XAPP067 (v2.0) May 13, 2002

Using Serial Vector Format Files to Program XC9500/XL/XV Devices In-System

Summary

This application note describes how to program XC9500™/XL/XV devices in-system, using standard Serial Vector Format (SVF) stimulus files.

Introduction

XC9500/XL/XV devices use a standard 4-wire Test Access Port (TAP) for both In-System Programming (ISP) and IEEE 1149.1 boundary scan (JTAG) testing. Therefore, manufacturers can reduce their overall system cost and reduce device damage due to unnecessary handling by using automatic test equipment (ATE) or Boundary Scan based tools development to both program and test these devices. The XC9500/XL/XV Boundary Scan architecture is shown in Figure 1.

The Xilinx iMPACT software helps make this possible by automatically generating a Serial Vector Format (SVF) file describing the programming and test algorithms required by the XC9500/XL/XV devices. Most ATE platforms and Boundary Scan based development tools accept SVF as a test vector input format. This application note describes the steps required to generate an SVF file and how to use this file to program and test a device.

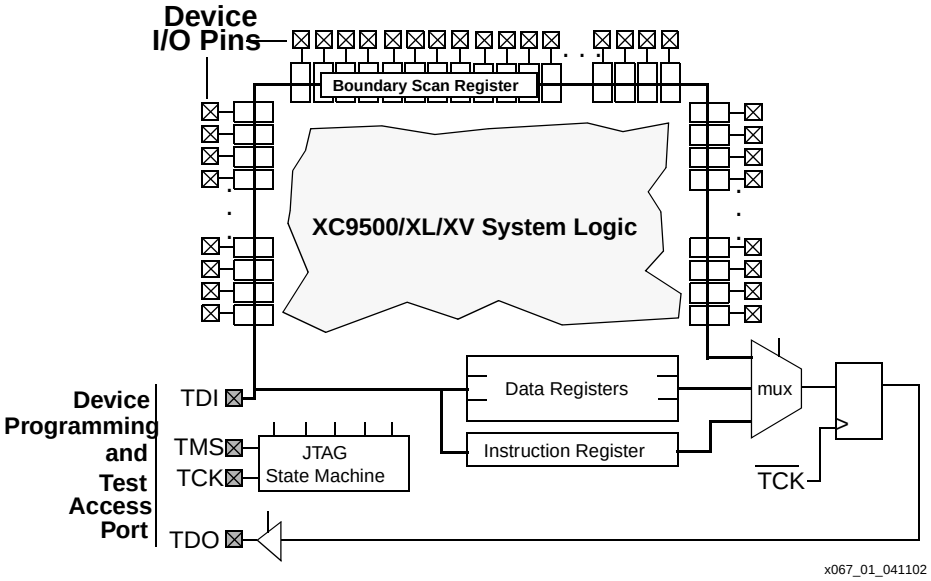

Figure 1: **XC9500/XL/XV Boundary Scan Architecture**

SVF Overview

The original Serial Vector Format was developed jointly by Texas Instruments and Teradyne in response to a need for the exchange of Boundary Scan test vectors between such tools as test generation software and ATE. At that time, usage of the IEEE standard 1149.1 was increasing but no common format or language existed to satisfy the need for a common data exchange.

The developers of SVF chose a format that did not use test vectors solely to provide TCK (clock) and TMS (mode control) signals to the IEEE 1149.1 TAP. Instead, the underlying models of the SVF format assume that all operations begin and end in stable states. This results in a much simpler and more concise description of the stimulus vectors.

Between mid-1991 and the autumn of 1994 three revisions of SVF were developed, with the goal of creating a format that was independent of the test application vehicle. By late 1994 over 100 companies had developed SVF-based tools and at least ten vendors of CAE tools and ATE were supporting SVF. The SVF was revised twice by 1999. However, the Xilinx iMPACT software does not take advantage of SVF enhancements since the 1994 Revision C.

SVF has proven itself to be a useful and reliable format for exchanging data between the Boundary Scan TAP and the software that drives it.

SVF Specification

For the purposes of XC9500/XL/XV ISP, only seven records of the thirteen SVF records that describe the standard are needed. Those seven records are discussed in this section.

An SVF file contains a set of ASCII statements. The maximum number of characters allowed on a line is 256, however one SVF statement can span more than one line. Each statement consists of a command and its associated parameters, terminated by a semicolon. SVF is not case sensitive and comments are indicated by an exclamation point (!) or a pair of slashes (//) at the beginning of a line, terminated by a carriage return.

Scan data within a statement is expressed in hexadecimal and is always enclosed in parenthesis. The scan data cannot specify a data string that is larger than the specified bit length; the Most Significant Bit (MSB) zeros in the hex string are not considered when determining the string length. The bit order for scan data defines the LSB (rightmost bit) as the first bit scanned into the device for scan data specified by the TDI and SMASK keywords, and is the first bit scanned out for data specified by the TDO and MASK keywords.

The following SVF Commands are supported by the XC9500/XL/XV iMPACT software:

- SDR (Scan Data Register).
- SIR (Scan Instruction Register).
- RUNTEST.
- HDR
- TDR
- HIR
- TIR

In each of the following command descriptions the parameters are mandatory. Optional parameters are enclosed in brackets ([ ]). Variables are shown in *italics*. Parenthesis “( )”are used to indicate hexadecimal values.

A scan operation is defined as the execution of an SIR or SDR command and any associated header or trailer commands.

SDR, SIR

```
SDR length [TDI (tdi)] [TDO (TDO)] [MASK (msk)] [SMASK (smask)][PIO (pio)];

SIR length [TDI (tdi)] [TDO (TDO)] [MASK (msk)] [SMASK (smask)][PIO (pio)];
```

These commands specify a scan pattern to be applied to the target scan registers. The SDR command (Scan Data Register) specifies a data pattern to be scanned into the target device Data Register. The SIR command (Scan Instruction Register) specifies a data pattern to be scanned into the target device Instruction Register.

Parameters:

length — A 32-bit decimal integer specifying the number of bits to be scanned.

[TDI (*tdi*)] — (optional) This specifies the value to be scanned into the target, expressed as a hex value. If this parameter is not present, the value of TDI to be scanned into the target device will be the TDI value specified in the previous SDR/SIR statement. If a new scan command is specified, which changes the length of the data pattern with respect to a previous scan, the TDI parameter must be specified, otherwise the default TDI pattern is undetermined and is an error.

[TDO (*tdo*)] — (optional) This specifies the test values to be compared against the actual values scanned out of the target device, expressed as a hex string. If this parameter is not present, no comparison will be performed. If no TDO parameter is present, the MASK will not be used.

[MASK (*mask*)] — (optional) This specifies the mask to be used when comparing TDO values against the actual values scanned out of the target device, expressed as a hex string. A “0” in a specific bit position indicates a “don’t care” for that position. If this parameter is not present, the mask will equal the previously specified MASK value specified for the SIR/SDR statement. If a new scan command is specified which changes the length of the data pattern with respect to a previous scan, the MASK parameter must be specified, otherwise the default MASK pattern is undefined and is an error. If no TDO parameter is present, the MASK will not be used.

[SMASK (*smask*)] — (optional) This specifies which TDI data is “don’t care”, expressed as a hex string. A “0” in a specific bit position indicates that the TDI data in that bit position is a “don’t care”. If this parameter is not present, the mask will equal the previously specified SMASK value specified for the SDR/SIR statement. If a new scan command is specified which changes the length of the data pattern with respect to a previous scan, the SMASK parameter must be specified, otherwise the default SMASK pattern used is undefined and is an error. The SMASK will be used even if the TDI parameter is not present.

Example:

```
SDR 24 TDI (000010) SMASK (0) TDO (818181) MASK (FFFFFF);

SIR 16 TDO (ABCD);
```

RUNTEST

RUNTEST *run_count* TCK;

This command forces the target IEEE 1149.1 bus to the Run- Test/Idle state for a specific number of TCK clock periods. This can be used to specify latency periods when operating the TAP.

Parameters:

run_count — The number of TCK clock periods that the 1149.1 bus will remain in the Run Test/Idle state, expressed as a 32 bit unsigned number.

Header and trailer bits for the scan operations are pre-defined with the HDR, TDR, HIR, and TIR commands. The HDR (header data register) command specifies the bit pattern to shift prior to the SDR-specified bit pattern. The TDR (trailer data register) command specifies the bit pattern to shift after the SDR-specified bit pattern. The HIR (header instruction register) command specifies the bit pattern to shift prior to the SIR-specified bit pattern. The parameters for the HDR, TDR, HIR, and TIR commands are the same as for the SDR and SIR commands.

Example:

RUNTEST 1000 TCK;

A Sample SVF File is shown in Figure 2.

```
! Begin Test Program

HIR 0;
TIR 0;
HDR 0;
TDR 0;
SIR 8 TDI (FE) MASK (FF)
SDR 14 TDI (3afe) MASK (3ff) TDO (0003)
    SMASK (3ff)
RUNTEST 100 TCK
!End test program
```

Figure 2: **Sample SVF File**

Using Xilinx iMPACT Software to Generate an SVF File

The Xilinx Foundation ISE, Alliance, and WebPACK software packages include the iMPACT download software. The iMPACT software can create SVF files to program Xilinx XC9500/XL/XV devices in a JTAG scan chain. See the *iMPACT User Guide* for instructions on creating SVF files. For the XC9500/XL/XV devices, the SVF must be created with the erase, program, and, optionally, verify operations.

SVF Interpretation

The simplicity of SVF is also one of its major weaknesses. Much of the behavior of SVF, while running, is left unspecified by the standard. In order to optimize SVF stimulus for an application, the interpretations of some operations must be defined more precisely.

SDR and SIR Bit Patterns

In the iMPACT SVF, the SDR and SIR commands specify data and instruction bit patterns for only the target device(s) in the JTAG scan chain. That is, SDR and SIR specify data and instructions for only the device(s) targeted for programming.

HDR, TDR, HIR, and TIR Bit Patterns

In the iMPACT SVF, the HIR and TIR commands specify BYPASS (or HIGHZ) instruction bits for the non-target device(s). The HDR and TDR commands specify bits to shift through the BYPASS registers of non-target devices.

RUNTEST TCK

For the XC9500/XL/XV devices, the RUNTEST command specifies a time to wait in the JTAG Run-Test/Idle state while an ISP operation is performed inside the device.

Many ATE Boundary Scan tool manufacturers prefer not to generate bursts of TCK activity because this results in significantly increased test vector file sizes. This increases the overall test cost and can cause the vector set to run inefficiently.

Because the iMPACT SVF is based on Revision C of the SVF specification, the RUNTEST command specifies a number of TCK cycles to execute. However, only the amount of time spent in the Run-Test/Idle state is important to the XC9500/XL/XV devices. The XC9500/XL/XV devices ignore the actual TCK clock pulses while in the Run-Test/Idle state. For the XC9500/XL/XV devices, the specified number of TCK cycles in the RUNTEST command must be interpreted as a **wait time in microseconds**.

SDR Predicted TDO Values

The SVF specification describes a method for specifying predicted TDO values. It does not, however, specify actions to be taken when the predicted TDO value does not equal the expected values.

When using Xilinx XC9500/XL/XV parts, the TDO values predicted reflect the status of the just completed operation (which could be an erase or a program operation). If the status is not the success status (which is the value predicted as the TDO value in the generated SVF file), then the following 1149.1 TAP controller state transition sequence must be followed to correctly erase and program the XC9500/XL/XV device (assuming the TDO validation failure is detected in the EXIT1-DR state):

1. EXIT1-DR
2. PAUSE-DR
3. EXIT2-DR
4. SHIFT-DR
5. EXIT1-DR
6. UPDATE-DR
7. RUN-TEST/IDLE

The above state transition sequence is illustrated in the 1149.1 TAP state diagram in Figure 3.

The net effect of the state transition sequence is to nullify the just-shifted-in programming or erase data and re-apply the previous program or erase data. Note that the application interpreting the SVF must acknowledge this by not advancing beyond the current SVF record.

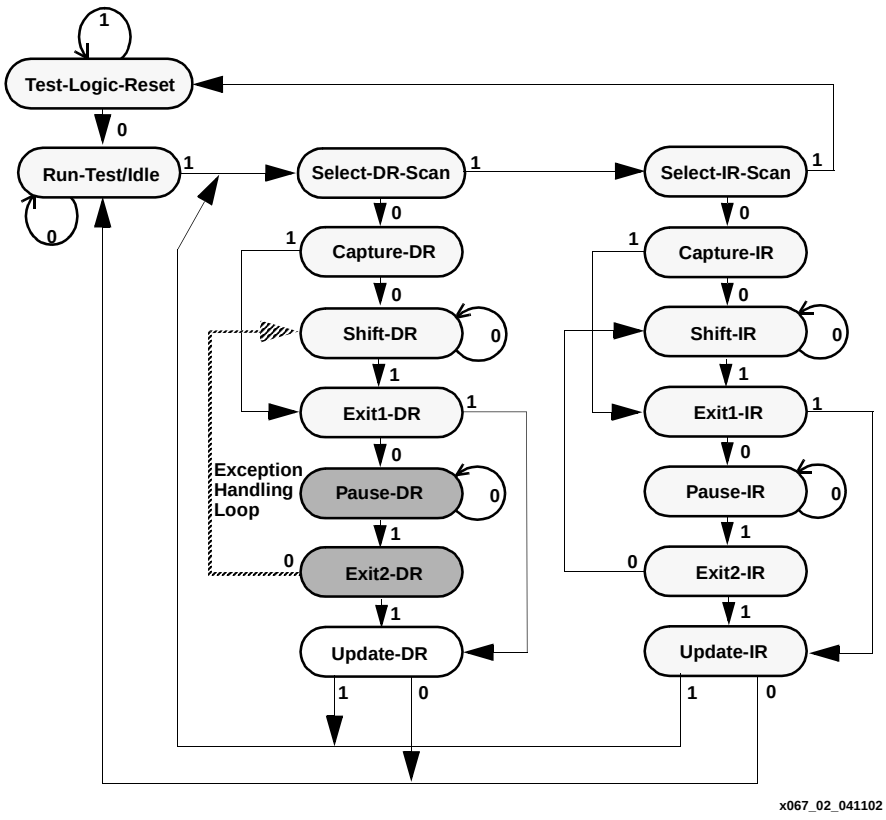

Figure 3: **Test Access Port State Diagram**

Figure 4 shows an SVF file fragment illustrating ATE flow.

```
// First SDR record
SDR 27 TDI (000003fe) SMASK (07ffffff); // Just apply the value - no test for
TDO
RUNTEST 160000 TCK; // Wait for 160 msec.
// Second SDR record
SDR 27 TDI (008003fe) SMASK (07ffffff) TDO (00000003) MASK (00000003);
// Apply value to TDI read TDO test for concurrence
// if not as expected do "failure recovery loop" - hold
// at this SDR instruction.
RUNTEST 160000 TCK; // Wait for 160 msec
// Third SDR record
SDR 27 TDI (010003fe) SMASK (07ffffff) TDO (00000003) MASK (00000003);
RUNTEST 160000 TCK; // Wait for 160 msec
// Fourth SDR record
SDR 27 TDI (018003fe) SMASK (07ffffff) TDO (00000003) MASK (00000003);
RUNTEST 160000 TCK; // Wait for 160 msec
```

Figure 4: **SVF File Fragment Illustrating ATE Flow**

Using the SVF file as an example, as shown in Figure 4, the required operation should then be as follows:

1. When reading an SDR instruction with a TDO specified (like the second one in Figure 4), the predicted TDO value must match the value output from the device on the tester. If it does not match, then the failure recovery loop is executed. In the RUN-TEST/IDLE state a pause is inserted for the amount of time specified in the previously applied RUNTEST instruction.
2. On exit from the RUNTEST instruction, re-apply that same SDR record (in this case the second one in the file) and test the TDO value again.
3. If the TDO matches the expected value, the TAP state machine is transitioned back to RUN-TEST/IDLE the normal way (via EXIT1-DR and UPDATE-DR) and is applied to the next SDR record.
4. This "recovery loop" is to be attempted no more than 32 times. If the TDO value does not match after 32 times, the part is considered defective and the process should abort with some failure indication supplied to the user.

Normally, less than 1% of the addresses fail the TDO check and require the additional erase or program time associated with execution of the failure recovery loop.

Pseudo-Code Algorithm for SVF-Based ISP

The following pseudo-code describes the sequence of operations that should be used in interpreting the SVF file on a generic SVF processor (ATE or Boundary Scan development tool).

1. Go to Test-Logic-Reset state
2. Go to Run-Test Idle state
3. Read SVF record
4. if SIR record then
   go to Shift-IR state
   Scan in <TDI value>
5. else if SDR record then
   set <repeat count> to 0
   store <TDI value> as <current TDI value>
   store <TDO value> as <current TDO value>

6. go to Shift-DR state
   scan in <current TDI value>
   if <current TDO value> is specified then

   if <current TDO value> does not equal <actual TDO value> then

   if <repeat count> > 32 then

   LOG ERROR

   go to Run-Test Idle state

   go to Step 3

   end if

   go to Pause-DR

   go to Exit2-DR

   go to Shift-DR

   go to Exit1-DR

   go to Update-DR

   go to Run-Test/Idle

   increment <repeat count> by 1

   pause <current pause time> microseconds

   go to Step 6)

   end if

   else
   go to Run-Test Idle state

   go to Step 3

   endif

7. else if RUNTEST record then
   pause tester for <TCK value> microseconds
   store <TCK value> as <current pause time>
   end if

Conclusion

By using the iMPACT-generated SVF files it is possible to streamline manufacturing flows by programming XC9500/XL/XV parts on automatic test equipment and third party Boundary Scan tools. This allows integration of the program and test steps of the system manufacturing process. This integration will result in higher system yields, better manufacturability, and simpler part inventory management.

Xilinx and/or its automatic test equipment partners provide support utilities to implement this SVF solution on the test equipment.

References

Serial Vector Format Specification, Rev E., Texas Instruments and Asset-Intertech. (**http://www.asset-intertech.com/support/svf.pdf**)

The Boundary Scan Handbook, Kenneth Parker, Klewer Academic Publishers, 1994.

IEEE Standard Test Access Port and Boundary Scan Architecture, IEEE Std 1149.1-1990 (including IEEE Std 1149.1a-1993)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/01/97	1.0	Initial Xilinx release.
05/13/02	2.0	Revised.