XAPP476 (v1.1) June 19, 2005

Using BSDL Files for Spartan-3 Generation FPGAs

Summary

BSDL (Boundary Scan Description Language) files are provided for every part and package combination of IEEE 1149.1 (JTAG) compatible devices produced by Xilinx, including all the Spartan-3 Generation FPGAs. Third-party Boundary-Scan tools use these files to generate test vectors and perform the tests. Xilinx programming software also uses BSDL files when configuring devices through Boundary Scan.

This application note applies to all Spartan™-3 Generation FPGA families, which include the Spartan-3 family, the Spartan-3L family, and the Spartan-3E family.

Boundary-Scan Overview

Boundary-Scan testing is used to identify faulty board-level connections, such as unconnected or shorted pins. Boundary-Scan tests allow designers to quickly identify manufacturing or layout problems, which otherwise could be nearly impossible to isolate, especially with high-count ball-grid packages. More recently, PLD vendors such as Xilinx have made use of Boundary Scan as a convenient way of configuring devices, including the Spartan-3 Generation FPGA families.

IEEE Standards

JTAG (Joint Test Action Group) is the commonly used name for IEEE standard 1149.1, which defines a method for Boundary Scan. JTAG compliant devices have dedicated hardware that comprises a state machine and several registers to allow Boundary-Scan operations. This dedicated hardware interprets instructions and data provided by four dedicated signals: TDI (Test Data In), TDO (Test Data Out), TMS (Test Mode Select), and TCK (Test Clock). The JTAG hardware interprets instructions and data on the TDI and TMS signals, and drives data out on the TDO signal. The TCK signal is used to clock the process.

IEEE 1532 is a superset of the IEEE 1149.1 JTAG standard. IEEE 1532 provides additional flexibility for configuring programmable logic devices. IEEE Std 1532 enables designers to concurrently program multiple devices, minimize programming times with enhanced silicon features, and produce robust systems that are more easily maintained. This standard defines the three additional items required to configure in-system programmable logic devices:

- Device architectural components for configuration
- Algorithm description framework
- Configuration data file

General information on the IEEE 1532 JTAG standard is available on the Xilinx website at http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=isp_standards_specs#1532.

Boundary-Scan Tools

Boundary-Scan testing requires specialized test equipment and software. The Boundary-Scan test software is used to generate test vectors, which are typically delivered to the Boundary-Scan chain using a test pod connected to a PC.

To develop vectors for Boundary-Scan testing, the test software must be provided with information about the scan chain:

1. The composition of the scan chain - how many devices, what type, and so forth.

The chain composition can be either specified by the user or automatically detected by the Boundary-Scan software.

2. The Boundary-Scan architecture of each device - the Instruction Register length, opcodes, number of I/Os, and how each of those I/Os behaves.

   The Boundary-Scan architecture of each device is defined in a Boundary Scan Description Language (BSDL) file.

3. How the device I/Os are connected to each other.

   This information typically is extracted from a board-level netlist.

BSDL Files

Any manufacturer of a JTAG-compliant device must provide a BSDL file for that device. The BSDL file contains information on the function of each of the pins on the device - which are used as I/Os, which are power or ground, and so forth. All Xilinx BSDL files have file extensions of `.bsd`.

BSDL files for Xilinx devices are available in the development system and on the Xilinx website at http://www.xilinx.com/xlnx/xil_sw_updates_home.jsp. BSDL files for other manufacturers typically can be found on the manufacturer's website.

Files for the IEEE 1532 extension to the BSDL files are also available for Xilinx products. They are included with the other BSDL files.

IEEE 1149.1 BSDL files appear as: `<device_name>.bsd`

For example: `xc3s50.bsd`

These BSDL files are the only ones needed for programming. For JTAG testing, the package-specific files are used.

For example: `xc3s50_pq208.bsd`

IEEE 1532 BSDL files appear as: `<device_name>_1532.bsd`

For example: `xc3s50_pq208_1532.bsd`

Note that IEEE Std 1532 BSDL files should not be used in place of or alongside 1149.1 BSDL files.

BSDL File Composition

BSDL files describe the Boundary-Scan architecture of a JTAG-compliant device, and are written in VHDL. There are eight main parts of a BSDL file:

1. Entity Declaration

   The entity declaration is a VHDL construct used to identify the name of the device that is described by the BSDL file.

   Example (from the `xc3s50_pq208.bsd` file):

```
entity XC3S50_PQ208 is
```

2. Generic Parameter

   The Generic parameter specifies which package the BSDL file describes.

   Example (from the `xc3s50_pq208.bsd` file):

```
generic (PHYSICAL_PIN_MAP : string := "PQ208" );
```

3. Logical Port Description

   The Logical Port Description lists all of the pads on a device, and states whether that pin is an input (`in bit;`), output (`out bit;`), bidirectional (`inout bit;`) or unavailable for Boundary Scan (`linkage bit;`).

   Example (from the `xc3s50_pq208.bsd` file):

```
port (
  GND: linkage bit_vector (1 to 28);
  CCLK_P104: inout bit;
  DONE_P103: inout bit;
  HSWAP_EN_P206: in bit;
  M0_P55: in bit;
  M1_P54: in bit;
  M2_P56: in bit;
  PROG_B: in bit;
  TCK: in bit;
  TDI: in bit;
  TDO: out bit;
  TMS: in bit;
  VCCAUX: linkage bit_vector (1 to 8);
  VCCINT: linkage bit_vector (1 to 4);
  VCCO0: linkage bit_vector (1 to 2);
  IO_P2: inout bit; --  PAD124
  IO_P3: inout bit; --  PAD123
```

4. Package Pin Mapping

   The Package Pin Mapping shows how the pads on the device die are wired to the pins on the device package.

   Example (from the `xc3s50_pq208.bsd` file):

```
constant PQ208: PIN_MAP_STRING:=
  "GND:(P1,P8,P14,P25,P30,P41,P47,P53,P59,P66," &
    "P75,P82,P91,P99,P105,P112,P118,P129,P134,P145," &
    "P151,P157,P163,P170,P179,P186,P195,P202)," &
  "CCLK_P104:P104," &
  "DONE_P103:P103," &
  "HSWAP_EN_P206:P206," &
  "M0_P55:P55," &
  "M1_P54:P54," &
  "M2_P56:P56," &
  "PROG_B:P207," &
  "TCK:P159," &
  "TDI:P208," &
  "TDO:P158," &
  "TMS:P160," &
  "VCCAUX:(P17,P38,P69,P89,P121,P142,P173,P193)," &
  "VCCINT:(P70,P88,P174,P192)," &
  "VCCO0:(P188,P201)," &
  "IO_P2:P2," &
  "IO_P3:P3," &
```

5. use statements

   The use statement calls VHDL packages that contain attributes, types, constants, and others that are referenced in the BSDL File.

   Example (from the `xc3s50_pq208.bsd` file):

```
use STD_1149_1_1994.all;
```

6. Scan Port Identification

   The Scan Port Identification identifies the JTAG pins: TDI, TDO, TMS, TCK, and TRST (if used). TRST is an optional JTAG pin that is not used by Xilinx devices.

   Example (from the `xc3s50_pq208.bsd` file):

```
attribute TAP_SCAN_IN    of TDI : signal is true;
attribute TAP_SCAN_MODE  of TMS : signal is true;
attribute TAP_SCAN_OUT   of TDO : signal is true;
attribute TAP_SCAN_CLOCK of TCK : signal is (33.0e6, BOTH);
```

7. TAP description

   The TAP description provides additional information on the device's JTAG logic. Some of this information includes the Instruction Register length, Instruction Opcodes, and device IDCODE. These characteristics are device specific, and may vary widely from device to device.

   Examples (from the xc3s50_pq208.bsd file):

```
attribute COMPLIANCE_PATTERNS of XC3S50_PQ208 : entity is
      "(PROG_B) (1)";
attribute INSTRUCTION_LENGTH of XC3S50_PQ208 : entity is 6;
attribute INSTRUCTION_OPCODE of XC3S50_PQ208 : entity is
      "EXTEST    (000000)," &
attribute INSTRUCTION_CAPTURE of XC3S50_PQ208 : entity is
      "XXXX01";
attribute IDCODE_REGISTER of XC3S50_PQ208 : entity is
  "XXXX" &      -- version
  "0001010" &   -- family
  "000001100" &    -- array size
  "00001001001" &  -- manufacturer
  "1";          -- required by 1149.1
```

8. Boundary Register description

   The Boundary Register description gives the structure of the Boundary-Scan cells on the device. Each pin on a device may have up to three Boundary-Scan cells, each cell consisting of a register and a latch. Boundary-Scan test vectors are loaded into or scanned from these registers.

   Example (from the `xc3s50_pq208.bsd` file):

```
attribute BOUNDARY_REGISTER of XC3S50_PQ208 : entity is
    "   0 (BC_2, *, controlr, 1)," &
    "   1 (BC_2, IO_P161, output3, X, 0, 1, PULL0)," & --  PAD30
    "   2 (BC_2, IO_P161, input, X)," & --  PAD30
```

BSDL File Verification

Xilinx verification of the supplied BSDL files has two levels. Preliminary files are generated using an automated, Xilinx-standard, BSDL generation process. The process is script-based and extracts information directly from the device design files, which fully describe the architecture and pinout. The quality of "Preliminary" BSDL files is very high, and the syntax is always tested. Files marked "Final" have undergone all the tests for syntax and hardware verification. To guarantee that the BSDL files exactly describe the operation of each pin, Xilinx uses an independent third-party boundary-scan tool vendor — Intellitech — to verify the actual silicon against the BSDL.

Xilinx BSDL files are checked for 1149.1 compliance with the Intellitech Eclipse product using 'strict' BSDL syntax checking. Every semantic check described in the IEEE 1149.1b-1994 (the standard for BSDL syntax) is performed using strict parsing. Test patterns then are generated from the BSDL file that include unique tests for every I/O pin. Each Xilinx device/package combination is tested on the Intellitech RCT (Reduced Contact Tester). The test patterns

include verification of Test-Logic-Reset and TAP controller operation, BYPASS/IDCODE/USERCODE instructions and registers, and pin mapping of the boundary register to every input/output/bidir/clock pin and control cell. Finally, each device is tested for 1149.1 compliance after the device is programmed by downloading a design and using the RCT tester to verify post configuration compliance.

For more information on BSDL files and their verification, see http://www.xilinx.com/isp/bsdl/bsdl.htm.

Using BSDLAnno for Post-Configuration Boundary-Scan Behavior

Whenever possible, Boundary-Scan tests should be performed on an unconfigured Spartan-3 Generation device. Unconfigured devices allow for better test coverage, since all I/Os are available for bidirectional scan vectors. Boundary-Scan tests should be performed after configuration when configuration cannot be prevented and when differential signaling standards are used. If the differential signals are located between Xilinx devices, both devices can be tested pre-configuration. Each side of the differential pair will behave as a single-ended signal.

The BSDL files provided by Xilinx reflect the Boundary-Scan behavior of an unconfigured device. After configuration, the Boundary-Scan behavior of a device changes. I/O pins that were bidirectional before configuration may now be input-only, output-only, bidirectional, or unavailable. Boundary-Scan test vectors typically are derived from BSDL files, so if Boundary-Scan tests are going to be performed on a configured Xilinx device, the BSDL file must be modified to reflect the device's configured Boundary-Scan behavior.

The Boundary-Scan architecture changes after the device is configured because the Boundary-Scan registers sit behind the I/O buffer and sense amplifier. The hardware is arranged in this way so that the Boundary-Scan logic operates at the I/O standard specified by the design. This allows Boundary-Scan testing across the entire range of available I/O standards.

Because certain connections between the Boundary-Scan registers and pad may change, the Boundary-Scan architecture is effectively changed when the device is configured. These changes often need to be communicated to the Boundary-Scan tester through a post-configuration BSDL file. If the changes to the Boundary-Scan architecture are not reflected in the BSDL file, Boundary-Scan tests may fail.

Xilinx offers the BSDLAnno utility to automatically modify the BSDL file for post-configuration testing. BSDLAnno obtains the necessary design information from the routed `.ncd` file and generates a BSDL file that reflects the post-configuration Boundary-Scan architecture of the device.

Use the following syntax to generate a post-configuration BSDL file with BSDLAnno:

```
bsdlanno [options] infile[.ncd] outfile[.bsd]
```

The `infile` is the routed (post-PAR) NCD design source file for the specified design. The `outfile[.bsd]` is the destination for the design-specific BSDL file. The `.bsd` extension is optional. For more details on BSDLanno, see the Development System Reference Guide at http://toolbox.xilinx.com/docsan/xilinx7/books/data/docs/dev/dev0136_20.html.

Software Support

Xilinx offers several tools for generating device files and for device programming. Boundary-Scan test functionality is available from several third-party vendors, as noted at http://www.xilinx.com/products/design_resources/config_sol/resource/isp_ate.htm.

iMPACT

iMPACT is a full featured software tool used for configuration and programming of all Xilinx FPGAs, CPLDs, and PROMs. It features a series of "wizard" dialogs that easily guide the user

through the every step of the configuration process. iMPACT supports a host of output file types including SVF. iMPACT configuration software enables users to easily configure all Xilinx FPGAs using three different modes; slave serial, SelectMAP (Slave Parallel), and JTAG IEEE 1149.1. iMPACT supports the Parallel Cable IV and MultiPRO cables.

iMPACT features a special function in the JTAG mode to test both the operation of the cable and the robustness of the JTAG chain. The user can test chain operation by instructing iMPACT to write to and read back from the user code location multiple thousands of times. It then counts the number of errors that occur in this operation. This gives the user the opportunity to evaluate the relative robustness of the JTAG chain and the susceptibility to noise and other influences like board layout.

For more information on iMPACT see the iMPACT help at http://toolbox.xilinx.com/docsan/xilinx7/help/iseguide/mergedProjects/impact/impact.htm.

SVF Files

Serial Vector Format (SVF) is an industry-standard file format that is used to describe JTAG chain operations in a compact, portable fashion. SVF files capture all of the device specific programming information within the SVF instructions. SVF files are useful because intricate knowledge of the device is not needed. The capability to create SVF files is included in the iMPACT tool. For more information on SVF files see XAPP503 at http://www.xilinx.com/bvdocs/appnotes/xapp503.pdf.

J Drive Engine for IEEE 1532 Programming

Xilinx has developed and introduced the world's first IEEE Std 1532 Programming Engine: J Drive™ Engine. Using this engine and a simple cable connected to the parallel port of any PC, users can easily configure Xilinx IEEE Std 1532 compatible PLDs. The designer provides J Drive Engine with the data and 1532 BSDL files for the device(s) to be programmed using a command line interface to configure the PLDs in the JTAG chain. For more information, see http://www.xilinx.com/xlnx/xil_prodcat_systemsolution.jsp?title=isp_jdrivemain_page and XAPP500 at http://www.xilinx.com/bvdocs/appnotes/xapp500.pdf.

Using the BSCAN_SPARTAN3 Macro

BSCAN_SPARTAN3 provides access to the BSCAN sites on a Spartan-3 Generation device. It is used to create internal Boundary-Scan chains. The four-pin JTAG interface (TDI, TDO, TCK, and TMS) contains dedicated pins in Spartan-3 Generation FPGAs. To use normal JTAG for Boundary-Scan purposes, just hook up the JTAG pins to the port and go. The pins on the BSCAN_SPARTAN3 symbol do not need to be connected unless those special functions are needed to drive an internal scan chain.

Spartan-3 Generation FPGAs provide hooks for two user-definable scan chains through the USER1 and USER2 instructions. These instructions may be used to provide access to the user design through the JTAG interface. To take advantage of the optional USER1 and USER2 instructions, the designer must instantiate the BSCAN_SPARTAN3 macro in the source code, and wire it to the user-defined scan chain.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/10/03	1.0	Initial Xilinx release.
06/19/05	1.1	Included all Spartan-3 Generation families, including Spartan-3E and Spartan-3L devices. Made additional clarifications.