BSML: AN APPLICATION OF XML TO ENHANCE BOUNDARY SCAN TEST DATA TRANSPORTABILITY

David E. Rolince, Teradyne Inc., (978) 370 1197, david.rolince@teradyne.com

ABSTRACT

The IEEE 1149.1 standard has successfully defined the rules of boundary scan design implementation. This has served as the catalyst for commercial companies to develop boundary scan test generation tools. Except for Serial Vector Format (SVF), an industry standard practice for expressing test vectors, each commercial offering uses proprietary formats for the data that is used to debug and diagnosis failures that are detected during test execution. This data tends to be in a form that is compatible only with a specific target test system. The introduction of XML (Extensible Markup Language) offers a solution for boundary scan test data that not only standardizes the expression of key information, but also enables this information to be retargeted easily through the use of XSL (Extensible Stylesheet Language) style sheets. The objective of this paper is to show how XML can be applied to boundary scan test data as a way to enable this data to be transported to applications and test systems that utilize this information.

Keywords: XML, XSL, Boundary Scan, browsers, test debug

1 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.

2 Requirements for Boundary Scan Test Data

The data required for generating a test of a UUT designed with boundary scan is very similar to that required for testing any other digital UUT. There needs to be a physical description of the UUT in sufficient detail to support the generation of test patterns and the diagnosis of failures detected by those patterns. Then there is the definition of the test patterns themselves.

Looking first at the physical design, there is the requirement for a description of all boundary scan devices on the UUT. This information is contained in the BSDL (Boundary Scan Description Language) model of each device. BSDL models are typically available directly from the device vendor, or can be obtained from synthesis tools in the case of custom ASICs. For boundary scan test generation and diagnostics, the required information from a BSDL model includes:

1. Identification of the numbers of boundary register cells in the device and the name(s) of the physical lead(s) associated with each cell,
2. Identification of any boundary scan output control cells and the register cells controlled by them,
3. The sequential order of boundary register cells in the device,
4. Identification of the boundary scan instructions supported by the device and the bit code that represents the instructions.

Boundary scan device chains are formed when the TDO lead of one boundary scan device is connected to the TDI lead of another. Chains can consist of several devices thereby forming a continuous path extending from the TDO lead of the first device to the TDI lead of the last device. A critical requirement is that the position of scan cells in the chain and their functionality be captured so that faulty circuit activity can be accurately diagnosed. One can see that a scan chain can be thousands of cells long for a UUT with several VLSI type devices supported by boundary scan

Likewise, the connectivity of device leads among scan and non-scan components in the UUT is important information, as well as the characteristics of those leads in terms of the direction of signal flow. This information includes which leads on a net are input, output, and bi-directional. For output leads, there may be control cells or leads that can force the output to a high Z state.

Test patterns serve two functions in a boundary scan test. Before any test can be applied, the instruction register of the boundary scan device must be programmed to put the device into the desired state, e.g., EXTEST, HIGHZ. Serial patterns representing the appropriate instruction must be clocked into the device's TDI port. Once the device is programmed, serial test data can be applied to the device's data registers. In addition to stimulus data, there is the requirement for expected response data, which is used to determine a go/no go condition, and provides valuable

information for pin level diagnosis. Boundary scan test patterns can be expressed in many formats, but SVF (Serial Vector Format) is the most prevalent in the industry.

3 Characteristics of Boundary Scan Test Data

The data generated and used by commercially available tools in the process of developing boundary scan tests generally satisfy the requirements described in the previous section. Unfortunately, the data tends to be written in proprietary formats that make it very difficult or impossible to use outside of the specific vendor's application. Generally the data can be grouped into three categories:

- 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 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. When these data are merged with the test pattern data set, the state of each boundary register cell for a given test vector is known. As a practical application, when data is exported to a graphical display utility, the relationship of test pattern data with physical leads and boundary scan data registers can be visualized. This 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. It is common to have pattern bursts consisting of 1 million bits of serial data. Keeping track of a stream of test patterns this deep 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.

4 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 data 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" based on the appropriate set of data requirements. The instance document defines the XML schema for the specific information in your application. It provides the rules that define the elements and structure of the new markup language, and serves as the guidelines for other developers to interface with the application.

For the example presented here, 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 we define from the specific data requirements we derived for our application. In this way XML allows the interrelationships of these schema to be expressed so that collections of related data could 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. An interesting sample of data requirements might look like this:

Schema circuit has elements

- Devices
- Nets
- TesterPins
- TopLevelScanPins
- ScanPaths

The element Devices, for example, is further broken down into individual Device instances each of which has the sub-element Leads. Device instances have attributes such as Name, Class, LeadCount. Sub-element Leads, in turn, has individual Lead instances associated with that device. Each Lead has an association with a Net and can have Properties attached to it. This relationship to the sub-element Net ties back to the second element in the list above and any sub-elements defined for it. If the Lead is a boundary scan lead, there is a scan cell associated with it and has Virtual I/O name and Control sub-elements.

Putting all this into the context of XML, the 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>
```

In the case of the device being a boundary scan device, there are the 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" />
                </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>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXX</tdo>
    </iscan>
    <dscan length="532">
        <tdi> 11100100100000111011001111101111…  (532 bits of data scanned in
        …</tdi>
        <tdo> XXXXXXXXXXXXXXXXXXXXX … (532 bits of data scanned out)
        …</tdo>
    </dscan>
    <iscan length="49">
        <tdi>0000000000000000000000000000000000000000000000000
        </tdi>
        <tdo>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXX</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>
```

5 A Practical Application

We now have a database of information derived from a set of data requirements and expressed in a highly transportable XML form. Let's look at an application that utilizes this work. As mentioned earlier, initial integration and debug of boundary scan tests can be a prodigious task. The application seeks to sort through the myriad of data generated from the boundary scan test generation tool, organize it in a meaningful way and display it in an intuitive form. It accomplishes this by using XSL stylesheets that extract specific information identified by its data tags from the boundary scan test database expressed in XML. The XSL stylesheets also define the way the data gets organized and displayed in a web browser.

So how can boundary scan test data represented in XML help in diagnosing these types of problems? 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

Figure 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.

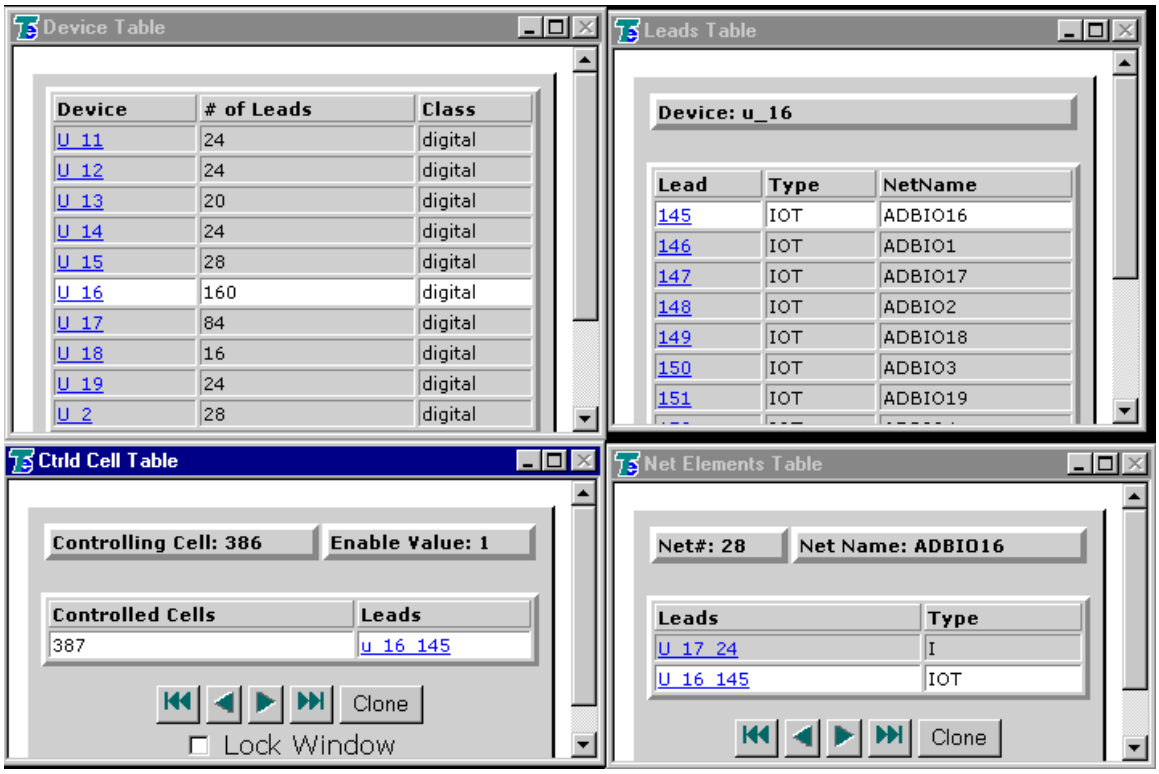

Figure 2. Interrelationships among device and lead characteristics with XML tags can be displayed in multiple web browser windows. XSL stylesheets support multi-way links.

6 CONCLUSION

Expressing large amounts of highly interrelated data in XML can significantly enhance its utility and transportability. It can also be used as a means for defining data requirements for building databases as well as extracting relevant information from them. 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.