



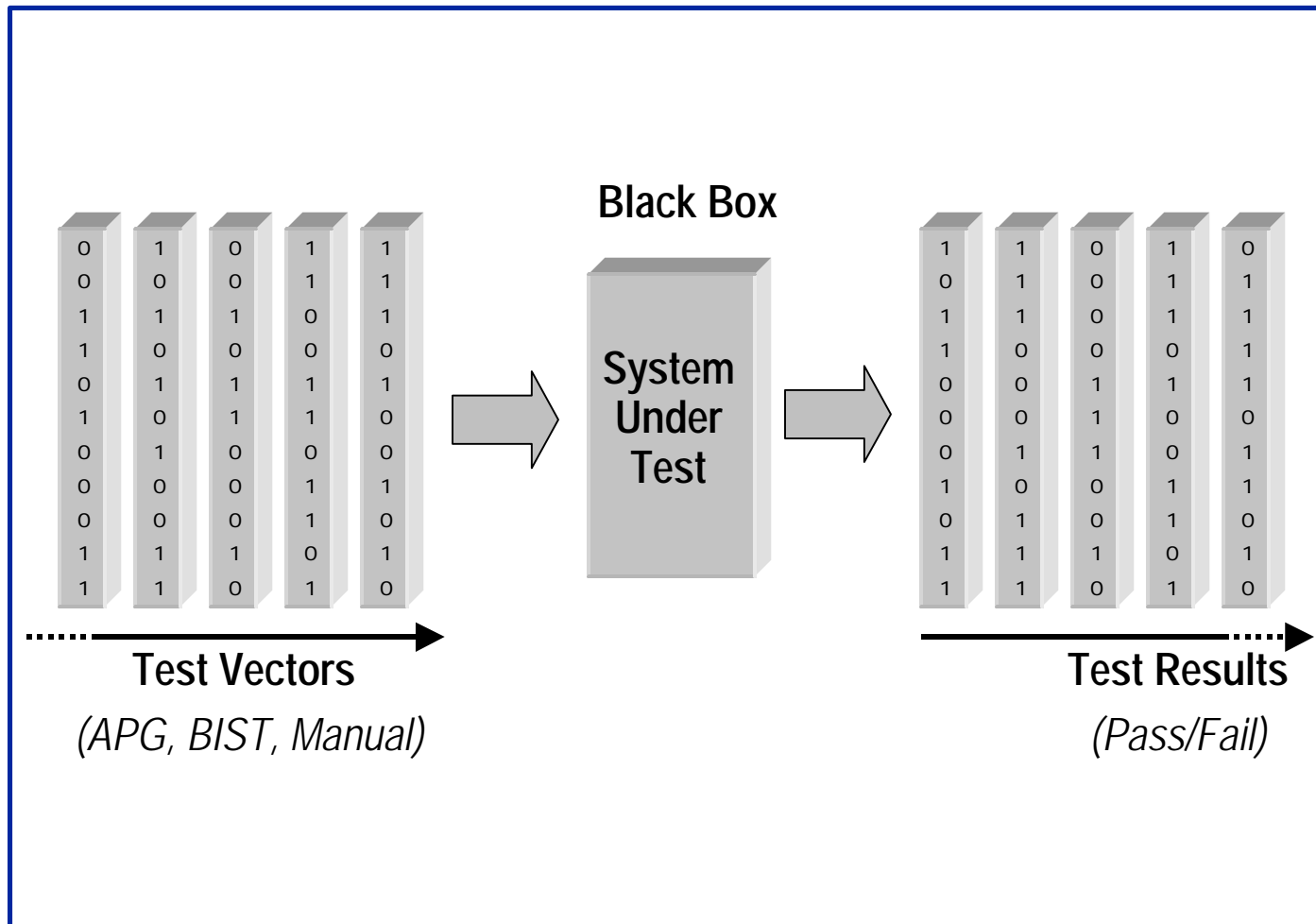
FPGA Device Test Using *JBits*

Prasanna Sundararajan
Steven A. Guccione
Scott McMillan

Outline

- ◆ Traditional Testing Methods
- ◆ FPGA Testing
- ◆ JBits based testing
- ◆ Results
- ◆ Future Work

Hardware Test: Overview



FPGA Test

- ◆ Typical approach:
 - Configure the device with test circuit
 - Exercise the test circuit with vectors
 - Interpret the output: Pass / Fail?
- ◆ Very high level of configurability
- ◆ Traditional tools to design test circuits
- ◆ Traditional Testers & BISTs

FPGA Test Problems

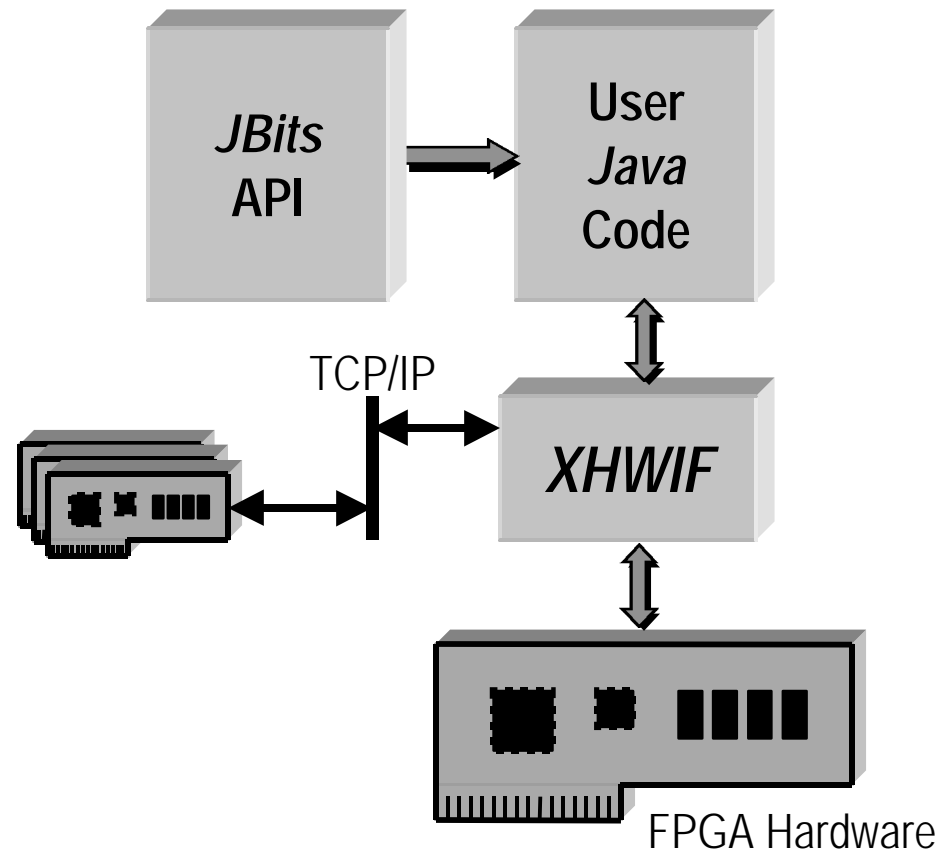
- ◆ Tools with less control over configurability
- ◆ Full coverage becomes difficult with large number of resources
- ◆ I/O bandwidth limitations
- ◆ Little support for Fault location and Isolation



JBits FPGA Test

- ◆ Use *JBits* to test low-level device components: Wires, LUTs, Flip Flops, IOBs
- ◆ Set and Probe components
- ◆ Use RTR to exercise individual device components
- ◆ Use readback to verify results of test
- ◆ Not only test, but detects and isolates faults

The *JBits* Test Environment



JBits FPGA Tests

- ◆ FPGA Tests:
 - Configuration memory test
 - LUT memory test
 - Flip flop memory test
 - Wires test
 - Timing/Delay Test

Configuration Memory Verification

- ◆ Verifies that all configuration bits are functional
- ◆ Configuration memory verification process:
 - Download configuration data
 - Readback configuration data
 - Compare individual bit values
- ◆ This process should be repeated until all bits have been set to both 0 and 1 for full coverage

Configuration Memory Verification (cont.)

- ◆ Single piece of code for all devices
- ◆ No recompilation necessary for new devices
- ◆ Does not rely on *JBits* API; uses the *XHWIF* portable hardware interface

Configuration memory testing does not verify wires, flip flops, LUTs or any other circuit elements

Configuration Memory Verification (cont.)

- ◆ Simple bitstream "diff":

```
/* Download bitstream */
board.setConfiguration(device, bs1);

/* Readback bitstream */
bs2 = board.getConfiguration(device, bytes);

/* Compare bitstreams */
for (i=0; i<bitstream1.length; i++)
    if (bs1[i] != bs2[i])
        System.out.println( Bitstream diff
                               in byte    + i);
```

LUT Memory Test

- ◆ Write, then read LUT values:

```
/* Set LUT values */
for (row=0; row<jBits.getClbRows(); row++)
    for (col=0; col<jBits.getClbColumns(); col++){
        jBits.set(row, col, LUT.SLICE0_F, 0);
        jBits.set(row, col, LUT.SLICE0_G, 0);
        jBits.set(row, col, LUT.SLICE1_F, 0);
        jBits.set(row, col, LUT.SLICE1_G, 0);
    }

/* Download bitstream */
bs1 = jBits.getAllPackets();
board.setConfiguration(device, bs1);
```

(cont.)

LUT Memory Test

```
(cont.)

/* Readback bitstream */
bs2 = board.getConfiguration(device, bytes);

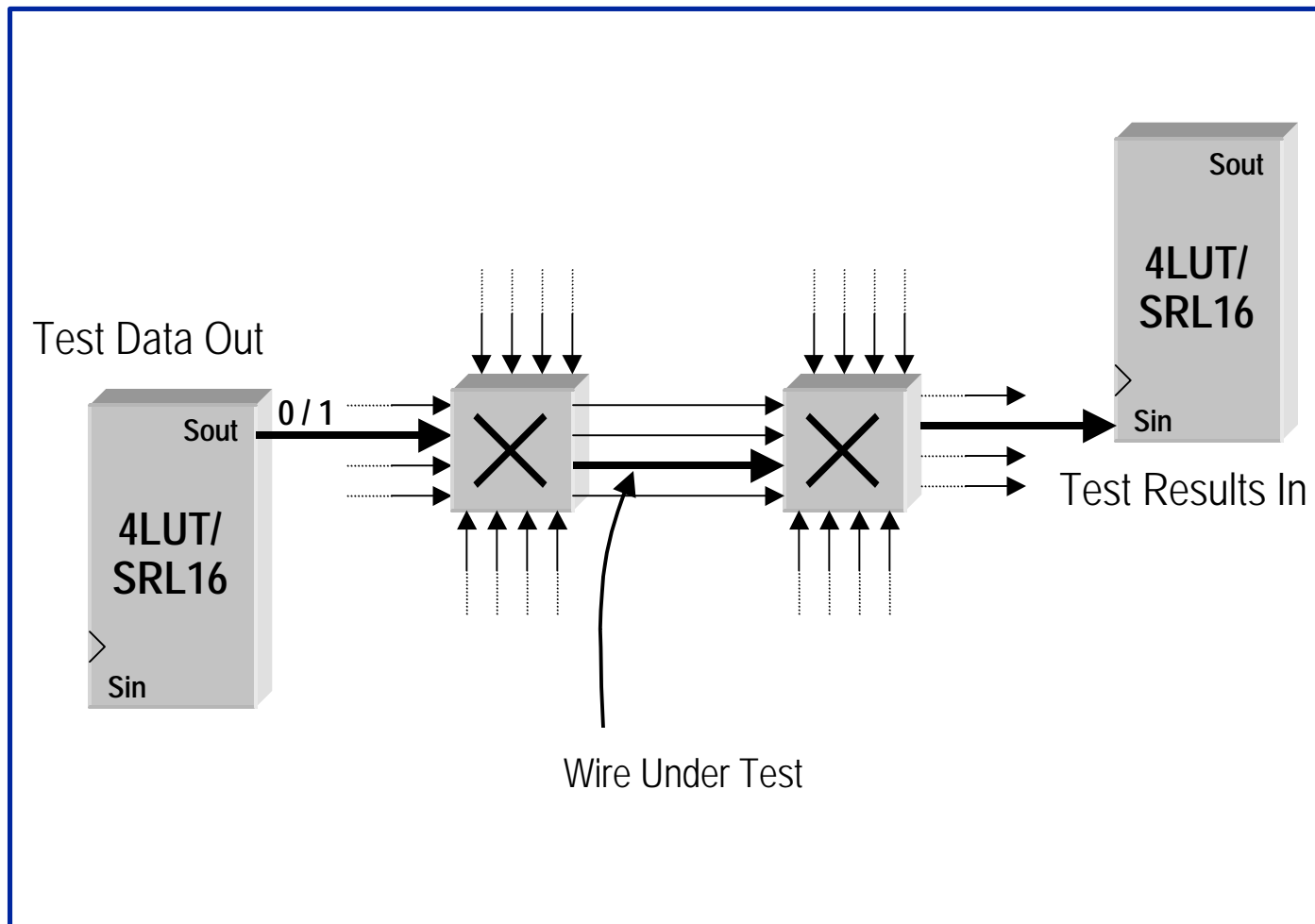
/* Set new bitstream data */
jBits.setClbConfig(bs2);

/* Check LUT values */
for (row=0; row<jBits.getClbRows(); row++)
    for (col=0; col<jBits.getClbColumns(); col++)
        if ((jBits.get(row, col, LUT.SLICE0_F) != 0) ||
            (jBits.set(row, col, LUT.SLICE0_G) != 0) ||
            (jBits.set(row, col, LUT.SLICE1_F) != 0) ||
            (jBits.set(row, col, LUT.SLICE1_G+ != 0))
            System.out.println( Error in LUT(  + row +
                , + col + ). );
```

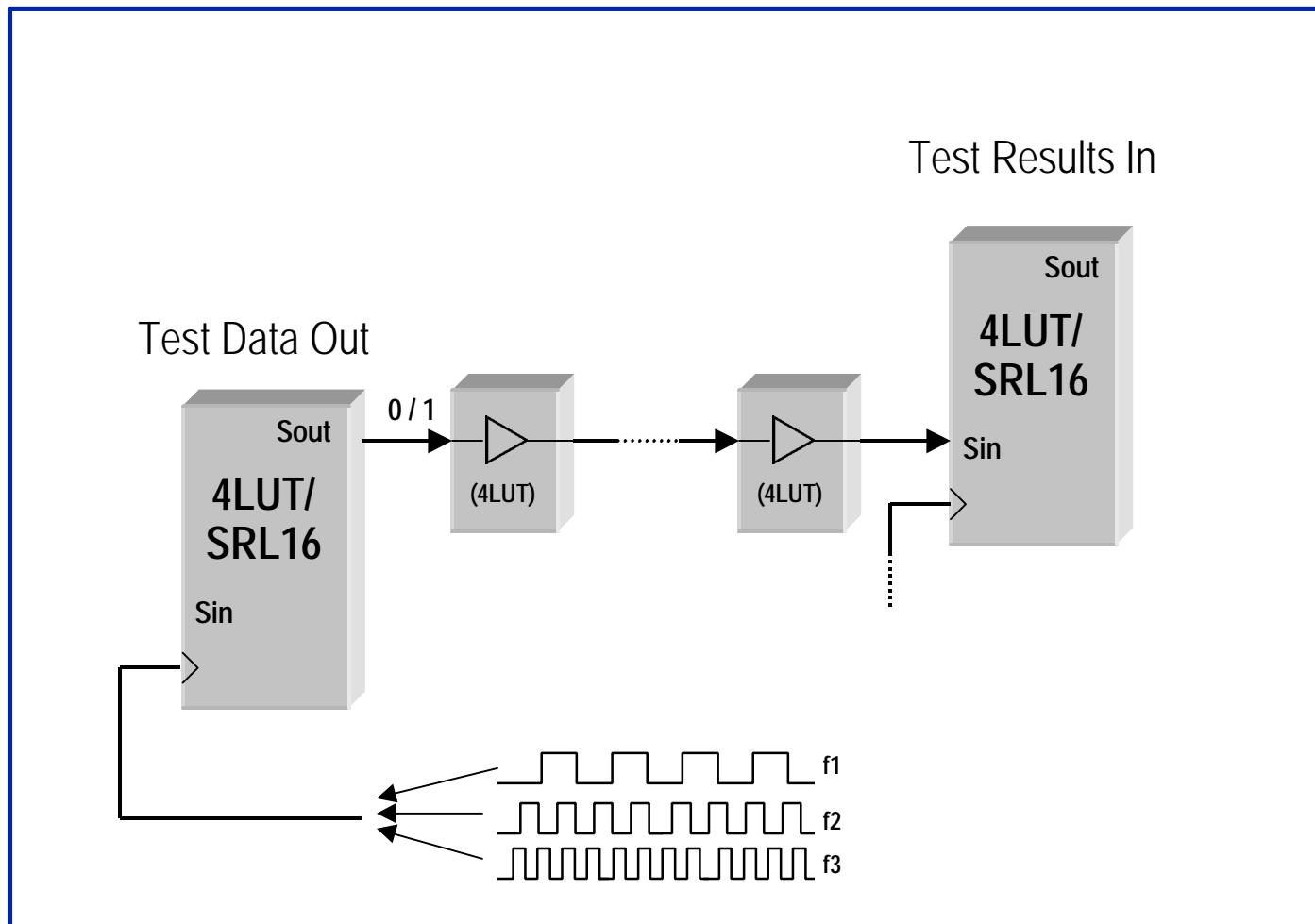
Wires Test

- ◆ Test each wire individually
- ◆ Several wires can be tested in parallel across the device
- ◆ General plan:
 - Drive a wire with a 1 / 0
 - Use a Virtex SRL16 to gather several samples before readback
 - Readback and verify the output

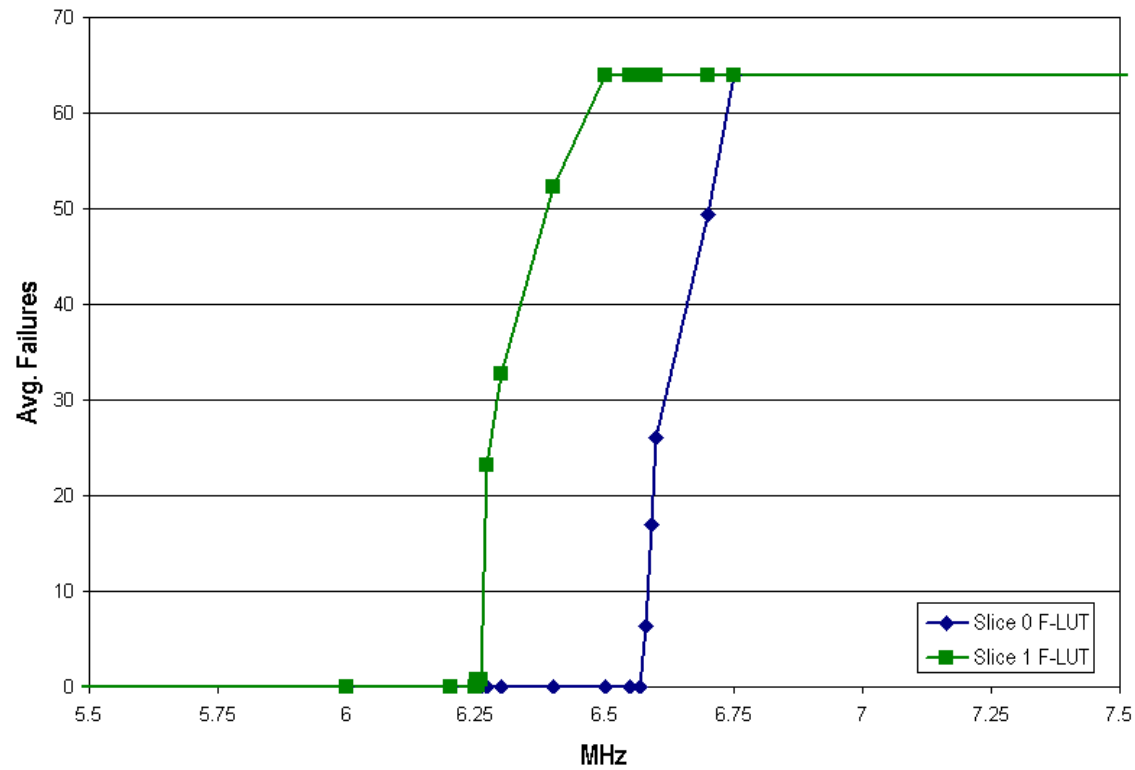
Wire Test



Timing Test



Timing Experiment Results



Timing Test

- ◆ Determine device timing characteristics
 - Clock test circuits at varying frequencies
 - Check for failures
- ◆ Can be used to speed-grade the devices
- ◆ Can also be used to test timing variance across device

Uses for *JBits* FPGA Device Test

- ◆ Factory testing
 - Verify newly manufactured devices
- ◆ Factor failure analysis
 - Isolate faults in newly manufactured devices
 - Gather statistics of types of faults
- ◆ End-user diagnostics / failure analysis
 - Run by FPGA users on fielded systems
 - Can help to diagnose and isolate problems

Future Work

- ◆ Develop Test Suites
- ◆ Use *JBits* to provide run-time test and diagnostics in-system
 - Useful to high-reliability systems
 - Can be used in conjunction with Defect Tolerance techniques to provide run-time Fault Tolerance
- ◆ Expand tests to measure component performance to aid in characterization (speed vs. temperature, etc)

Conclusions

- ◆ *JBits* methodology to test FPGAs
- ◆ Allows complete FPGA device-level test
- ◆ Defects can be isolated, not just detected
- ◆ Useful in factory test, post-mortem test and in-system diagnostics
- ◆ A necessary component of on-line fault tolerance