INTERCONNECT TESTING WITH BOUNDARY SCAN

Paul Wagner
Honeywell, Inc.
Solid State Electronics Division
12001 State Highway 55
Plymouth, Minnesota 55441

Abstract

Boundary scan is a structured design technique which can be used to simplify the testing of digital circuits, boards, and systems. With boundary scan, test patterns can be generated which provide 100% stuck-at and bridging fault coverage of board interconnections. The paper describes the advantages and disadvantages of boundary scan along with the application and implementation of boundary scan circuitry. Algorithms for generating interconnect test patterns for stuck-at and bridging fault coverage are also presented

Introduction

Advances in VLSI technology have increased the density and speed of integrated circuits. Thus, the complexity and cost of testing digital integrated circuits, boards, and systems have also increased. By providing a simple means to access the periphery of digital circuits, boundary scan can greatly simplify the task of testing and maintaining systems which use these circuits. This advantage allows boundary scan to reduce the costs of wafer-level IC testing, board and system testing, and system field maintenance.

Wafer-Level Testing

At the wafer level, boundary scan can be used to reduce the need for complex probing fixtures and high-pin-count testers[1]. By using boundary scan to access the primary chip 1/O, a simple probe card consisting of power, ground, and serial test interface signals can be used to test chips with hundreds of 1/O pads[2]. The decrease in fixturing complexity simplifies test setup, reduces test fixturing costs, and reduces the possibility of damaging the device-under-test during probing.

Besides simplifying testing fixturing, boundary scan also reduces test equipment requirements. Since the boundary scan path provides access to the primary I/O, the testing process is reduced to serially shifting the test pattern into place, executing one or more clock operations, and serially shifting out the results as the next pattern is shifted in. Thus a small, inexpensive testing computer can be used to perform chip testing. This simple setup is shown in **Figure 1.**

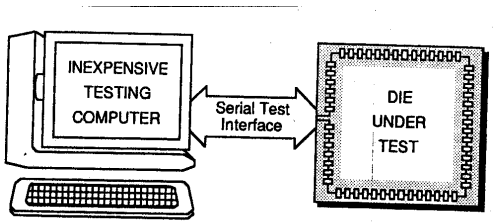

Figure 1: Wafer-Level Testing With Boundary Scan

At the board level, boundary scan can be used to resolve testing difficulties introduced by new packaging technologies associated with surface mount devices and mufti-chip packages. Traditional methods for digital board testing include through-the-hole probing to gain access to the primary component I/O with a "bed-of-nails" testing fixture. Difficulties with the "bed-of-nails" approach include degraded reliability due to over-driving connections from other board components, physical limitation of through-the-hole accessibility, difficulty of reproducing tests, and expenses involved with developing the "bed-of-nails" testing fixture[3,4]. These problems, combined with the increasing use of surface-mount technology[5] and the need for high speed and high pin count testers, have resulted in extremely expensive board-level testing costs.

Boundary scan can reduce the problems associated with board-level testing. As shown in Figure 2, boundary scan provides serial access to the primary component I/O and their interconnections. This allows any component to be partitioned from the rest of the board during testing and eliminates the need for a "bed-of-nails" testing fixture. Also, boundary scan reduces the time and cost associated with test pattern generation because test patterns used on the component at wafer level can be modified and applied through the boundary scan path. This can be useful when components are purchased from outside vendors and

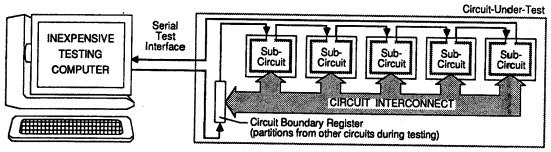

Figure 2: Board-Level Testing With Boundary Scan

Figure 2. Board-Level Testing with Boundary Scan

knowledge of the internal circuitry is limited. Since board interconnections are easily accessed, simple algorithms can be used to generate test patterns which provide 100% stuck-at and bridging fault coverage.

As was the case at wafer-level testing, boundary scan greatly simplifies the setup required for board testing as shown in Figure 2. This setup reduces testing costs because the test patterns can be applied serially with an inexpensive test computer through a simple test interface consisting of the boundary scan-in signal, boundary scan-out signal, and necessary control lines.

Field Testing

Boundary scan can also reduce the cost of system field maintenance. Since boundary scan tests the input buffers, the output buffers, and all *component interconnect, it* provides excellent coverage of the most common field failures. Furthermore, the procedure for testing with boundary scan in the field is nearly identical to that described for board-level testing. Thus field testing can be performed using a simple testing computer accessing a serial test interface. Since very few interconnect test patterns are required, the testing computer can be as simple as a lap-top personal computer, which is ideally-suited for field maintenance.

Boundary scan can be used to test the system interconnections and to partition the system into separately-tested modules. In this case, testing will isolate the fault to a single module or to a faulty interconnections) if the individual modules can themselves can be adequately tested. If boundary scan is extended to the component level, the fault can be isolated to the individual *component. Thus,* cost-effective repair of the module is possible since the faulty component or interconnection can be easily identified for replacement or repair.

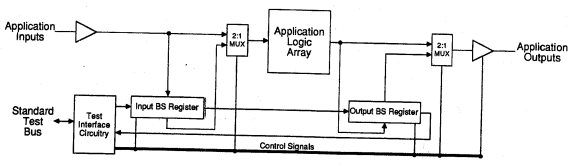

Figure 3: A Conceptual Diagram For Boundary Scan

Figure 3. A Conceptual Diagram for Boundary Scan

Boundary scan can also be used as part of a system self-test strategy. By allowing a system test processor to access the boundary scan paths in the system, boundary scan can be used to test the system interconnections and to partition the system into smaller self-testable units. The easy execution of self-test and improved fault isolation provided by boundary scan reduce the mean-time-to-repair; thereby increasing system availability.

IMPLEMENTING BOUNDARY SCAN

Before actually implementing boundary scan, a number of options must be considered which affect both the design and capabilities of the boundary scan circuitry. These options include: the use of application registers as boundary scan registers, the control of output buffers, the selection of a test interface, and the implementation of the boundary registers. These options and others are addressed in the following sections on implementing the components of a boundary scan technique.

Dedicated Boundary Scan Registers

Boundary registers can either be dedicated for boundary scan testing or they can be used in both functional and test modes. When implementing boundary scan on high-speed bipolar integrated circuits, we found that there were a number of advantages to using some functional registers for boundary scan testing. First, the high-speed of the system mandates that most of the chip inputs and outputs be registered directly at the I/O buffer. Since we already incorporate serial scan[6] in our chip designs, these registers were easily added to the boundary scan path. Dedicated boundary "shadow" registers are then added to any I/O which are not directly registered. A 2:1 multiplexer is used to make the shadow registers visible during the test mode and invisible during the functional mode. This approach of exploiting existing registers substantially reduces both the circuit and power overhead associated with boundary scan and eliminates a 2:1 multiplexer delay from the path of critical signals.

If boundary scan is to be implemented on a gate array product, associating dedicated shadow scan registers with the I/O buffers at the periphery of the array has a number of advantages. First, the user of the gate array can utilize boundary scan with little or no design effort. Furthermore, array cells are not consumed when implementing boundary scan and numerous signal routings are eliminated. Finally, implementing dedicated boundary scan registers on a CMOS gate array product2 will not significantly increase the chip power (contrary to bipolar designs).

The Boundary Scan Bit-slice

The boundary scan registers consist of bit-slices that are attached to each application input and output. Our implementation of this bit-slice is shown in **Figure 4.** This consists of a 3:1 multiplexer which allows data to be loaded in the functional mode, serial data to be shifted in the test mode, and a reset operation to be performed. The output of the multiplexer is then fed to the scan flip-flop which in turn drives the scan out signal and the chip output.

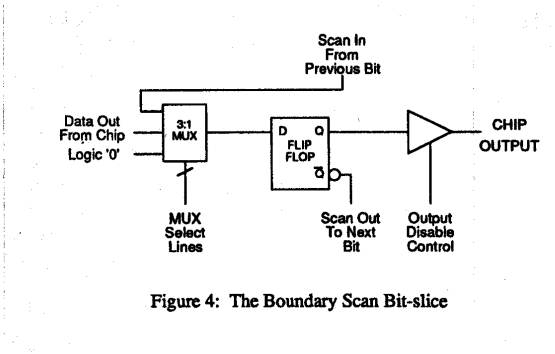

Figure 4: The Boundary Scan Bit-slice

As data is shifted through the boundary scan path, the chip outputs must be latched or disabled to prevent unwanted and possibly damaging output conditions. For example, the scan operation could damage output buffers by forcing two separate output drivers on the same net to different logic levels. Also, the shifting operation may cause a large number of output buffers to change state at the same time; resulting in excessive noise on power and ground busses. For these reasons, a global output buffer disable signal is included in our implementation of boundary scan and can be controlled by the test interface circuitry.

If the testing of asynchronous sequential logic is necessary, a latch must be added to between the flip-flop and output buffer to hold the output state during shifting operations. A similar latch would also be required at the input boundary register if the application logic array contained asynchronous sequential logic to be tested with the boundary scan circuitry. Typically, we do not include this latch because we infrequently use asynchronous sequential logic in our digital system designs.

Another implementation concern involves connecting the boundary scan bit-slices as inverting serial shift registers or as non-inverting serial shift registers. The advantages of an inverting serial shift path include the easy identification of faults in the shift path. To test the shift path, the entire path is reset to either a logic 0 or a logic 1 and the contents are shifted out. The serial output pin is then examined for an alternating pattern of ones and zeros. If the data remains at a logic 1 or 0 after k clocks, then we know that a fault exists k bits back from the output pin. With this information, we can quickly isolate the cause of the fault. Without the inverting boundary scan path, finding the fault could be tedious and difficult task. For this reason, we frequently make use of inverting serial shift paths when implementing boundary scan.

The Test Interface

Selecting an appropriate test interface is a very important part of the boundary scan implementation. A common interface will allow the boundary scan paths of multiple chips on a complex circuit board to be easily accessed. Without this common interface, many of the advantages of using boundary scan at the board level are diminished due to the difficulty in using the technique.

To resolve this problem, we are using the VHSIC standard Element Test and Maintenance Bus[7] (ETM-Bus) as our serial test interface for boundary scan and other on-chip design-for-test techniques[8]. If the serial test bus is to be connected solely to on-chip boundary scan, a simplified version of this interface logic can be used

INTERCONNECT TEST PATTERN GENERATION

When testing interconnection nets on a digital module, both stuck-at and bridging faults must be considered. Since the boundary scan path provides direct access to these nets, test patterns can be generated which provide 100% coverage of these faults. The following sections discuss algorithms we use for the generation and application of boundary scan test patterns which detect all possible stuck-at and bridging faults.

Stuck-at Fault Test Pattern Generation

Because stuck-at faults occur on a variety of bus configurations, different test pattern generation algorithms are required for wired-AND, wired-OR, and three-state interconnect nets.

Testing wired-AND interconnection nets. As the name implies, the values forced on a wired-AND interconnection net are logically ANDed to obtain the resulting value. Thus, the wired-AND net can be treated in the same way as an AND gate where 100% of all the stuck-at faults can be detected with k + 1 test patterns where k is the number of inputs. The test patterns can be divided into k patterns which test for stuck-at '1' faults and a single pattern which tests for all stuck-at '0' faults. Figure 5 shows the steps we use for testing wired-AND interconnection nets.

1) The driver to be tested is set to a logic '0'

2) All other drivers on the net are set to a logic ' 1'

3) The data is clocked into the receivers

4) All receivers on the net are examined for a logic '0'

5) Repeat steps 1-4 until each driver is tested

6) Every driver is set to a logic '1'

7) The data is clocked into the receivers

8) Every receiver is examined for a logic '1'

Figure 5: S-A Faults Testing Steps for Wired-AND Nets

Testing wired-OR interconnection nets. Generating test patterns for a wired-OR interconnection net is nearly identical to the wired-AND case. For a wired-OR net with **k** drivers, 100% of all stuck-at faults can be detected with **k + 1** test patterns. In this case, the test patterns can be divided into k patterns which test for stuck-at '0' faults and a single pattern which tests for all stuck-at '1' faults. Figure 6 shows the steps we use for testing wired-OR interconnection nets.

1) The driver to be tested is set to a logic '1'

2) All other drivers on the net are set to a logic '0'

3) The data is clocked into the receivers

4) All receivers on the net are examined for a logic '1'

5) Repeat steps 1-4 until each driver is tested

6) Every driver is set to a logic '0'

7) The data is clocked into the receivers

8) Every receiver is examined for a logic '0'

Figure 6: S-A Faults Testing Steps for Wired-OR Nets

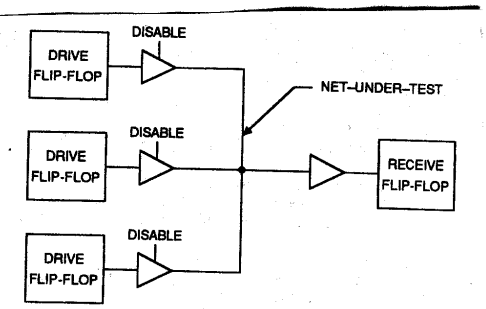

Figure 7: A Three-state Interconnection Net

Testing three-state interconnection nets. When a three-state interconnection net is used, multiple drivers control one or more receivers as shown in Figure 7. Since only a single driver can be enabled at any one time, a special restriction is imposed on the generation of the three-state interconnect test patterns. In order to achieve 100% stuck-at fault coverage, each driver on the net must be individually for stuck-at '1' and stuck-at '0' faults while the remaining drivers are disabled. Since this requires **2** test vectors per driver, 100% stuck-at fault coverage can be achieved using **2 • k** test vectors where k is the number drivers on the net. The steps we use for testing three-state interconnect nets are shown in Figure 8.

1) The driver to be tested is enabled and set to a logic '1'

2) All other drivers are set to a logic '0' and disabled

3) The data is clocked into the receivers

4) The receivers are examined for a logic '1'

6) Repeat steps 1-5 until all drivers have been tested

7) The driver to be tested is enabled and set to a logic '0'

8) All other drivers are set to a logic ' 1' and disabled

9) The data is clocked into the receivers

10) All receivers are examined for a logic '0'

11) Repeat steps 7-11 until all drivers have been tested

Figure 8: S-A Fault Testing Steps for Three-state Nets

Bridging Fault Test Pattern Generation

In addition to testing for stuck-at faults, we also test the interconnects for bridging faults. A bridging fault occurs when two nets are electrically connected as shown in Figure 9. A procedure which detects this fault is described in Figure 10.

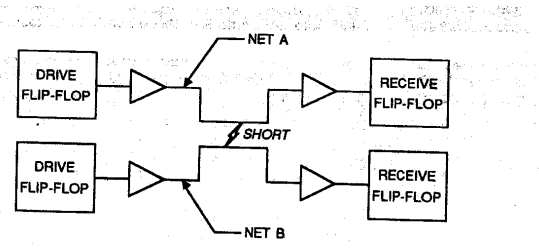

Figure 9: A Bridging Fault Between Two Nets

1) Enable the drivers on each net

2) Apply a logic '1' to all drivers on the first net

3) Apply a logic '0' to all drivers on the second net

4) Clock the data into the receivers

5) Examine at least one receiver on each net

6) If the data at the receiver of either net does not correspond with the data applied at the respective driver, then a bridging fault exists between the nets

Figure 10: The Procedure for Detecting a Bridging Fault

The procedure described in Figure 10 operates on two nets. Since a digital module may contain hundreds of interconnection nets, this procedure must be applied to every possible pair of nets to achieve 100% bridging fault coverage. Since separate pairs of nets can be tested at the same time, 100% bridging fault coverage can be achieved with $\log_2(n + 2)$ test vectors where n is the number of nets on the board[9].

Bridging fault test generation example. The algorithm we use to generate the $\log_2(n + 2)$ test patterns for bridging fault detection is best illustrated through a simple example. The example given below uses a board with 8 interconnect nets.

Step 1 - Determine the total number of nets on the board In this example, n = 8 which requires $\log_2(8 + 2)$ or 4 test vectors.

Step 2 - Assign each interconnect net a unique number. Assignments should begin with the number 1 and continue in increments of 1. In this example, the first net is given the number 1, the second net is given the number 2, and the last net is given the number 8.

Step 3 - Assign binary values to each net. Since 4 test vectors are required, assign each net the 4-bit binary equivalent of the net number assigned in the previous step as shown in Figure 11.

```
interconnect net 1 - 0001
interconnect net 2 - 0010
interconnect net 3 - 0011
interconnect net 4 - 0100
interconnect net 5 - 0101
interconnect net 6 - 0110
interconnect net 7 - 0111
interconnect net 8 - 1000
```

Figure 11: The Binary Numbers Assigned to the 8 Nets

Step 4 - Determine the test vectors. The first test vector is comprised of all the bits in the least significant position of the binary numbers. The second test vector is comprised of the bits in the second least significant position. This is continued until all bit positions of the binary numbers have been used. The resulting test vectors are shown in Figure 12.

```
test vector 1 - 10101010
test vector 2 - 01100110
test vector 3 - 00011110
test vector 4 - 00000001
```

Figure 12: The Bridging Fault Test Vectors for the 8 Nets

Isolating the faulty interconnects. The bridging fault test pattern generation scheme described in the previous section provides a quick and easy method of bridging fault detection. Although this scheme determines if any bridging faults exist, it does not isolate every interconnection net with a bridging fault. If repairing interconnection nets with bridging faults is possible, all of the faulty interconnects need to be identified. This can be accomplished using the test patterns generated by the algorithm described in the previous example along with an additional $\log_2(n + 2)$ test patterns. Thus, $2 \cdot \log_2(n + 2)$ test patterns can be used to provide complete bridging fault isolation of the interconnection nets.

```
test vector 5 - 01010101
test vector 6 - 10011001
test vector 7 - 11100001
test vector 8 - 11111110
```

Figure 13: Additional Test Vectors for Isolating Faulty Nets

The additional $\log_2(n + 2)$ test patterns are generated by simply inverting the binary values of the first $\log_2(n + 2)$ test vectors. For the previous example, these test vectors are shown in Figure 13. To identify those interconnects with bridging faults, a list of the faulty nets can be maintained during testing. When a bridging fault is detected, the corresponding interconnect net can be identified and added to this list. After all the test patterns have been applied, the list will contain all of the faulty interconnection nets.

CONCLUSIONS

Boundary scan simplifies the testing of digital circuits, boards, and systems. Since boundary scan provides easy access to the periphery of digital circuits through a serial shift path, the setup needed for testing is simplified to an inexpensive computer and a simple test interface. This reduces the complexity and costs of wafer-level testing, board-level testing, and field maintenance.

Boundary scan allows easy partitioning of board components and interconnects, thus wafer-level test patterns can be modified and used to test the components on the board. Also, the simple algorithms presented generate test patterns which provide 100% stuck-at and bridging fault coverage of board interconnects. These advantages allow boundary scan to significantly reduce test and maintenance costs while maintaining a high percentage of fault coverage at the circuit, board, and system level.

REFERENCES

[1] J. J. Zasio, "Shifting Away From Probes For Wafer Test;" *COMPCOM S'83,* San Francisco,-CA, pp. 317-320.

[2] R. Lake, "A Fast 20k Gate Array With On-chip Test System;" VLSI Systems Design, June 1986, pp. 47-55.

[3] F. P. M. Beenker, "Systematic and Structured Methods for Digital Board Testing;" *IEEE International Test Conference 1985 Proceedings,* pp. 380-385.

[4] H. Bleeker and D. van de Lagemaat, "Testing A Board with Leaded and Surface Mounted Components;' *IEEE International Test Conference 1986 Proceedings,* pp. 317-320.

[5] W. Booth, "VLSI Era Packaging;" VLSI *Design,* December *1986,* pp. 22-35.

[6] H. W. Miller, "Design for Test Via Standardized Design and Display Techniques;" *Electronics Test,* October *1983,* pp. 38-61.