Built-In Self-Test for Programmable I/O Buffers in FPGAs and SoCs

Sudheer Vemula and Charles Stroud
Electrical and Computer Engineering
Auburn University

presented at 2006 IEEE Southeastern Symp. On System Theory
Outline of Presentation

- Motivation and Background
  - Overview of programmable I/O buffers
- Built-In Self-Test Architecture
  - Results Retrieval and Diagnosis
- Embedded Processor Based BIST
- Experimental Results
  - Atmel AT94K series SoC
  - Xilinx Virtex-4 series FPGAs
- Summary
Programmable I/O Buffers

- Programmable I/O buffer can be:
  - Input, Output, Bi-directional
- Package connections
  - Bonded or unbonded
- Different types (some FPGAs)
  - Primary, Secondary, Clock
- Programmable resources
  - Logic
    - Multiplexers, flip-flops, latches
      - Virtex-4: 32 MUXs & 10 FFs
  - Active levels
    - Enables, set/reset, clocks
  - Pull-up, pull-down, keeper
  - Drive capabilities
    - Delays, slew rate
    - I/O voltage standards (Virtex-4: 69)
  - Routing resources to/from FPGA core

Boundary Scan Access

Tri-state Control

Output Data

Input Data

to/from internal programmable routing resources

PAD

D = primary I/O buffer

D = secondary I/O buffer

CLB

3/22/2006  VLSI Design & Test Seminar Series
Motivation and Background

- Need to test programmable I/O buffers in FPGAs to insure proper system operation
  - Boundary Scan lacks access to all resources in programmable I/O buffers
- Configurable SoCs often incorporate programmable I/O buffers for FPGA cores
- FPGA synthesis tools often use resources in unbonded I/O buffers for system function
- Prior work in Built-In Self-Test for FPGAs
  - BIST for internal logic and routing resources
  - No BIST for programmable I/O buffers
## Increase in Resources in I/O Buffers

<table>
<thead>
<tr>
<th>FPGA/Soc</th>
<th>No. of Registers per I/O Buffer</th>
<th>Year</th>
<th>Max. No. of I/O Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMEL AT40(10)K</td>
<td>0</td>
<td>1999</td>
<td>384 (AT40K40)</td>
</tr>
<tr>
<td>ATMEL AT94K</td>
<td>2</td>
<td>2002</td>
<td>256 (AT94S40)</td>
</tr>
<tr>
<td>XILINX VIRTEX</td>
<td>3</td>
<td>2002</td>
<td>512 (XCV1000)</td>
</tr>
<tr>
<td>XILINX VIRTEX E</td>
<td>3</td>
<td>2002</td>
<td>804 (XCV3200E)</td>
</tr>
<tr>
<td>XILINX VIRTEX II PRO</td>
<td>6</td>
<td>2003</td>
<td>1164 (XC2VP70/100/XC2VPX70)</td>
</tr>
<tr>
<td>XILINX SPARTAN3</td>
<td>6</td>
<td>2005</td>
<td>784 (XC3S5000)</td>
</tr>
<tr>
<td>XILINX VIRTEX 4</td>
<td>10</td>
<td>2005</td>
<td>960 (XC4VLX200)</td>
</tr>
</tbody>
</table>
I/O Buffer BIST Architecture

- Configure bi-directional buffer under test (BUT)
- Output response analyzer (ORA) comparison-based
  - Latches mismatches due to faults in BUT
  - Output of BUT compared by 2 ORAs with 2 other BUTs
- Test pattern generator (TPG) = counter or LFSR
  - Implemented in FPGA Configurable Logic Blocks (CLBs)
  - Multiple TPGs prevent CLBs faults from masking IOB faults

Diagram:

- Green circles = TPG
- Cyan circle = ORA
I/O Buffer BIST Architecture

- Test patterns applied to all possible inputs to I/O buffers to test routing resources
- Multiple BIST configurations needed to completely test I/O buffer in all modes of operation
ORA Designs

- Comparator-based design
  - feedback & flip-flop latch any mismatch due to faults in Buffers Under Test (BUTs)

- ORA results retrieval
  - Integrated ORA and scan chain
    - More logic for scan chain
    - Interface to Boundary Scan
  - Configuration memory readback
    - No added logic for scan chain
    - Partial configuration memory readback
    - Readback from embedded processor
      - Diagnosis performed by embedded processor
Circular Comparison Diagnosis

**Step 1:** Record ORA results

**Step 2:** Mark all Buffers Under Test (BUTs) associated with two or more consecutive ORAs with 0s (0=fault-free)

**Step 3:** Recursively mark BUTs with 1 (1=faulty) for every consecutive 0 and 1 followed by empty cell

**Step 4:** Inconsistencies mean fault in BUT-to-ORA routing resources or in ORAs if they have not been tested and known to be fault-free

**Step 5:** Unique diagnosis if all BUTs marked faulty or fault-free

Note: buffers B3 and B4 have equivalent faults

<table>
<thead>
<tr>
<th>O91</th>
<th>B1</th>
<th>O12</th>
<th>B2</th>
<th>O23</th>
<th>B3</th>
<th>O34</th>
<th>B4</th>
<th>O45</th>
<th>B5</th>
<th>O56</th>
<th>B6</th>
<th>O67</th>
<th>B7</th>
<th>O78</th>
<th>B8</th>
<th>O89</th>
<th>B9</th>
<th>O91</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Embedded Processor Based BIST

- Given access to FPGA configuration memory, an embedded processor can:
  - Reconfigure I/O buffers in various modes of operation
  - Execute BIST sequence
  - Retrieve BIST results from ORAs
  - Perform diagnosis based on BIST results

- Processor can be hard or soft core

- Significant improvements in:
  - Memory for storing BIST configurations
  - Test time
Atmel AT94K Implementation

- Three sets of BIST configurations
  - **primary I/O buffers**
    - 13 configurations total
  - **secondary I/O buffers**
    - 10 configurations total
  - **global reset in primary & secondary**
    - $\approx 2 \times 3N$ configurations total
    - where $N = \#$CLBs in one dimension
      - N=24 for AT94K10, N=48 for AT94K40 (280 I/O buffers)

- Subsequent BIST configurations via dynamic partial reconfiguration from AVR processor core
  - Reduces test time compared to download for 303 BIST configs
  - Compared to 68 configs for FPGA core
Fault Detection in Atmel AT94K

- Primary I/O buffer
- Secondary I/O buffer

Additional global reset BIST configuration obtains 100% fault coverage (including routing resources)

Fault Coverage (FC) vs. BIST Configurations

Bar chart showing fault coverage for different BIST configurations.
Architecture of the Virtex-4 I/O Buffer

Main Components of an I/O Buffer

- ILGOIC (Input to FPGA)
- OLOGIC (Output from FPGA)
- PAD_LOGIC
OLOGIC Block:-

- Sources output to pad
- 6 storage elements
  - 3 for tri-state control
  - 3 for output data
  - Both sets of registers have same functionality
  - Only upper register can be configured as either Flip-Flop or latch
- This is a sequential circuit when operated in DDR mode
ILOGIC Block:-

- Gets input from the pad
- Consists of:
  - 64 tap delay element (variable or fixed)
  - Flip-Flops (registered outputs and Double Data Rate (DDR) registers)
  - 3 different outputs
    - Unregistered direct connection
    - Different modes of operation of DDR registers
- Only upper register can be configured as either Flip-Flop or latch
Pad Logic:-

- **Virtex-4 buffers**
  - 69 I/O standards
    - Only 5 can be used in bi-directional mode
  - 7 drive capabilities
    - All can be tested in bi-di mode
  - 2 slew rate options
    - All can be tested in bi-di mode
  - Pull-up, Pull-down and Keeper circuits
    - All can be tested in bi-di mode
Proposed Virtex-4 Implementation

- Circular comparison architecture
  - Multiple TPGs to prevent fault masking
- All programmable I/O buffers are identical
  - 32 multiplexers plus 10 flip-flops per I/O buffer
    - # I/O buffers
      - 320 for LX15, SX25, FX12, FX20 (smallest devices)
      - 960 for LX100, LX160, LX200 (largest devices)
  - All can be tested concurrently
    - Only one set of BIST configurations
- Seven BIST configurations total
  - Tests only bi-directional I/O voltage standards
  - Does not test all routing resources associated with I/O buffers
Fault Detection in Virtex-4

- Can only test bi-directional buffer modes
- Undetectable faults due to bi-directional mode

![Graph showing fault detection and coverage for different configurations and cores]

- Individual FC
- Cumulative FC
- Embedded processor cores
  - PowerPC hard core
  - MicroBlaze soft core
Capabilities and Limitations

- Can detect all catastrophic faults in routing, logic, and configuration memory bits
- Cannot detect some parametric faults
  - $V_{IL}$, $V_{IH}$, $V_{OL}$, $V_{OH}$, delay, drive capabilities, etc.
- Can be used for manufacturing testing
  - Package independent test
- May not always be usable for system level testing without some failures
  - Must be able to tri-state all outputs of other devices on PCB
  - Sensitive to system component connections

Example: LED bias current
Summary

- **Previous BIST approaches for FPGAs**
  - Programmable logic and routing in core of FPGA

- **No previous BIST for programmable I/O buffers**
  - Unbonded I/O frequently used by synthesis tools for additional logic/routing

- **BIST approach for I/O buffers**
  - Circular comparison based approach
    - Good fault detection capabilities and diagnostic resolution
  - Embedded processor (hard or soft) reduces test time
    - Dynamic partial reconfiguration
    - BIST execution, results retrieval and diagnosis
  - Can detect catastrophic faults in logic and routing
    - But cannot detect all parametric faults
System-Level BIST for Programmable I/O Buffers in FPGAs and SoCs

Lee Lerner, Sudheer Vemula and Charles Stroud
Electrical and Computer Engineering
Auburn University

may be presented at 2006 IEEE North Atlantic Test Workshop
Outline

- System Level BIST
- Back driving problem
- BIST Generation
  - VHDL approach
  - MGL or XDL approach
- Reconfiguration from embedded processor
- Summary
System Level Implementation

- Testing FPGA/SoC which is already placed on the PCB
- The FPGA/SoC will have input and output connections
Back Driving Problem

- Problems due to back drive
  - Signals from other devices on the PCB may cause BIST failure indications
    - Even when tri-stated
  - FPGA may be damaged due to high current
BIST Generation

**VHDL Approach:**

- Limited to supported VHDL macros
- Develop a parameterized VHDL
  - Cannot test unbonded I/O Buffers
  - Cannot route to unconnected signals
  - Less Fault Coverage
  - Parameterized VHDL can be developed easily
    - Can be applied for any FPGA
  - Still better than Boundary Scan
- The I/O Buffers placed using constraints file
  - Can be easily done by user
- Uses normal design flow
BIST Generation

- MGL or XDL Approach (High Effort):
  - MGL – Atmel’s Macro Generation Language
  - XDL – Xilinx Design Language
  - Parameterized program for MGL or XDL
    - Can test unbonded I/O Buffers
    - Can route to unconnected signals
    - Higher Fault Coverage
      - Better control over the resources
      - Most resources can be tested
    - More difficult development than VHDL approach
      - Cannot be applied for any FPGA (other vendors)
  - I/O Buffers placed by parameterized program
  - Does not use normal design flow
Embedded Processor Reconfiguration

- **Atmel**
  - Configuration memory is Byte addressable
  - Configuration memory readback is not present
    - Can't test logic if logic bits are mixed with routing bits
    - Lower fault coverage

- **Xilinx**
  - Configuration memory is Frame addressable
    - No. of 32-bit words per frame = 1312 (Virtex 4)
    - Longer partial reconfiguration time
  - Configuration memory readback is present
    - Read-Modify-Write operations to configuration memory
    - Logic can be tested independent of routing
    - Higher fault coverage
Write-only Reconfiguration Problem

- **Initial Configuration:** Q = ‘1’, B = ‘1’
- **Second Configuration:** Q = ‘0’, FF = ‘1’
- Can’t be written without knowing the multiplexer select line (A, B, C, D or E)
  - With configuration memory read back, selected bits can be modified
VHDL Experimental Results

- **Atmel** - using existing BIBUF VHDL macro
  - Portable between AT40K series FPGA and AT94K series SoC
  - Can only test about 25% of resources

- **Xilinx** – using existing IOBUF VHDL macro
  - Portable between all Xilinx FPGAs
  - Resources that can be tested varies with FPGA family and synthesis tools
    - Limited control of resources under test
    - Flip-flops not always synthesized into I/O buffers
System-Level BIST Summary

- Two approaches
  - VHDL
    - Limited by macro support
  - MGL/XDL

- Boundary Scan comparison
  - Higher fault coverage
    - Unless FPGA supports INTEST
  - Shorter test time & lower memory storage
    - Further reductions with partial reconfiguration via embedded processor

<table>
<thead>
<tr>
<th>Test</th>
<th>VHDL</th>
<th>MGL/XDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>some</td>
<td>yes</td>
</tr>
<tr>
<td>Routing</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Drive/slew</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Pull-up/down</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Non-bonded</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>I/O Voltages</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Fault coverage</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>Development</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>Portability</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

* Can detect catastrophic faults but not all parametric faults