Development of FPGA-Based Verification Simulation Accelerator

Presenter: Gary Burke  
Authors: Gary Burke and Jane Oh  
Affiliation: Jet Propulsion Laboratory, California Institute of Technology  
Address: 4800 Oak Grove Drive, Pasadena, CA 91109  
Email: Gary.R.Burke@jpl.nasa.gov  
Phone: 818-354-1616  
Fax: 818-393-4272
Presentation Outline

• Key Challenge
• Long Term Objectives
• Technical Approach Highlights
• Expected Benefits
Objectives

• Key challenge:
  – FPGAs are becoming a more critical component of space systems. Techniques and methodologies for assuring and verifying FPGAs must be developed that adequately address the increased complexity of the devices required by today’s missions.

• Long term objectives:
  – Reduce the amount of time required to verify critical FPGAs by an order of magnitude
  – Develop assurance methodologies that applied during the development of FPGAs will reduce by 50% the defect escapes
Technical Approach Highlights

• Design of Test FPGA
  – Define sampling nodes, sampling speed, muxing and buffering
  – Define insertion strategy, pattern buffers, and sequential behavior
  – Define block diagram, communication ports, buffers, and map onto existing part

• Development of Test FPGA
  – Define I/Os, block diagram, and function of all blocks
  – Write verilog code, test benches, verification matrix, and simulate

• Development of Test Software
  – Define registers, commands, data format,
  – Define software architecture, modeling strategy, user interface, and error detection
  – Write C code, test cases, verification matrix, and simulate
Typical FPGA Verification

- **Simulation**
  - Extensive HDL test benches
  - Model external world

- **Breadboard**
  - FPGA COTS or Custom Board
  - Re-programmable FPGA preferred
  - Use Bench test Equipment (BTE)

- **System test**
  - Engineering Model (EM) in system
  - Run system with software, external hardware

- **Assembly and Test**
  - Assemble final board and test in system.
Issues

• Simulation will get most of the problems
  – Time consuming to cover every case
• Subtle errors remain due to
  – Some cases not covered in test bench
  – External world not modeled correctly
  – Unexpected interaction with other components
• Breadboard test will get most of remaining problems but difficult to find source of problems
  – Lack of probe points inside FPGA
    • Need to bring out internal nodes onto unused pins
  – Lack of probe points on board
    • Difficult to probe small parts
  – Hard to set up error conditions
Issues - continued

- Problems at EM level will cause major cost and schedule delays
- System hard to probe, embedded cards in chassis.
- Subtle bug hard to capture

- Worse case is problem manifests itself at Final Assembly and Test
- Problem due to last minute updates/ modifications not in EM system
- Very costly to debug and fix

- Worst case is problem appears after launch!
  - Possible loss of mission
Testability

- Testability needs to be included in design
- Test ports on FPGA and board will help
  - BB test
  - EM test
  - Final A&T
- Using standard format Test Port allows re-use of generic BTE
Existing FPGA debug

• Existing methods for probing an FPGA during test exist and are effective

• Chipscope is very useful as a way to probe Xilinx FPGAs
  – Uses embedded code compiled with user code
  – Uses JTAG port

• Silicon Explorer
  – Uses FPGA structure to probe any node
  – Uses JTAG/Probe pins
New Debug Tool

- Works with any FPGA
- Has more capability than existing tools
  - See chart
- Provides a standard FPGA BTE for any Board
- Speeds up debug process by giving
  - High visibility of FPGA nodes
  - Easy user interface
  - Comparison of actual with model
  - Method to step to sequence causing the problem
## Expected Benefits

<table>
<thead>
<tr>
<th></th>
<th>Silicon explorer</th>
<th>Chipscope</th>
<th>Our Test FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General purpose</strong></td>
<td>No: Actel FPGA only</td>
<td>No: Xilinx FPGA only</td>
<td>Any FPGA/ASIC</td>
</tr>
<tr>
<td><strong>Monitor internal nodes</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>At speed monitoring</strong></td>
<td>Yes: but high speed problem with signal integrity</td>
<td>Yes: but limited sample size</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Number of signals monitored</strong></td>
<td>2-4 only (depends on device type)</td>
<td>limited by internal memory: typical 16</td>
<td>limited by bus bandwidth; typically 100</td>
</tr>
<tr>
<td><strong>Logic analyzer display</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Comparison against model</strong></td>
<td>No</td>
<td>No</td>
<td>Yes: by comparing against the model, problems can be found before they have a major effect on the I/Os</td>
</tr>
<tr>
<td><strong>Static Stimulus</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes: large number of static stimulus possible</td>
</tr>
<tr>
<td><strong>Dynamic stimulus</strong></td>
<td>No</td>
<td>No</td>
<td>Yes: full pattern generator included. This allows easy setup of conditions leading to possible problem.</td>
</tr>
<tr>
<td><strong>Internal FPGA resources needed</strong></td>
<td>No</td>
<td>Yes: large amount of on-chip storage needed to store results</td>
<td>Yes: but no onchip storage needed, and on-chip logic is a very small overhead.</td>
</tr>
</tbody>
</table>