Apply all four possible input combinations to C and D
Example: Observability

- **How do you observe the output of gate 1?**
  - Set input $C$ to '1' and input $D$ to '0' (Set gates 2 and 3 so that they pass output of gate 1)

- **How do you observe the output of gate 2?**
  - Set input $D$ to '0' (Set gate 3 so that it passes output of gate 2)

Extending the Concept

- A set of inputs and outputs designed to test a specific gate is called a test vector

- A collection of inputs designed to test a specific gate or area completely is called a test vector suite

- Once a test vector suite is developed for a block, you need to develop a way to get those inputs to the block and directly observe the block’s outputs

Some Problems

- #1: Four additional test vectors are required for every additional 2-input gate
  - The number of test vectors increases as the number of gates increases

- #2: The number of gates is increasing much faster than the number of inputs/outputs
  - This is making both controllability and observability more difficult

- #3: Sequential circuits can be notoriously difficult to test
  - For example, a 16-bit counter has to cycled through its entire count for complete testing.

- Some strategies have emerged to deal with these problems

Design for Testability (DFT)

- DFT refers to hardware design styles or added hardware that reduces test generation complexity

- **Motivation:** Test generation complexity increases exponentially with the size of the circuit

- **Example:** Test hardware applies tests to blocks A and B and to internal bus; avoids test generation for combined A and B blocks
Fault Detection in a Simple Circuit

(a) Circuit

(b) Faults detected by the various input valuations

<table>
<thead>
<tr>
<th>Test Value</th>
<th>a/0</th>
<th>a/1</th>
<th>b/0</th>
<th>b/1</th>
<th>c/0</th>
<th>c/1</th>
<th>d/0</th>
<th>d/1</th>
<th>f/0</th>
<th>f/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Automated Test Vector Generation

- Software exists to analyze a digital design and perform the following functions:
  - Answer the question “What percentage of gates have been completely tested by the test vectors applied to the device?”
    This is known as **Fault Coverage**
  - Generate test vectors to cover the remaining gates

- **Fault Coverage curves look like:**

Issues in Test Vector Generation

- **Issue #1: Diminishing Returns**
  - Vectors start covering smaller and smaller fractions of the device

- **Issue #2: Some tests may be nearly impossible or truly impossible (non-testable faults)**

- Without extra hardware added explicitly to help with testing, fault coverage is typically in the 90 to 95% range

Built-In Self-Test (BIST)

- One way to add reliability to a component or part of a chip is with Built-In Self Test (BIST)

- **An example is the Motorola 68030 MCU**
  - Motorola has added a small set of instructions which they certify tests a significant fraction of the 68030’s gates
  - Can be incorporated into the low level startup process to detect and processor faults.
Testing Arrangement

- Randomly chosen test vectors give good results
  - Fault coverage depends on number of tests performed
- Can incorporate the self-testing capability into the circuit

BIST Approach

- Add circuity so that on powerup a small ROM or pseudo-random sequence generator applies a series of inputs
- The outputs are tested for a correct answer, usually by accumulating the outputs into a checksum value (either by XOR or Cyclic Redundancy Check - CRC)
- The pseudo-random pattern generation and checksum accumulation does not need many gates
- The probability of passing self-test with an undiagnosed fault can be made small (depends on number of patterns).

Linear Feedback Shift Register (LFSR)

- A LFSR can be used as a Pseudo-Random Binary Sequence Generator (PRBSG)
- For an N-bit shift register, the maximum length sequence that can be generated is $2^N-1$ (sequence will repeat after this number of shifts)

Single-Input Compressor Circuit (SIC)

- The signal $p$ is the output of circuit under test
- Pattern generated by SIC is called the signature of the tested circuit for the given test sequence
Multiple-Input Compressor Circuit (MIC)

- Used for circuits with multiple outputs
- Pattern generated by MIC is called the signature of the tested circuit for the given test sequence

Summary

- The purpose of testing is to uncover faults in a system (chip or board)
- Faults are found by applying test vectors and observing results
- Test vectors can be generated manually or automatically