Programming Xilinx XC9500 on a Teradyne Z1800

EZTag Version

June 16, 1997 (Version 1.0) *Printed in U.S.A.*

April 10, 1997

Preface

About This Manual

This manual describes how to program Xilinx XC9500 CPLDs on Teradyne Z1800 testers.

Before using this manual, you should be familiar with the operations that are common to all Xilinx's software tools: how to bring up the system, select a tool for use, specify operations, and manage design data.

Manual Contents

This manual covers the following topics.

- Chapter 1, "Introduction," lays out the basic procedure for programming an XC9500 CPLD in a Teradyne test environment.
- Chapter 2, "Creating SVF Files," discusses how to create SVF files using EZTag on PCs, and on Sun and HP workstations.
- Chapter 3, "Creating Teradyne Test Files," discusses how to generate a Teradyne Binary Vector File, and how to create the executable **`ptprog.exe`**, and integrate into your test environment.
- Appendix A, "Troubleshooting," contains information on troubleshooting a problem.

Conventions

In this manual the following conventions are used for syntax clarification and command line entries.

- `Courier font` indicates messages, prompts, and program files that the system displays, as shown in the following example.

  ```
  speed grade: -100
  ```

- **`Courier bold`** indicates literal commands that you must enter in a syntax statement.

  `rpt_del_net=`

- *Italic font* indicates variables in a syntax statement. See also, other conventions used on the following page.

  `xdelay` *design*

- Square brackets "[ ]" indicate an optional entry or parameter. However, in bus specifications, such as bus [7:0], they are required.

  `xdelay [`*option*`]` *design*

- Braces "{ }" enclose a list of items from which you choose one or more.

  `xnfprep` *designname* **`ignore_rlocs={true|false}`**

- A vertical bar " | " separates items in a list of choices.

  `symbol` *editor* `[bus|pins]`

Other conventions used in this manual include the following.

- *Italic font* indicates references to manuals, as shown in the following example.

  See the *Development System Reference Guide* for more information.

- *Italic font* indicates emphasis in body text.

  If a wire is drawn so that it overlaps the pin of a symbol, the two nets *are not* connected.

- A vertical ellipsis indicates repetitive material that has been omitted.

```
IOB #1: Name = QOUT'
IOB #2: Name = CLKIN'
.
.
.
```

- A horizontal ellipsis "..." indicates that the preceding can be repeated one or more times.

  `allow block` *`blockname loc1 loc2 ... locn`* `;`

Chapter 1

Introduction

This document describes the procedures necessary to program Xilinx XC9500 CPLD designs in a Teradyne test environment. The procedures described in this document lay out the necessary steps you need to perform; they are:

- Creating an SVF File Using EZTag
- Generating a Teradyne Binary Vector File
- Creating the **`ptprog.exe`** executable
- Modifying the **`PT2.INI`** File
- Integrate the executable and **`.img`** file into your test program

EZTag translates JEDEC files into Serial Vector Format (SVF) files. The **`xilinx`** bat file translates SVF files to Teradyne Binary Vector Files.

The following is needed to run this software:

- Digital Function Processor option
- Windows95 or WindowsNT, a 32 bit environment
- Turbo C++ or Microsoft Visual C++ Compiler (the Microsoft Visual C++ must be able to produce DOS executables to produce the 16 bit executable **`ptprog.exe`**)
- Teradyne F1 Software

Note: The installation procedure is found in the README file on the disk. This is an ASCII text file and can be viewed from the DOS editor (**`edit readme.txt`**) or the Windows Notepad.

Chapter 2 describes the procedure for creating an SVF file from EZTag from both PC and workstation environments. Chapter 3 describes how to produce Teradyne compatible tester programming.

Hardware Considerations

This software and methodology applies to the following Teradyne testers running Teradyne software release F1 or greater.

- Z1800 with Digital Functional Processor

Chapter 2

Creating SVF Files

Creating an SVF File Using EZTag

This procedure describes how to create an SVF file; it assumes that the Xilinx XACT-CPLD version 6.0.1 (or newer) or XABEL-CPLD version 6.1.1 (or newer) is being used, which includes the XC9500 fitter and the EZTag software.

Note: XABEL-CPLD runs on PCs only.

The directory that EZTag writes an SVF file to depends on the method from which EZTag was invoked. SVF files are written into directories as shown in Table 2-1.

Table 2-1 SVF File Locations

Product	Invoked from:	Files written to:
XABEL-CPLD	Tools Menus	Project Directory
XACT-CPLD	Icon in Program Group	\xact9000
XACT-CPLD	Icon in Design Manager	Work directory of open project

On a PC

If you are using a PC with Windows 3.1 or 95, use the following procedure.

1. Fit the design and create a JEDEC output file.

2. Double-click on the EZTag icon. The EZTag-JTAG Download Software will appear.

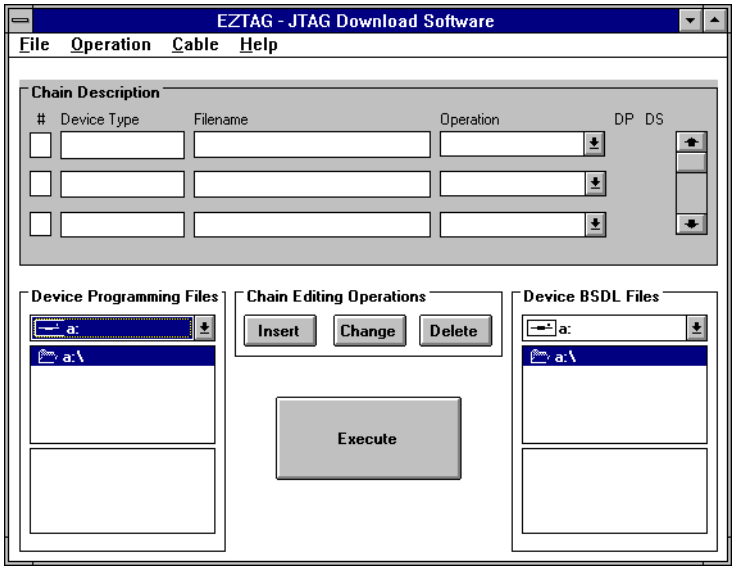

Figure 2-1 EZTag - JTAG Download Software

3. Select SVF as the execution mode.

   `Cable > SVF`

   EZTag is now set to create SVF files.

4. Assign device types for each device in the chain. For each device:

   a) Under **`Device Programming Files`** find the disk and directory containing the files you want to use.

   b) Double-click on the file name. The **`Device Type`** and **`File name`** should appear in the **`Chain Description`**.

   c) Click once on the **`Operation`** down arrow to select the type of operation you want to select. Your options are:

   - **`Erase`** generates an SVF file for the bit sequence needed to erase the specified part.
   - **`Verify`** generates an SVF file that specifies the bit sequence to read back the device contents and compare it against the contents of the specified JEDEC file.

- **Program** generates an SVF file that specifies the bit sequence to program the specified part from a JEDEC file named *designName*.jed.

5. Specify the BSDL files for all non-XC9500 parts as follows:

   a) Under **Device BSDL Files**, find the disk and directory containing the files you want to use.

   b) Double-click on the *filename* of the device. Type "OTHER." The selected *filename* should appear in the chain description.

   c) The operation will be set to BYPASS and cannot be changed.

Note: For any non-XC9500 part in the JTAG chain make certain that the BSDL file for the specified part is available along the XACT path and is called *deviceType*.bsd. The design name in this case can be any arbitrary name.

6. On a **PC** to execute click once on the **EXECUTE** softkey.

The program will generate an SVF file for each and every device in the chain.

Note: The SVF files will be named *designName*.svf, and will be created in the current working directory (the directory in which EZTAG is being run). Consecutive operations on the same *designName* file will overwrite the SVF file each time. The SVF file contains all data and commands necessary to perform the specified function.

On a Workstation

If you have a **Sun or HP workstation**, from the XACT command line use the following procedure:

1. Fit the design and create a JEDEC output file.
2. Invoke the EZTag software

   ```
   eztag -svf
   ```

   The following message appears:

   ```
   Xilinx (R) EZTAG XC9500-CPLD-6.0.3-JTAG Boundary-Scan
   Download

   Copyright (C) Xilinx Inc. 1991-1996. All Rights
   Reserved.

   ----------------------------------------------------
   ```

```
SVF GENERATION MODE.
EZTAG ?
```

3. Set up the device types and assign design names.

   On a **Sun or HP workstation**, type the following command at the **EZTAG ?** prompt:

   part *deviceType1*:*designName1*
   ** *deviceType2*:*designName2***
   ** *deviceTypeN*:*designNameN* <CR>**

   where ***deviceType*** is the name of the BSDL file for that device and ***designName*** is the name of the design to translate into SVF. Multiple *deviceType:designName* pairs are separated by spaces. For example:

   ```
   part xc95108:abc12 xc95216:wxyz
   ```

   This command defines the JTAG device chain, from one to any number of devices. The parts specified in the **part** command should be arranged in order beginning with the first device to receive TDI and ending with the last device to output TDO.

Note: For any non-XC9500 part in the JTAG chain make certain that the BSDL file for the specified part is available along the XACT path and is called *deviceType*.bsd. The design name in this case can be any arbitrary name.

4. Execute an EZTag operation. Your options are erase. verify, and program.

- **erase *designName*** — generates an SVF file for the bit sequence needed to erase the specified part.
- **verify *designName* [-j *jedecFileName*]** — generates an SVF file that specifies the bit sequence to read back the device contents and compare it against the contents of the specified JEDEC file.
- **program [-b] *designName* [-j *jedecFileName*]** — generates an SVF file that specifies the bit sequence to program the specified part from a JEDEC file named *designName*.jed (or alternately, the JEDEC file name specified after the "-j"). The program command option adds the following functionality:

  -b — Used to specify programming of a blank part. This is typically specified for parts delivered blank from the factory.

5. Exit EZTag

On a **Sun or HP workstation** you will exit EZTag by entering the following command:

```
quit
```

Note: The SVF file will be named *designName*.svf, and will be created in the current working directory (the directory in which EZTAG is being run). Consecutive operations on the same *designName* file will overwrite the SVF file each time. The SVF file contains all data and commands necessary to perform the specified function.

Software Restrictions

EZTAG can generate SVF files only for devices for which XC9500 JEDEC files can be created.

The current software can only generate SVF files for operations on one part at a time. If there are several parts to be programmed, additional program commands must be executed — one for each part, creating multiple SVF files. In each SVF file, one device will be programmed while the others are held in bypass mode.

Chapter 3

Creating Teradyne Test Files

Introduction

Use the SVF file that you generated above as input for the creation of a Teradyne Binary Vector File (**`.img`**). This chapter explains how to create executable Teradyne programs from SVF files.

You first need to check that your PC is setup with the following files that were contained in the zip file (setup file). If you do not find these files, your unzip utility probably failed.

Table 3-1 Directories and Files

Directory	Files
dfp	svf.c xilinx.c
doc	svfp2dts.pdf readme.txt
xlate	svfp.exe xilinx.bat bsv.dll dfpbv.dll failrep.dll svfc.dll svfp.dll util.dll wrpsrv.dll

The **`xlate`** directory contains the executable and libraries you will need to parse the SVF file output of your design tool. Add the **`xlate`** directory to the PATH variable on your PC.

Generating a Binary Vector File

To create a binary vector file, run the translator by entering the **xilinx** command.

```
xilinx filename.svf
```

Note: xilinx is a DOS **.bat** file that will invoke the **svfp** program with the following parameters:

-period 1000 -svf *filename*.**svf -target dfpbv**.

The **-period** parameter is used by the **svfp** program to determine the real time clock value used. The **-svf** parameter refers to the **.svf** file that contains your programming information, and the **-target** parameter indicates that the binary vector file will be output such that it can be run on the Digital Functional Processor.

In general, the file size for the **.img** file produced by the **svfp** program will be smaller than the original **.svf** file.

After running **xilinx.bat** copy the resulting **.img** file to the directory containing your data files (on the DFP).

Creating the Executable ptprog.exe

Source code for the Digital Functional Processor is located in the dfp directory. Compile **xilinx.c** into an executable called **ptprog.exe**.

Note: Teradyne currently supports Turbo C++ Version 3.0 and Microsoft Visual C++. You must have a version of Visual C++ that produces DOS 16 bit executables.

xilinx.c has the channel assignments for the JTAG connections hardcoded. If your design has multiple devices in the JTAG scan chain, you will need to hook the TDI of the first device to channel 209 and TDO of the last device in the chain to channel 208. You should have also generated your SVF file with the same chain configuration. The table below shows the mapping that the **xilinx.c** expects. The TRST is not needed for Xilinx parts.

Table 3-2 Mapping

Signal	DFP Card	DFP node/s
TDO	Card 0	208 (Port C bit 0)
TDI	Card 0	209 (Port C bit 1)
TCLK	Card 0	210 (Port C bit 2)
TMS	Card 0	211 (Port C bit 3)
TRST	Card 0	212 (Port C bit 4)

TCLK is running at the fastest rate the Z1800 DFP will allow (600 kHz).

The **`ptprog.exe`** when executed will look for two input files: the **`.img`** binary vector file, and the **`pt2.ini`** file (described below).

Modifying the pt2.ini File

The **`pt2.ini`** file provides additional information to the **`ptprog.exe`** program at run time. A sample **`pt2.ini`** file is shown below.

Sample pt2.ini - XC95108 :

L,IC1,XC95108,pgm.svf,,,0,1

R,format = No translation

where:

`L`	= local device tag	
`IC1`	= board identifier	Modify the board identifier to your own board identifier.
`XC95108`	= device type	Make this be the device type for the device that is being programmed in the SVF file.
`pgm.svf`	= data source file	Make this the name of the original SVF file. This will generate a **`.img`** file with the same root name as your **`.svf`** file (**`pgm`** in this example)

Blank	= format of data source file (91 = Jedec fuse file)	Not needed for Xilinx devices.
Blank	= number of fuses	Not needed for Xilinx devices.
0	= chain position	Not needed for Xilinx devices.
1	= fill character	Not needed for Xilinx devices.
R	= remarks / comments	

Adding Xilinx Programming to your Z1800 Board Test Program

1. Edit the PRGMVARS section of your program. Enable the DFP and specify the Source File Directory path, usually the board directory.
2. Create a subdirectory of the source file directory and copy the **`*.img`**, **`pt2.ini`** and **`ptprog.exe`** files to it.
3. Create a DFP worksheet and specify the subdirectory name in the source directory field of the worksheet. No arguments are needed.
4. Press the update button and you are ready to program.

Appendix A

Troubleshooting

ATE environments tend to be very noisy. The presence of electrical noise can contribute to erratic ISP behavior. Consider the following tips if you suspect noise problems.

If you are not able to program the Xilinx parts reliably, try turning down the TCLK rate. You can do this by modifying the following **`xilinx.c`** statements as follows:

Uncomment `#define CCC_IO_MAP`

Comment out `#define CCC_MEM_MAP`

This will reduce the TCK clock frequency to about 300 kHz.