LEVERAGING XML AND WEB TECHNOLOGIES FOR BOUNDARY SCAN TEST DEBUG

David E. Rolince
Teradyne, Inc.

Introduction

There is no denying that the World Wide Web has spread to every corner of our technological cultural. Web browsers are used to carry us through the Internet and serve as the user interface to a wide array of applications. As its usage has increased, the data carrying capabilities of the web have been strained to the point where new data structuring methodologies were required to overcome the "world wide wait" and enable a whole new class of data to be transported over the web. This was a key motivation for the emergence of XML as means for dealing with the representation and transmission of complex and highly interrelated sets of data.

While the data produced by commercially available boundary scan test generation tools is not normally associated with web applications, it represents an excellent example of large amounts of highly interrelated data that can exploit XML and web browser technology. Most boundary scan test products provide automatic diagnostics that indicate the source of failures to a test operator during production testing. However, these diagnostics can be incorrect or misleading if invoked during initial test integration and program debug when the source of program failures is more likely to be unrelated to a physical defect on the unit under test (UUT). Nonetheless, the test engineer debugging the problem needs to refer to the boundary scan test database in the process of resolving it. By knowing the expected response of the test, the observed response of the test, and the state of other device leads, clues as to the source of the problem can be obtained. All the necessary data is normally available, but not in a form that is quickly and easily understood. By representing boundary scan data in XML and using a web browser to display the data, debugging failures encountered in testing electronic modules that incorporate boundary scan can be significantly simplified.

XML Background

Extensible Markup Language (XML) is a meta language that was developed by the World Wide Web Consortium in response to a growing need to increase the efficiency and speed of web-based applications, especially those that involved the processing and transfer of information. XML is human-readable, machine-understandable, and is applicable to a wide range of applications (databases, e-commerce, Java, web development, searching, etc.). The power of

XML is in its support for custom tags that enable the definition, transmission, validation, and interpretation of data between applications and between organizations.

Before XML was introduced in 1998, HTML was the primary language of the Web. HTML does an excellent job at defining the structure of a document as well as the style of its presentation in a web browser. It does a rather poor job of information exchange and does not allow for semantic interpretation of the contents of a Web page. XML was designed to make information self-describing through the use of specific tags that define what the information is. In this way, searches and organization of specific information can be performed quickly and without having to peruse through a mountain of unrelated or tangentially related data that often occurs in HTML-based searches.

Coupled with XML is XSL (Extensible Stylesheet Language). XSL is used to target database-neutral and device-neutral data marked up in XML to specific output devices, such as web browsers, for display. Linking abilities in XML allow for multi-way links. This is useful when you want to display multiple, related attributes of a characteristic specified in the link each in its own browser window. Links are defined in the XSL stylesheet.

Characteristics of Boundary Scan Test Data

The data used and generated in the process of developing boundary scan tests for opens and shorts at and between device leads, and wrong components for Unit Under Test (UUT) can be grouped into three sets:

- Topology – the physical layout and electronic component composition of the UUT
- Position – the arrangement of boundary scan data registers and associated device leads
- Test pattern – the serial vectors used to propagate faults in the UUT to tester channels for detection

The topology and position data sets provide a vast array of information about the interrelationships among boundary registers cells, physical components and their leads, and the input/output characteristics of interconnected device leads on nets. Boundary scan device chains are formed when the TDO lead of one device is connected to the TDI lead of another boundary scan device. Chains can consist of several devices thereby forming a continuous path extending from the TDI lead of the first device to the TDO lead of the last device. It is the position data set that contains the information about the order of scan cells in the chain, which can reach into the thousands for a UUT with several VLSI type devices. It is clear to see that a strict accounting of the position of scan cells in the chain and their functionality is important if faulty circuit activity is to be accurately diagnosed.

When topology and position data sets are merged with the test pattern data set, the state of each boundary register cell for a given test vector is known. When presented to a graphical display application, the juxtaposition of test pattern data with physical leads and boundary scan data registers is enormously valuable to a test engineer or test operator who is trying to debug a state mismatch problem when the test is applied to a UUT. In most cases, probing device leads on the UUT is not possible due to the lack of physical access, or, as in the case of control cells, the register of interest is internal to the device, thereby underscoring the value of a visual rendition.

Compounding the boundary scan test debugging effort is the enormous number of test vectors that typically exist. The length of a boundary scan test vector is directly related to the number of boundary register cells there are in the chain. At a minimum there is one boundary register cell for every boundary scan lead on a device. For bi-directional leads there are three, one for the lead as an input, one for the lead as an output and one control cell that determines the directionality of the lead. For optimum fault coverage and control, every bi-directional boundary lead should have its own control cell. On CPLD devices and other VLSI circuits, it is common to have 500 - 1000 boundary scan register cells. String several of these devices together and single scan vector can be tens of thousands of bits long.

The total number of test patterns will equal the number of boundary scan test vectors times the length of each vector. The Wagner algorithm (counting patterns and their complement) is a popular method used for automatically generating high fault coverage test patterns for detecting device interconnect failures. The number of vectors required to attain 100% interconnect fault coverage is related to the total number of boundary scan nets, and follows the formula

$$\#\text{ of vectors} = \log 2\,(n+2) \qquad \text{where } n = \#\text{ of boundary scan nets}$$

For example, a circuit board with 1000 boundary scan nets would require 20 vectors (10 counting vectors + their complement) to get 100% opens and shorts pin coverage. If there are 5000 boundary scan register cells associated with the complete scan chain, there would be 100,000 test patterns required. In practice, large numbers of device leads changing state simultaneously, which is the case with a boundary scan test when an update Data Register command is executed, can cause instability in a circuit board due to ground bounce. To reduce the effects of ground bounce, the test can be forced to toggle fewer nets during each vector. While this effectively increases the stability of the test, it also increases the total number of test patterns that need to be applied to attain the same level of high fault coverage.

Keeping track of a stream of 100,000+ test patterns and accurately correlating them with their associated boundary scan register cells and physical device leads, then displaying that information in a fast and simplified way requires a great deal of database organization.

Boundary scan test patterns can be expressed in many formats, but SVF (Serial Vector Format) is the most prevalent in the industry. One advantage of SVF is pattern compression – each SVF "bit" in a pattern stream is a hexadecimal digit. The disadvantage of this is readability and association of bits in the serial pattern stream with physical device leads or boundary register cells. All is well until a test fails, then the problem becomes trying to determine the cause of the failure. During the integration phase of a test program any number of things can contribute to the apparent failure – patterns out of sequence, electrical noise, incorrect BSDL, cabling or fixture errors, or other things unrelated to the UUT. Having the test pattern, UUT topology and boundary scan cell position data in a form that is easily displayed and easy to visualize interrelationships would be extremely helpful.

Applying XML to Boundary Scan Test Integration and Debug

Given the inherent flexibility and data-neutrality of XML, applying it to the large amount and diverse nature of boundary scan test would seem to help simplify the presentation of that data for initial test integration and debug. Since XML is a language for creating a markup language, the first step to harnessing XML is to develop an "instance document" which defines the XML schema for the specific information in your application. The instance document provides the rules that define the elements and structure of the new markup language. It will also serve as the guidelines for other developers to interface with the application.

In our application example, the schema for boundary scan test data is organized in three instance documents with main elements <circuit> for topology data, <boundary_scan_data> for position data, and <SerialVectors> for pattern data. Under each main element are sub-elements that further define the information contained in each schema. In this way XML allows the interrelationships of these schema to be expressed so that collections of related data can be easily located and displayed in a web browser. This is possible because the XML tags in the elements and sub-elements are defined to indicate specific information types. Let's look at how data is tagged in each of the three instance documents identified for boundary scan test data.

The Circuit instance document contains information about the physical composition of the UUT and is typically derived from CAD or netlist files. It is used to identify the components making up the UUT by reference designator and component type or class. It also can contain information about the leads of every component as well as connectivity between a particular lead and other leads that share the same net. The XML language syntax and tag identifiers for topology data might look like this:

```
<Circuit>
        <Devices>
                <Device> Name="U_17" Class="digital" LeadCount="84"
                        <Leads>
                                <Lead Name="2" Type="I">
                                        <Net>Data14</Net>
                                </Lead>
                                <Lead Name="34" Type="O">
                                        <Net>Addr07</Net>
                                </Lead>
                                  .
                                  .
                                  .
                        </Leads>
```

If the device is a boundary scan device, there could be additional tags for the lead as a virtual primary input or output relative to the other non-scan device leads on the net, or as a tri-state control lead.

```
<Device> Name="U_10" Class="digital" LeadCount="24"
        <Leads>
                <Lead Name="2" Type="OT">
                        <Net>Ctrlbit01</Net>
                        <Vpi>243</Vpi>
                        <Control Path="1" Position="135" >
                </Lead>
        </Leads>
</Device>
```

A key advantage of XML is that additional tags may be created to make reference to other information types that may be deemed to have importance.

The boundary_scan_data instance document, or position data, contains detailed information about the boundary scan components on the UUT and how the boundary register cells are arranged in the serial chain created by connecting the devices together. Position data is generally derived from the BSDL file of every boundary scan component type and the chain description file. The types of information contained in the BSDL files that is useful for test generation and debug include instruction and data cell attributes, relationships between data cells and physical device leads, and supported instructions and their op codes. A very important bit of information contained in BSDL files is the behavior of control cells on tri-state and bi-directional data registers. Tagging control cells and the leads they control makes this information, which is internal to the boundary scan device, available for applications used in debugging boundary scan tests. The chain description file defines the relative position each boundary scan component on the UUT occupies in the chain of components. When this information is provided as part of the position data set, the relative position of every boundary register cell can also be derived.

The example below illustrates a sample of XML language syntax and tag identifiers for the boundary_scan_data instance document. Note the organization of the information into sub-elements instruction_cells and data_cells. This was done because the end application generates a TAPIT test to verify the integrity of the boundary scan chain as well as tests for interconnected boundary scan devices and non-scan devices. This illustrates once again the power of XML to create tags that precisely specify information that can be exported to another application.

```
<boundary_scan_data>
  <position_data>
      <instruction_cells length="49">
          <device id="u_12" type="ti374" package="dw_package"
          length="8">
              <instruction id="bypass">
                <opcode>11111111</opcode>
                <opcode>10001000</opcode>
                <opcode>00000001</opcode>
              </instruction>
              <instruction id="extest">
                <opcode>00000000</opcode>
                <opcode>10000000</opcode>
              </instruction>
              <instruction id="highz">
                <opcode>00000110</opcode>
                <opcode>10000110</opcode>
               </instruction>
              <instruction id="clamp">
                <opcode>00000111</opcode>
                <opcode>10000111</opcode>
              </instruction>
          </device>
      </instruction_cells>

      <data_cells length="532">
          <device id="u_12" type="ti374" package="dw_package"
          length="532">
            <cell pos="64">
              <output lead="10" type="controlled" control="80"
              on="0" />
```

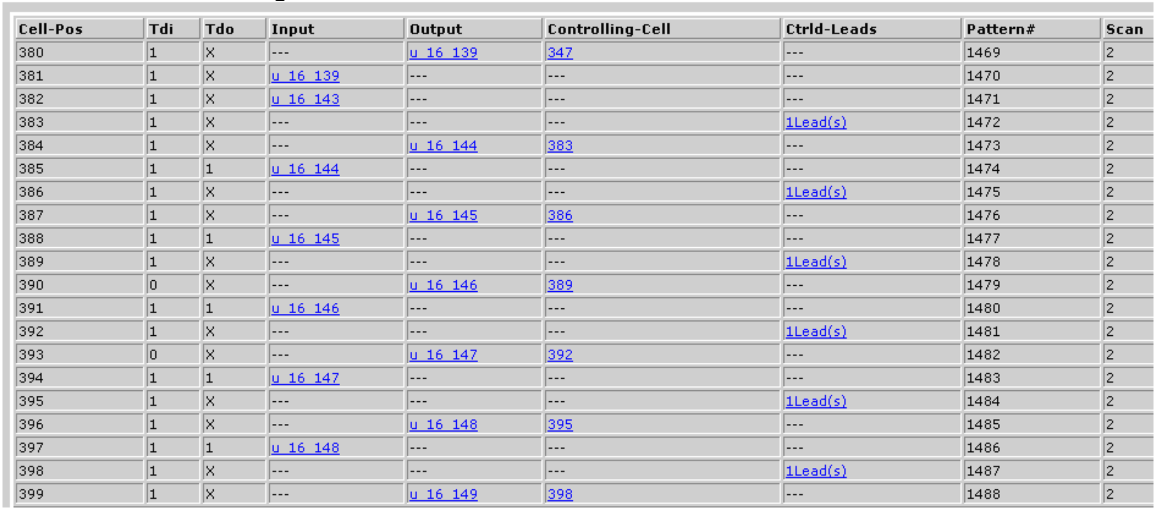
```
                        </cell>
                        <cell pos="72">
                            <input lead="15" type="simple" />
                        </cell>
                        <cell pos="80">
                            <input lead="24" type="simple" />
                            <control on="0">
                                <controlled_output lead="2" />
                                <controlled_output lead="3" />
                                <controlled_output lead="4" />
                                <controlled_output lead="5" />
                                <controlled_output lead="7" />
                                <controlled_output lead="8" />
                                <controlled_output lead="9" />
                                <controlled_output lead="10" />
                            </control>
                        </cell>
                    </device>
                /data_cells>
        </position_data>
    </boundary_scan_data>
```

Test pattern data is defined in the SerialVectors schema. Sub-elements with the tags iscan and dscan, identify data scanned into the instruction registers of all devices in the chain, and data scanned into and out of the data registers. The tags on sub-elements <tdi> and <tdo> represent data clocked into the UUT Test Data In port and the expected response from the UUT Test Data Out port, respectively. In the interest of saving space in the example below, the actual one’s and zero’s data at the <tdi> and <tdo> tags are only provided in the first scan vector.

```
    <SerialVectors>
    <major_comment>TAP reset sequence</major_comment>
        <iscan length="49">
            <tdi>1000100000001000000010000000100000010001001000000
            </tdi>
            <tdo>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            XXXXXXXX</tdo>
        </iscan>
        <dscan length="532">
            <tdi> 11100100100000111011001111110111…  (532 bits of data
            scanned in …</tdi>
            <tdo> XXXXXXXXXXXXXXXXXXXXX … (532 bits of data scanned
            out) …</tdo>
        </dscan>
        <iscan length="49">
```

```
            <tdi>000000000000000000000000000000000000000000000000
            </tdi>
            <tdo>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            XXXXXXXX</tdo>
        </iscan>
        <dscan length="532">
            <tdi>   (……. 532 bits of test data scanned in …..)    </tdi>
            <tdo>     (……..532 bits of  test data scanned out…)   </tdo>
        </dscan>
        <dscan length="532">
            <tdi>      (……. Next 532 bits of test data scanned in …..)    </tdi>
            <tdo>     (……..Next 532 bits of  test data scanned out…)   </tdo>
        </dscan>
    </SerialVectors>
```

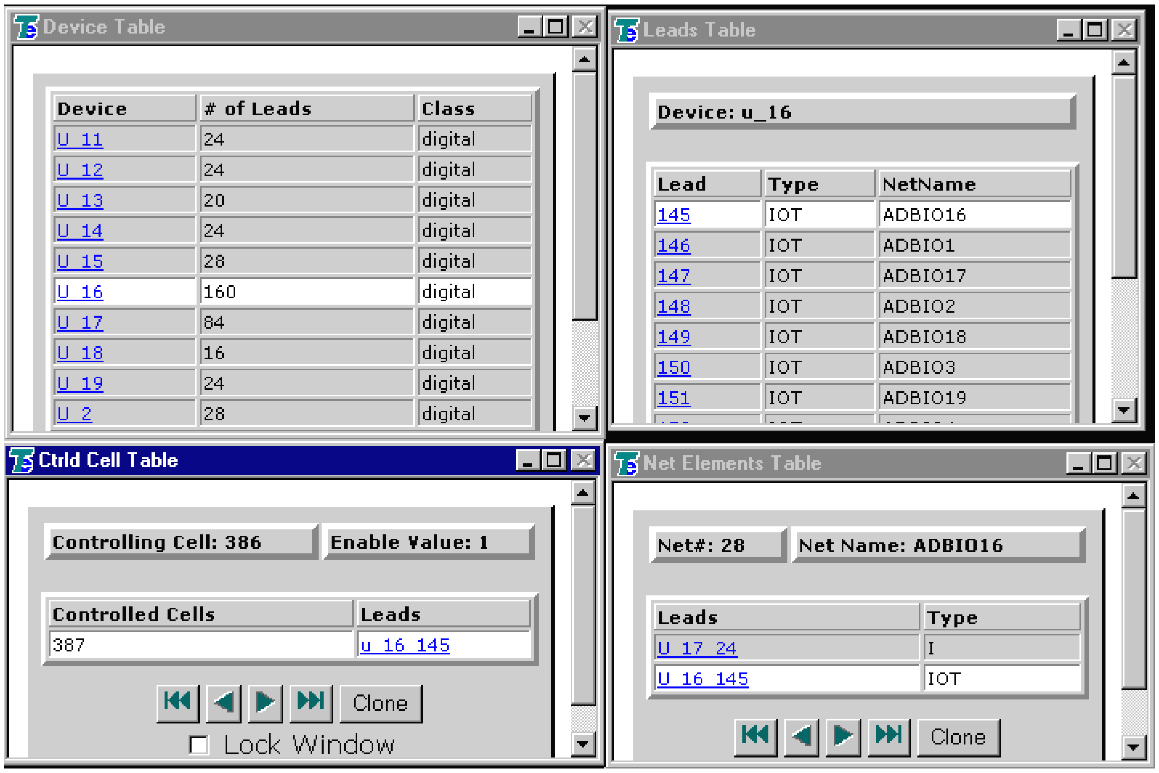
How can boundary scan test data represented in XML help in diagnostics? Consider a test where pattern 1477 in scan vector 2 has failed. Fig. 1 illustrates how we get more information to help diagnose this failure. The test pattern database can tell us the expected response at pattern 1477 in scan 2 is a logic 1. This pattern corresponds to boundary scan cell position 388. The position database tells us that the scan cell associated with bit position 388 is an input cell of a bi-directional lead, U_16_145, and that control cell 386 is associated with the scan cell register.



Cell-Pos	Tdi	Tdo	Input	Output	Controlling-Cell	Ctrld-Leads	Pattern#	Scan
380	1	X	---	u_16_139	347	---	1469	2
381	1	X	u_16_139	---	---	---	1470	2
382	1	X	u_16_143	---	---	---	1471	2
383	1	X	---	---	---	1Lead(s)	1472	2
384	1	X	---	u_16_144	383	---	1473	2
385	1	1	u_16_144	---	---	---	1474	2
386	1	X	---	---	---	1Lead(s)	1475	2
387	1	X	---	u_16_145	386	---	1476	2
388	1	1	u_16_145	---	---	---	1477	2
389	1	X	---	---	---	1Lead(s)	1478	2
390	0	X	---	u_16_146	389	---	1479	2
391	1	1	u_16_146	---	---	---	1480	2
392	1	X	---	---	---	1Lead(s)	1481	2
393	0	X	---	u_16_147	392	---	1482	2
394	1	1	u_16_147	---	---	---	1483	2
395	1	X	---	---	---	1Lead(s)	1484	2
396	1	X	---	u_16_148	395	---	1485	2
397	1	1	u_16_148	---	---	---	1486	2
398	1	X	---	---	---	1Lead(s)	1487	2
399	1	X	---	u_16_149	398	---	1488	2

Fig 1. Information from a boundary test database with XML tags displayed in a web browser window.

By clicking on the hyperlink U_16_145, we see in Fig. 2 that U_17_24, an input lead, is also on this net named ADBIO16. By clicking on the hyperlinks, we can follow the information paths for this lead to find out about the scan cells and register states associated with them. It is easy to see how these interrelationships can be revealed using XML tags and XSL stylesheets to display the data in a way that is clear and familiar for the end application, which in this case is a web browser window.



Device	# of Leads	Class
U_11	24	digital
U_12	24	digital
U_13	20	digital
U_14	24	digital
U_15	28	digital
U_16	160	digital
U_17	84	digital
U_18	16	digital
U_19	24	digital
U_2	28	digital

Device: u_16

Lead	Type	NetName
145	IOT	ADBIO16
146	IOT	ADBIO1
147	IOT	ADBIO17
148	IOT	ADBIO2
149	IOT	ADBIO18
150	IOT	ADBIO3
151	IOT	ADBIO19

Controlling Cell: 386 Enable Value: 1

Controlled Cells	Leads
387	u_16_145

Net#: 28 Net Name: ADBIO16

Leads	Type
U_17_24	I
U_16_145	IOT

Fig. 2 Interrelationships among device and lead characteristics with XML tags can be displayed in multiple web browser windows. XSL stylesheets support multi-way links.

Conclusion

The use of XML to tag key elements of data found in boundary scan test databases can aid significantly in the way information important to debugging boundary scan tests is made available to test engineers and operators. XML offers the advantage of expressing information and complex interrelationships among it in a human-readable, data-neutral format that is compatible with web browsers. More importantly it provides a means for expressing information in a non-proprietary format for use between different applications dependent on the same data.

References

1. XML: Structuring Data for the Web: An Introduction. Web Developer's Virtual Library, http://www.wdvl.com/Authoring/Languages/XML/Intro . May 1998

2. XML and the Second-Generation Web. Jon Bosak and Tim Bray. Scientific American, http://www.sciam.com/1999/0599issue/0599bosak.html . 1999

3. XML Schema Part 0: Primer. http://www.w3.org/TR/2000/CR-xmlschema-0-20001024 . October 2000.