

Verification Continuum™

ZeBu® Server

Site Administration Guide

Version Q-2020.03, March 2020



Copyright Notice and Proprietary Information

©2020 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

Preface.....	11
About This Book	11
Intended Audience.....	11
Contents of This Guide	12
Related Documentation	13
 1 Safety Recommendations	 15
 2 Overview of Administration Activities	 17
2.1 ZeBu Server Installation Process	18
2.2 Hardware and Software Requirements for the Host PC	19
2.2.1 License Software Compatibility	19
2.2.2 Synopsys Interoperable Technologies	20
2.2.3 Third-Party Tools	21
 3 Installing ZeBu Server Software	 23
3.1 Downloading Packages for Installation	24
3.1.1 Downloading and Extracting the Synopsys Installer.....	24
3.1.2 Downloading the ZeBu Software Package.....	25
3.2 Downloading Required Interoperable Technologies	27
3.3 Downloading Diagnostics Patches for ZeBu Server 3	28
3.3.1 Diagnostics Packages for ZeBu Server 3	28
3.3.2 Diagnostics for ZeBu Server 3	29
3.4 Diagnostics for ZeBu Server 4	32
3.5 Installing the ZeBu Server Software	32
3.5.1 Using Graphical Mode to Install ZeBu Server	33
3.5.2 Using Script Mode to Install ZeBu Server	35
3.6 Post Installation Activities	37
3.7 Installing Licenses	37
3.7.1 Installing ZeBu and Xilinx Licenses	38
3.7.2 Modifying License Files.....	39
3.7.3 Starting the License Server.....	40

4 Initializing the PCIe Boards	43
4.1 Overview of Initialization of PCIe Boards	44
4.2 Prerequisites of Initializing PCIe Boards	44
4.3 Initializing PCIe Boards on Host PC Running ZeBu 2017.03 or Later Releases	45
4.4 Initializing PCIe Boards on Host PC Running 2020.03 and Releases Earlier than 2017.03 (2015.09 or Earlier)	46
4.5 Initializing the PCIe Boards Automatically When Booting Linux	49
4.6 Precompiling the ZeBu Device Driver on Host PC	49
4.7 PCIe Boards Detection Order	50
4.8 Viewing ZeBu Server Installation Log Files	53
4.9 Updating PCIe Boards Firmware	55
5 Setting Up ZeBu Server	57
5.1 zSetupSystem Tool	58
5.2 Choosing the System Directory	59
5.2.1 One Directory for All PCs	59
5.2.2 Naming Rules	60
5.3 Preparing the Setup File (zini File)	60
5.3.1 Declaring the System Directory and Diagnostic Patches in the zini file	61
5.3.2 Managing Calibration History	61
5.3.3 Testing the Memories	61
5.3.4 Skipping Memory Tests	63
5.4 Launching the zSetupSystem Tool	63
5.5 Content of the System Directory	65
5.5.1 System Directory Files	66
5.5.2 Subdirectories in the System Directory	66
5.5.3 Information Log Files	67
5.5.4 ZeBu Runtime Log Files	75
6 Initializing the ZeBu Server System	79
7 Generating the Target Hardware Configuration File	81
7.1 Generating a Configuration File for One ZeBu Server System	82
7.2 Generating a Configuration File for Several ZeBu Server Systems ...	83
7.3 Generating a Hardware Configuration File for Relocation	84

7.4 Generating a Default Configuration File	85
7.4.1 Default Configuration File Using a List of Modules	85
7.4.2 Default Configuration File Using an FPGA Count	87
7.4.3 Default Configuration File for a Specific Interconnection Topology	88
7.5 Using an Example Configuration File	89
8 Configuring the End-User Environment	91
8.1 Setting Mandatory Environment Variables for ZeBu Software	91
8.2 Activating a ZeBu and Xilinx License	92
8.3 Declaring the System Directory for Emulation Runtime	93
8.4 Enabling ZeBu Server 4 Bitstream Cache	93
8.5 Runtime Settings for Relocation	94
8.5.1 Runtime Settings for Relocation on ZeBu Server 3	94
8.5.2 Runtime Settings for Relocation With a Multiunit or Multi-PCIe System	96
8.5.3 Runtime Settings for Relocation on ZeBu Server 4	96
8.5.4 Runtime Settings for Speed Adapters	97
8.5.5 Checking the Position of a ZeBu System	98
9 Troubleshooting ZeBu Server Installation	99
9.1 Missing Kernel Source Files for the Linux Kernel	100
9.2 ZeBu Server Unit is Not Detected by the Host PC	102
9.3 ZeBu Kernel Module Version Mismatch	102
9.4 ZeBu Software Does Not Recognize the Hardware	103
9.5 zInstall Accidentally Called	103
9.6 zServer or zUtils is Running While Trying to Install a Driver	104
9.7 ZeBu Driver is Not Installed (Versions before L-2016.06)	105
9.8 Problems When Compiling With gcc	105
9.9 Cannot Communicate With Units When a PC is OFF	107
9.10 Runtime Errors Caused by Faulty Memories	108
9.11 Synopsys Installer Does Not Install Packages	111
10 Appendix A: Examples of setup_template.zini	113
11 Appendix B: Getting Board Labels	123

List of Figures

Step 1: ZeBu Server Installation Process 23

**Interpreting Diagnostics Patch Name for a 2-Slot, Single Unit ZeBu Server
3 System 29**

**Interpreting Diagnostics Patch Name for a 5-Slot, Single Unit ZeBu Server
3 System 30**

**Example: Interpretation of Diagnostics Patch for a Single Unit ZeBu Server
3 System 32**

Step 2: ZeBu Server Installation Process 43

Step 3: ZeBu Server Installation Process 57

Step 4: ZeBu Server Installation Process 79

Step 5: ZeBu Server Installation Process 81

Step 5: ZeBu Server Installation Process 91

List of Tables

Synopsys Interoperable Technologies..... 20

Tested Third-party Tools 21

Files in the System Directory 66

Subdirectories in the System Directory 66

ZeBu Server 3 Syntax Rules 86

ZeBu Server 4 Syntax Rules 86

Available Example Configuration Files..... 90



About This Book

This guide contains information on administration tasks for ZeBu Server 3 and ZeBu Server 4. It describes the procedures for software installation and information on troubleshooting. Before proceeding with the installation, make sure you read the safety recommendations outlined in [Safety Recommendations](#).

The **ZeBu Server 3 Site Planning Guide** and **ZeBu Server 4 Site Planning Guide** contains information on ZeBu hardware installation and host PC requirements.

Note

In this guide, the term "ZeBu Server system" refers to a single unit or multiunit configuration for ZeBu Server 3 and ZeBu Server 4.

Intended Audience

This guide is written for experienced EDA hardware and software engineers to help them administrate their ZeBu Server unit. These engineers should have experience working with the Linux operating system.

Contents of This Guide

This guide has the following chapters:

Chapter	Describes...
Safety Recommendations	The safety-related recommendations, which you must read before proceeding with software installation
Overview of Administration Activities	Steps for ZeBu installation and initialization
Installing ZeBu Server Software	Procedure for installing the ZeBu software
Initializing the PCIe Boards	Initialization of PCIe boards on host PC
Setting Up ZeBu Server	The <code>zSetupSystem</code> tool for completing the ZeBu Server setup process
Initializing the ZeBu Server System	Initialization of ZeBu Server with the <code>zSetupSystem</code> tool
Generating the Target Hardware Configuration File	Target hardware configuration files for ZeBu Server systems and for relocation
Configuring the End-User Environment	Operations for configuring end-user environment, such as defining mandatory environment variables.
Troubleshooting ZeBu Server Installation	Troubleshooting scenarios and workaround for each scenario
Appendix A: Examples of <code>setup_template.zini</code>	The <code>setup_template.zini</code> file for each ZeBu Server
Appendix B: Getting Board Labels	Board labels of ZeBu Server

Related Documentation

Document Name	Description
<i>ZeBu Server 4 Site Planning Guide</i>	Describes planning for ZeBu Server 4 hardware installation.
<i>ZeBu Server 3 Site Planning Guide</i>	Describes planning for ZeBu Server 3 hardware installation.
<i>ZeBu Server Site Administration Guide</i>	Provides information on administration tasks for ZeBu Server 3 and ZeBu Server 4. It includes software installation.
<i>ZeBu Server Getting Started Guide</i>	Provides brief information on using ZeBu Server.
<i>ZeBu Server User Guide</i>	Provides detailed information on using ZeBu Server.
<i>ZeBu Server Debug Guide</i>	Provides information on tools you can use for debugging.
<i>ZeBu Server Debug Methodology Guide</i>	Provides debug methodologies that you can use for debugging.
<i>ZeBu Server Unified Command-Line User Guide</i>	Provides the usage of Unified Command-Line Interface (UCLI) for debugging your design.
<i>ZeBu Server Functional Coverage User Guide</i>	<p>Describes collecting functional coverage in emulation.</p> <p>For VCS and Verdi, see the following:</p> <ul style="list-style-type: none">- Coverage Technology User Guide- Coverage Technology Reference Guide- Verification Planner User Guide- Verdi Coverage User Guide and Tutorial <p>For SystemVerilog, see the following:</p> <ul style="list-style-type: none">- SystemVerilog LRM (2017)
<i>ZeBu Server Power Estimation User Guide</i>	<p>Provides the power estimation flow and the tools required to estimate the power on a System on a Chip (SoC) in emulation.</p> <p>For SpyGlass, see the following:</p> <ul style="list-style-type: none">- SpyGlass Power Estimation and Rules Reference- SpyGlass Power Estimation Methodology Guide
<i>ZeBu Verdi Integration Guide</i>	Provides Verdi features that you can use with ZeBu. This document is available in the Verdi documentation set.
<i>ZeBu Server LCA Features Guide</i>	Provides a list of LCA features available with ZeBu Server.
<i>ZeBu Server Release Notes</i>	Provides enhancements and limitations for a specific release.

1 Safety Recommendations

This section provides safety-related recommendations. Before proceeding with software installation, read the ZeBu Server safety recommendations outlined here.

Usage Guidelines for ZeBu Server Unit

The following guidelines have to be followed when using the ZeBu Server unit:

- The ZeBu Server unit must be used, installed, and stored as described in this guide.
- The ZeBu Server unit is intended for indoor use only.
- Before connecting the ZeBu Server unit to your PC(s), hard core(s) or target(s), the power to the PC must be switched OFF, and the power cord removed.
- Internal voltages inside the PC are potentially lethal.
- The ZeBu Server unit is not intended to be serviced onsite. Service operations should only be performed by Synopsys qualified and trained personnel or representatives.
- It is the user's responsibility to ensure that the ZeBu Server unit is used in safe working conditions. While the ZeBu Server unit is in use, make sure the covers are not removed.
- In the event of damage of the ZeBu Server unit due to an action or omission by the user (including the actions or omissions described hereafter), the user shall promptly notify Synopsys and shall follow Synopsys's instructions based on your License Agreement.

Damaged ZeBu Server Unit Replacement Limitations

Synopsys shall not replace the ZeBu Server unit in case of improper use or of damage caused by the user, including the following cases:

- Damage during shipment other than shipment from Synopsys to the user.
- Damage caused by accidents, such as objects dropping, board falling to the ground, spilled liquid, or immersion in liquid.
- Damage caused by failure to provide a suitable installation environment for the ZeBu Server unit, as described in this guide.
- Damage caused by installation of the ZeBu Server unit that is not compliant with this guide.

- Damage caused by an overheat due to the use of the ZeBu Server unit in a room exceeding the normal operating temperature range indicated in this guide.
- Damage caused by power failure or power surges.
- Damage caused by use of the ZeBu Server unit for purposes or in any manner other than those indicated in this guide.
- Damage caused by short-circuit coming from a target system or produced by the user on an in-circuit emulation connector.
- ZeBu Server unit configuration has been modified by the user.
- Any minor ZeBu Server unit malfunction that would not alter the functionalities or that would not significantly affect the level of performance of the product as per the agreement.
- Damage caused by conditions of use or storage of the ZeBu Server unit which do not conform to this guide.

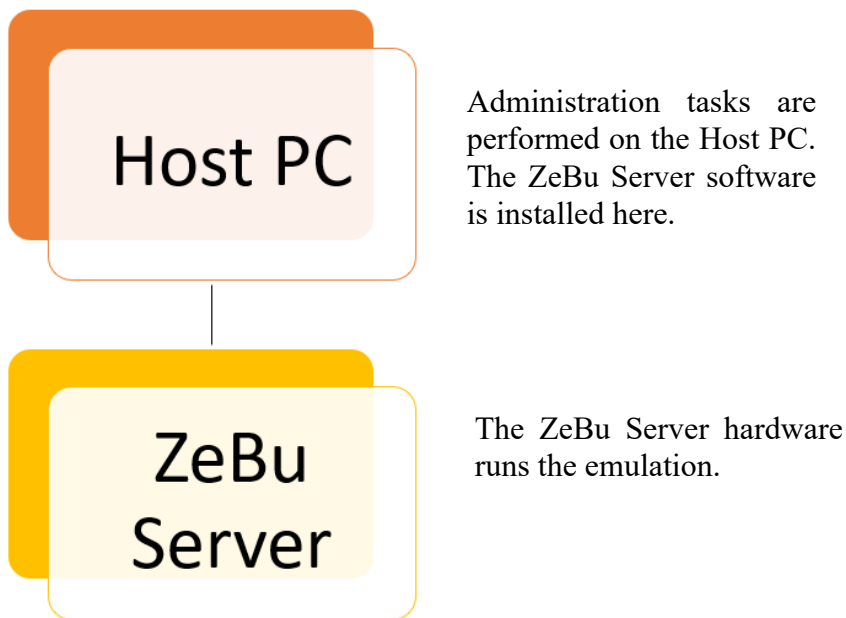
Any modification of the configuration of the ZeBu Server unit or installation of an accessory or upgrade shall be performed by Synopsys. No maintenance services, update or damaged ZeBu Server unit replacement is performed on ZeBu Server unit or on ZeBu Server unit's configuration altered by the user.

Any accessory or upgrade shall remain Synopsys's property, and on termination of the agreement executed between Synopsys and the user, the user shall return all Synopsys property to Synopsys.

For technical assistance, contact zebu_support@synopsys.com.

2 Overview of Administration Activities

After reading the [Safety Recommendations](#), you can proceed with installing ZeBu Server. The ZeBu Server emulator is connected to a Host PC, as shown in the following figure. This section describes the high-level tasks that a ZeBu Server administrator needs to perform on a Host PC before using the ZeBu Server system.

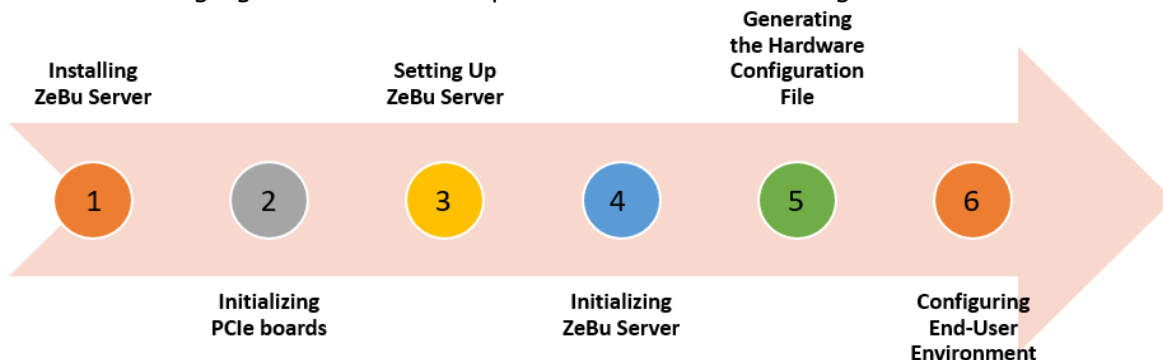


For more information, see the following sections:

- [ZeBu Server Installation Process](#)
- [Hardware and Software Requirements for the Host PC](#)

2.1 ZeBu Server Installation Process

The following figure shows the steps involved when installing ZeBu Server.



The ZeBu Server installation process steps are as follows:

- **Step 1:** Installing the ZeBu Server software involves various activities, such as downloading the packages for installation and installing licenses. For more information, see [Installing ZeBu Server Software](#).
- **Step 2:** Initializing PCIe boards on the host PC requires loading a specific ZeBu Device Driver or a ZeBu loadable kernel module. For more information, see [Initializing the PCIe Boards](#).
- **Step 3:** Setting up the ZeBu Server system involves using the `zSetupSystem` tool. You need to perform this step after each hardware configuration change because it launches system calibration and diagnostics, tests the memories, and generates a configuration file that should be used for compilation. For more information, see [Setting Up ZeBu Server](#).
- **Step 4:** Initializing the ZeBu Server system involves using the `zSetupSystem` tool. You must reinstall the ZeBu software every time you receive a new ZeBu release. You need to perform Step 4 every time the system is switched ON. For more information, see [Initializing the ZeBu Server System](#).
- **Step 5:** Generating the hardware configuration file is required for each hardware configuration change. The file is used when compiling a design. For more information, see [Generating the Target Hardware Configuration File](#).
- **Step 6:** Configure the end-user environment involves activities, such as setting mandatory environment variables. For more information, see [Configuring the End-User Environment](#).

2.2 Hardware and Software Requirements for the Host PC

You install the ZeBu Server software on a Host PC. The Host PC can either be connected or not physically connected to the ZeBu Server unit. See the ***ZeBu Server 3 Site Planning Guide*** for the Host PC hardware requirements when installing ZeBu Server 3. See the ***ZeBu Server 4 Site Planning Guide*** for the Host PC hardware requirements when installing ZeBu Server 4.

For more information on the Host PC software requirements, see the following subsections:

- [License Software Compatibility](#)
- [Synopsys Interoperable Technologies](#)
- [Third-Party Tools](#)

Note

The absolute path of the installation directory must be identical on the PC from which the ZeBu software is installed and on the PC from which the ZeBu software is used.

After installation, the ZeBu software must not be copied to another directory in the network.

2.2.1 License Software Compatibility

Using ZeBu requires two FLEXnet license servers:

- The ZeBu software uses the Synopsys Common Licensing (SCL) version 11.9 or higher.
- The Xilinx ISE, Vivado, and Triton Placer for Vivado (see [Table 2](#)) uses a separate license server.

The license servers and the ZeBu software can run on the same PC or on different PCs based on your IT configuration.

For more information about the versions of ISE and Vivado supported by this ZeBu Server release, see ***ZeBu Server Release Notes***.

ZeBu License Software Compatibility

For information on the software compatibility, see [Supported SCL Platforms by Operating](#)

[System & Platform Keyword](#) page from the [Synopsys Licensing QuickStart Guide](#), available at the Synopsys website.

Xilinx License Software Compatibility

The Xilinx license software runs on the following operating systems:

- 32-bit Linux
- 64-bit Linux

2.2.2 Synopsys Interoperable Technologies

Synopsys has many technologies that work with ZeBu Server and some of its specific features. The following table shows the technologies that have been successfully tested with ZeBu Server.

For more information on the tested versions of these technologies, see the ***ZeBu Server Release Notes***.

TABLE 1 Synopsys Interoperable Technologies

Tool	Description
VCS MX	HDL simulator It is mandatory to install and run the VCS release provided with the ZeBu release package.
MVtools	Synopsys Low Power Verification Tools Suite
Verdi ³	Debug environment. It is mandatory to install and run the Verdi release provided with the ZeBu release package.
Siloti	Verdi ³ waveform viewer.

For more information on these technologies and the ZeBu features requiring them, see section [Downloading Required Interoperable Technologies](#).

2.2.3 Third-Party Tools

ZeBu also works in association with third-party tools. Some of these tools are mandatory and others are left to your choice.

It is recommended that third party tools be installed and run in 64-bit mode on 64-bit PC configurations. This is mandatory for runtime tools such as gcc or SystemC.

Tested Third-Party Tools

The following table provides examples of third-party tools that have been successfully tested by Synopsys.

TABLE 2 Tested Third-party Tools

Tool	Description
SystemC	Required for SystemC cosimulation (http://www.systemc.org)
gcc	C compiler, part of the Linux operating system distribution (http://gcc.gnu.org/) Required for C/C++, SystemC cosimulation, transaction-based verification. Required during installation process in case of kernel compilation.
Vivado Place & Route for ZeBu Server 3 and ZeBu Server 4	Xilinx tools for FPGA Place & Route during ZeBu compilation. http://www.xilinx.com
Triton Placer for Vivado in Zebu Server 3	Included in ZeBu Server software package (see sections Xilinx Place & Route Software and Downloading the ZeBu Software Package).

The third-party tools tested with a given ZeBu software release are listed with version information in the corresponding **ZeBu Server Release Notes**.

FPGA Synthesizers

ZeBu includes its own FPGA synthesizer.

Xilinx Place & Route Software

The ZeBu delivery package includes specific Xilinx ISE, Vivado, and Triton packages. They are subsets of Xilinx software supporting FPGA Place & Route for ZeBu.

The versions of the Xilinx ISE, Vivado, and Triton Placer for Vivado that have been tested for a ZeBu software version are mentioned in the corresponding **ZeBu Server Release Notes**.

Some specific patches developed by Xilinx for ZeBu might be included in this subset. Also, the ZeBu software has been optimized for a specific subset of Xilinx Place & Route software. Therefore, it is **mandatory** to use the Xilinx release provided with the ZeBu release.

Note

Installing the Xilinx subset for ZeBu during installation of the ZeBu software package is mandatory.

After the installation, ZeBu compilation settings for FPGA Place & Route force the use of this dedicated Xilinx Place & Route installation (see chapter [Configuring the End-User Environment](#)), independently of any other Xilinx installation.

3 Installing ZeBu Server Software

You install the ZeBu Server software on a Host PC. The Host PC can either be connected or not physically connected to the ZeBu Server unit. The absolute path to the installation directory must be identical on the PC from which the ZeBu software is installed and on the PC from which the ZeBu software is used.

The following figure shows this step in the ZeBu Server installation process.

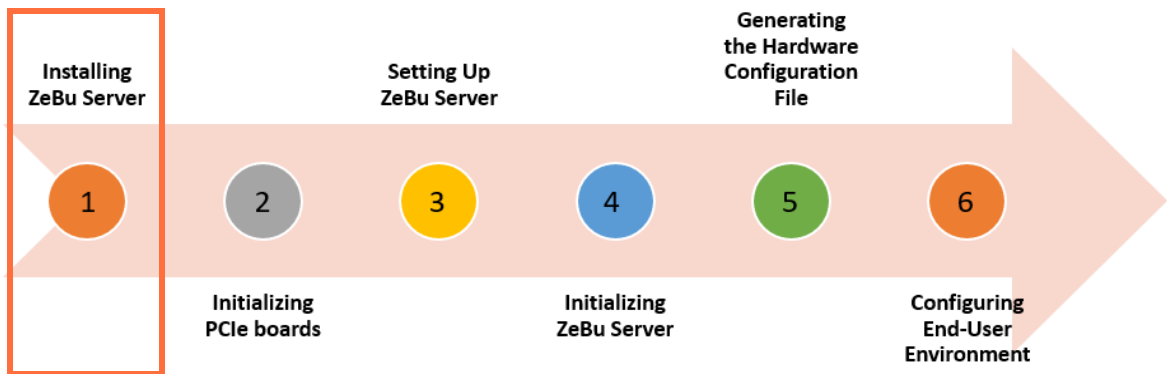


FIGURE 1. Step 1: ZeBu Server Installation Process

This section provides information on the following:

- [Downloading Packages for Installation](#)
- [Installing the ZeBu Server Software](#)
- [Downloading Required Interoperable Technologies](#)
- [Downloading Diagnostics Patches for ZeBu Server 3](#)
- [Diagnostics for ZeBu Server 4](#)
- [Post Installation Activities](#)
- [Installing Licenses](#)

After installation, the ZeBu software must not be moved or copied to any other directory in the network. If you want to reinstall the ZeBu software, make sure you install it in a new directory.

3.1 Downloading Packages for Installation

Before you can download a ZeBu Server package, you must first download the Synopsys Installer from SolvNetPlus.

For more information, see the following subsections:

- [Downloading and Extracting the Synopsys Installer](#)
- [Downloading the ZeBu Software Package](#)

3.1.1 Downloading and Extracting the Synopsys Installer

Before installing the ZeBu software, it is mandatory to first download the **Synopsys Installer** from SolvNetPlus and then extract it.

1. Log on to the Synopsys SolvNetPlus website (<https://solvnetplus.synopsys.com>) with your credentials.
2. Click **Downloads** on the SolvNPlus home page.

Welcome to the Synopsys Support Community!

A place where you can easily find solutions and ask questions



3. From the **Downloads** list, choose the **Synopsys Installer** and then choose the latest version available.
4. In the next page, click **Download Here**.
The **Electronic Software Transfer** terms and conditions are displayed.
5. Click **YES, I AGREE TO THE ABOVE TERMS** at the end of **Electronic Software Transfer** terms. The **Synopsys Electronic Software Transfer (EST)** page opens.
6. Download the self-extracting file `SynopsysInstaller_v5.1.run` containing the **Synopsys Installer** software to a dedicated directory. For example, `/home/<user>/install_files`. For more details, see **Synopsys Installer on SolvNet**.
7. After the file is downloaded, execute the following command to extract its content:

Downloading Packages for Installation

```
$ cd /home/<user>/install_files
$ SynopsysInstaller_v5.1.run
```

The **Synopsys Installer** script prompts you to specify the directory where it should be extracted.

Note

You can add the `-d` option to the command, to specify the path where the Synopsys Installer should be extracted.

For more information, see the downloadable file `zebu_installer_INSTALL_README.txt` available on the **Synopsys Electronic Software Transfer (EST)** page.

3.1.2 Downloading the ZeBu Software Package

After the **Synopsys Installer** software is downloaded and extracted, download the ZeBu Software Package from SolvNetPlus.

1. Log on to the Synopsys SolvNetPlus website (<https://solvnetplus.synopsys.com>) with your credentials.
2. Click the **Downloads** link on the page.

Welcome to the Synopsys Support Community!

A place where you can easily find solutions and ask questions



3. From the **Downloads** list, choose **ZeBu Server Software - All > Q-2020.03**.
4. In the next page, click **Download Here**.
The **Electronic Software Transfer terms and conditions** are displayed.
5. Click **YES, I AGREE TO THE ABOVE TERMS** at the end of **Electronic Software Transfer** terms.

The **Synopsys Electronic Software Transfer (EST)** page opens.

- Download all files required for the installation of the ZeBu software release:

- ☐ zebu_INSTALL_README.txt
- ☐ zebu_vQ-2020.03_linux64.spf
- ☐ zebu_vQ-2020.03_common.spf

NOTE: The zebu_vQ-2020.03_common.spf file is split into 18 parts due to size limitations and has to be reconstructed before the installation.

- Download the following files containing the Xilinx Place & Route software based on your hardware:

- ☐ Xilinx Vivado Place & Route for ZeBu Server 3 and ZeBu Server 4:
zebu_xilinx_vivado_2019_1_vQ-2020.03_common.spf

NOTE: This file has been split in 8 parts due to size limitations and has to be reconstructed before the installation.

- ☐ Xilinx Triton Placer & Route for ZeBu Server 3:
zebu_xilinx_triton_2017_1_sp42h_vQ-2020.03_common.spf

- Download the ZeBu hardware diagnostics packages matching your ZeBu configuration among the following files:

- ☐ zebu_zs3_2_10u_vQ-2020.03_common.spf
- ☐ zebu_zs3_2s_vQ-2020.03_common.spf
- ☐ zebu_zs3_5s_vQ-2020.03_common.spf

For more information on the diagnostics packages, see section [Downloading Diagnostics Patches for ZeBu Server 3](#).

Note

For ZeBu Server 4, there is no diagnostics package as the diagnostics patches are provided in the release package.

Tip: To reduce disk space usage, you should download only diagnostics packages matching your requirements.

After you have downloaded all the files, move them to the directory containing the Synopsys Installer software, which is described in [Downloading and Extracting the Synopsys Installer](#) (for example, in the /home/<user>/install_files directory).

- Download the **ZeBu Device Driver Switching Solution** package that contains the following drivers:

Downloading Required Interoperable Technologies

- ❑ For ZeBu Server 3, see <https://solvnet.synopsys.com/retrieve/2648882.html>
 - ❑ For ZeBu Server 4, see <https://solvnet.synopsys.com/retrieve/2648858.html>
- For more information, see [Initializing the PCIe Boards](#).

3.2 Downloading Required Interoperable Technologies

ZeBu Server requires specific Synopsys interoperable technologies.

VCS

The appropriate packages for VCS are located in the same directory as the ZeBu release:

- `vcs-mx_vQ-2020.03_common.spf`
- `vcs-mx_vQ-2020.03_linux.spf`
- `vcs-mx_vQ-2020.03_linux64.spf`

Verdi

To use enhanced debugging with ZeBu Server, download and install the supported releases of Verdi 3 that is listed in the *ZeBu Release Notes*. The location to download Verdi depends on the required release. If the Verdi versions ends with -Z (for example, `Q-2020.03-Z`), the corresponding package is found with the ZeBu release:

- `verdi_vVerdi_Q-2020.03-Z_common.spf`
- `verdi_vVerdi_Q-2020.03-Z_linux64.spf`

If the Verdi versions does not end with -Z (for example, `Q-2020.03`), the corresponding package is found in the Verdi section of SolvNetPlus.


3.3 Downloading Diagnostics Patches for ZeBu Server 3


Diagnostics patches are specific to the architecture of the ZeBu system (number and types of FPGA modules in the system) and to each release. The diagnostics tool **zutils** uses them to check your system after the installation is complete (typical use is described in chapter [Setting Up ZeBu Server](#)).

Diagnostics patches are grouped into diagnostics packages, which are described in [Diagnostics Packages for ZeBu Server 3](#). However, for ZeBu Server 4, the diagnostics patches are provided in the release package.

For the targeted ZeBu version, check the *ZeBu Server Release Notes* for a list of available diagnostics packages and patches.

Note

 To reduce disk space usage, download only the diagnostics packages matching your ZeBu configuration.

 For ZeBu Server 4, there is no diagnostics package as the diagnostics patches are provided in the release package.

3.3.1 Diagnostics Packages for ZeBu Server 3

Diagnostics are grouped into diagnostics packages. The following packages are available for ZeBu Server 3:

- `zebu_zs3_2s_vq-2020.03`: For a 2-slot, single-unit ZeBu Server 3 systems
- `zebu_zs3_5s_vq-2020.03`: For 5-slot, single-unit ZeBu Server 3 systems
- `zebu_zs3_2_10u_vq-2020.03`: For 2-unit to 10-unit, ZeBu Server 3 systems

3.3.2 Diagnostics for ZeBu Server 3

After installation, Diagnostics for ZeBu Server 3 are located in `<installation_directory>/etc/firmwares/ZSE/dt/V6.0.`

See the name of the diagnostic patch to determine the intended system configuration. For more information, see the following examples:

- [Example of 2-slot Single Unit ZeBu Server 3 System](#)
- [Example of 5-slot Single Unit ZeBu Server 3 System](#)
- [Example of Multiunit ZeBu Server 3 System \(with up to 5 units\)](#)
- [Example of Multiunit ZeBu Server 3 System \(with more than 5 units\)](#)

3.3.2.1 Example of 2-slot Single Unit ZeBu Server 3 System

This example is for a 2-slot single unit ZeBu Server 3 system, which has one 9F/ICE module and one 9F module. The package name is as follows:

`XC7V-203_92_81.diag`

The following figure shows the interpretation of this ZeBu Server 3 diagnostic patch name. The patch name starts with `XC7Z`, which means that the patch is intended for a ZeBu Server 3 system.

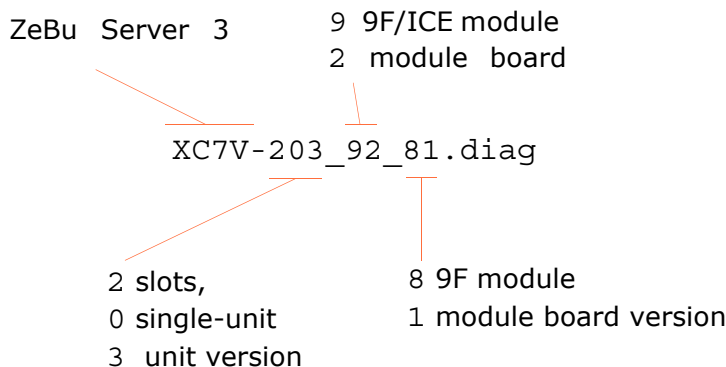


FIGURE 2. Interpreting Diagnostics Patch Name for a 2-Slot, Single Unit ZeBu Server 3 System

3.3.2.2 Example of 5-slot Single Unit ZeBu Server 3 System

This example is for a 5-slot single unit ZeBu Server 3 system, which has five 9F/ICE module. The package name is as follows:

XC7V-503_92_92_92_92_92.diag

The following figure shows the interpretation of this ZeBu Server 3 diagnostic patch name. The patch name starts with XC7Z, which means that the patch is intended for a ZeBu Server 3 system.

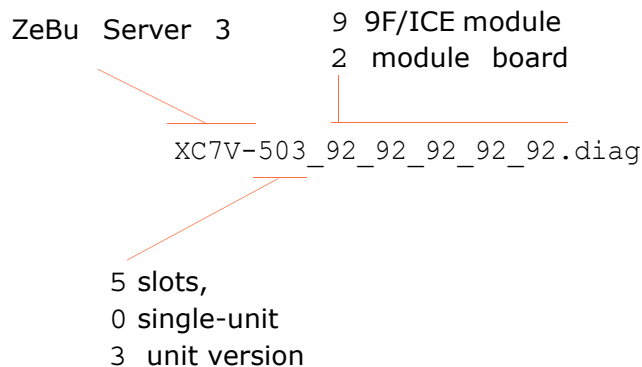


FIGURE 3. Interpreting Diagnostics Patch Name for a 5-Slot, Single Unit ZeBu Server 3 System

It is intended for the following system configuration:

- XC7V → ZeBu Server 3 system
- 503 → 5:5 slots, 0: single-unit, 1: unit version
- 92 → 9: 9F/ICE module, 2: module board version

3.3.2.3 Example of Multiunit ZeBu Server 3 System (with up to 5 units)

The package name is as follows:

XC7V-C00-02-514_81_81_81_81_81-514_81_81_81_81_81.diag

It is intended for the following system configuration:

- XC7V → ZeBu Server 3 system

 Downloading Diagnostics Patches for ZeBu Server 3

- C00-02 → <Cxx-yz>, where:
 - Cxx → Cable configuration label (C00 to C99)
 - y → Id of the unit where the hub is plugged (0: unit Id; default unit: U0)
 - z → Hub version (2: hub version)

NOTE: C00-00: Multiunit system in a 1-unit configuration (-00 → no hub).

- 514 → 5: 5 slots, 1: multi unit, 4: unit version
- 81 → 8:9F module, 1: module board version

3.3.2.4 Example of Multiunit ZeBu Server 3 System (with more than 5 units)

XC7V-C02-10U_84481.diag

It is intended for the following system configuration:

- XC7V → ZeBu Server 3 system
- C02 → Cable configuration label (C00 to C99)
- 10U → Number of units (06U to 10U)
- 84481 → Interconnection topology

Note

In the previous example (configurations with more than 5 units), a different file format is used because of a length constraint in the file names.

3.4 Diagnostics for ZeBu Server 4

Diagnostics for ZeBu Server 4 are included in the release and do not need to be downloaded separately.

After installation Diagnostics for ZeBu Server 4 are located in:

```
<installation_directory>/etc/firmwares/ORION/dt/V3.0/XCVU-1U-12C.diag
```

Example Unit Configuration of ZeBu Server 4

The XCVU-1U-12C diagnostics adapt to your system configuration. The 1U-12C diagnostics are compatible with a mono unit system, with any number of modules from 1 to 4.

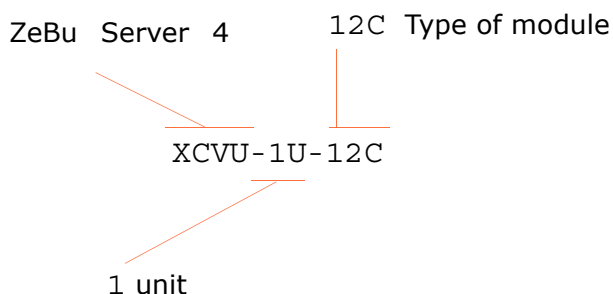


FIGURE 4. Example: Interpretation of Diagnostics Patch for a Single Unit ZeBu Server 3 System

3.5 Installing the ZeBu Server Software

After you have downloaded all the required files in the `/home/<user>/install_files` directory, launch the installation of the ZeBu software.

The installation process is similar when you install the ZeBu software for the first time or for modification of an already installed release (for example, to install a diagnostics patch).

Installing the ZeBu software and the Xilinx Place & Route software can be done with

user privileges. Make sure appropriate write accesses is granted on the disk space where the software is installed.

Note

Before launching the installation, make sure the `DISPLAY` variable is correctly set in your installation environment.

You can install the ZeBu Server software using either a graphical interface or by using a script. For more information, see the following subsections:

- [Using Graphical Mode to Install ZeBu Server](#)
- [Using Script Mode to Install ZeBu Server](#)

3.5.1 Using Graphical Mode to Install ZeBu Server

To install the ZeBu Software using a graphical interface, perform the following:

1. Open a terminal and enter the following commands:

```
$ cd /home/<user>/install_files
$ ./installer -gui -new_installer
```

The graphical interface of the **Synopsys Installer** is displayed.

2. Click **Start**.
3. Enter your site information and click **Next**.
4. Enter or browse the path to the directory where you have downloaded the ZeBu software release, the Xilinx Place & Route software, and the required diagnostics package(s), then click **Next**.

The **Synopsys Installer** extracts the content of the ZeBu software release in a temporary directory `snps_installer_temp_<username>`.

5. Enter or browse the path to the installation directory and click **Next**.
6. In the **Product and Release Selection** window, perform the following steps:
 - a. Select the **ZeBu (R) Server-3 / ZeBu (R) Server-4 Emulation** product (marked as **STAND-ALONE**).
 - b. Select the Xilinx Place & Route software based on your hardware (marked as **OVERLAY**):

- ◆ For ZeBu Server 3 and ZeBu Server 4: **xilinx Vivado 2019.1 for ZeBu®**

♦ For ZeBu Server 3: **Xilinx Triton 2017.1-sp42H for ZeBu®**

7. Select the diagnostics package corresponding to your hardware (marked as **OVERLAY**).
8. Click **Next**. Several pop-up windows with the Xilinx license agreement are displayed.

In each window, click **I have read the license agreement**, and click **I Accept**.

9. For all the products that you want to install, confirm, or modify in the next screen, the following information is displayed:
 - ☐ The installation path
 - ☐ The platform and operating system matching your environment
10. Click **Next**.
11. If you install the Xilinx Place & Route software along with the ZeBu software, a pop-up window states that Place & Route software (**OVERLAY** product) is installed in the same directory as the ZeBu Server software (**STAND-ALONE** product). Click **Yes** to continue.

NOTE: If you click **No**, a warning pop-up window appears stating that you cannot install the item. The installation is then stopped.

12. If you install one or several diagnostics packages along with the ZeBu Server software, a pop-up window states that the diagnostics package(s) (**OVERLAY** product) is installed in the same directory as the ZeBu Server software (**STAND-ALONE** product). Click **Yes** to continue.

NOTE: If you click **No**, a warning pop-up window appears stating that you cannot install the item. The installation is then stopped.

13. In the summary screen, review the installation parameters and click **Accept, Install** to proceed with the installation.
14. Click **Finish** to exit the Synopsys Installer.

A **Release Notes** window appears with the following information:

- ☐ The Synopsys Common Licensing
 - ☐ Post installation requirements
15. Click **Dismiss** to exit.
- The installation is completed.

3.5.2 Using Script Mode to Install ZeBu Server

To install the ZeBu Software in using a script, Open a terminal and execute the following commands:

```
$ cd /home/<user>/install_files  
$ ./installer
```

The script mode of the **Synopsys Installer** is launched. You must fill in all the requested parameters for the installation.

Note

Unlike the graphical mode described in [Using Graphical Mode to Install ZeBu Server](#), the script mode does not allow you to install the ZeBu software (STAND-ALONE product) along with the Xilinx Place & Route software and the diagnostics packages simultaneously (OVERLAY products).

3.5.2.1 Installation of Xilinx Place & Route Software

To install the Xilinx Place & Route software, perform the following steps:

1. At the end of the ZeBu software installation, press **b** in the following script:

```
Installation has finished successfully.  
Entry "b" to go back to install another product or any other key/  
Return to quit[]:
```

2. Select the appropriate menu entries for the Xilinx software:

```
Select Synopsys product(s) to install:

    [ 1] zebu - ZeBu (R) Server 1 / ZeBu (R) Server 2 / Zebu
(R) Server 3 / ZeBu (R) Blade 2 Emulation  (STAND-ALONE)

    [ 2] zebu_xilinx_triton_2017_1_sp42H - Xilinx triton-
2017.1-sp42H for ZeBu (R)  (OVERLAY)

    [ 3] zebu_xilinx_vivado_2019_1 - Xilinx Vivado 2019.1 for
ZeBu (R)  (OVERLAY)

...

    [17] zebu_zs3_2_10u - ZeBu (R) Server 3 Multi-Unit
diagnostics package  (OVERLAY)

    [18] zebu_zs3_2s - ZeBu (R) Server 3 2 Slots diagnostics
package  (OVERLAY)

    [19] zebu_zs3_5s - ZeBu (R) Server 3 5 Slots diagnostics
package  (OVERLAY)

    [ b] back - Back to Select Another Version
```

3.5.2.2 Installation of a Diagnostics Package

To install a diagnostics package, perform the following steps:

1. At the end of the ZeBu software installation, press **b** in the following script:

```
Installation has finished successfully.

Entry "b" to go back to install another product or any other key/
Return to quit[]:
```

2. Select the appropriate menu entries for the desired diagnostics package:

```
Select Synopsys product(s) to install:

    [ 1] zebu - ZeBu (R) Server 1 / ZeBu (R) Server 2 / Zebu
(R) Server 3 / ZeBu (R) Blade 2 Emulation  (STAND-ALONE)
```

```
...  
[17] zebu_zs3_2_10u - ZeBu (R) Server 3 Multi-Unit  
diagnostics package (OVERLAY)  
[18] zebu_zs3_2s - ZeBu (R) Server 3 2 Slots diagnostics  
package (OVERLAY)  
[19] zebu_zs3_5s - ZeBu (R) Server 3 5 Slots diagnostics  
package (OVERLAY)
```

3.6 Post Installation Activities

The installation directory (`/usr/share/zebu/Q-2020.03/` in the previous example) contains the ZeBu tools and the installed Xilinx Place & Route software.

The diagnostics patches have to be installed after the initial installation of the ZeBu software release with the Synopsys Installer, as described in section [Downloading Diagnostics Patches for ZeBu Server 3](#).

The directory containing the installation files (where a temporary installation directory named `snps_installer_temp_<username>` was also created) might be deleted after the installation is complete.

At this point, you have an operational compilation chain for any Linux PC after configuring the end-user's environment as described in chapter [Configuring the End-User Environment](#) and creating the target hardware configuration file as described in [Generating the Target Hardware Configuration File](#).

Note

After the ZeBu software is installed, it must not be moved or copied to another directory in the network.

3.7 Installing Licenses

ZeBu Server requires the following FLEXnet licenses:

- For the ZeBu Server software license:
 - The FLEXnet license daemon (`lmgrd`)

- ☐ The Synopsys vendor license daemon (`snpslmd`)
- ☐ The license file
- For the Xilinx ISE or Vivado Place & Route software license:
 - ☐ The FLEXnet license daemon (`lmgrd`)
 - ☐ The Xilinx vendor license daemon (`xilinxld`)
 - ☐ The license file

For information about FLEXnet licensing tools, see the documentation from the Flexera™ website at <http://www.flexerasoftware.com>.

FLEXnet recommends you to launch the license manager without root privileges for security reasons.

For more information, see the following subsections:

- [Installing ZeBu and Xilinx Licenses](#)
- [Modifying License Files](#)
- [Starting the License Server](#)

3.7.1 Installing ZeBu and Xilinx Licenses

This section describes the following:

- [Installing ZeBu License Server](#)
- [Installing Xilinx License Server](#)

3.7.1.1 Installing ZeBu License Server

For more information on the ZeBu license server installation, see the [Downloading & Installing SCL](#) section of the [Synopsys Licensing QuickStart Guide](#), available on the Synopsys website.

3.7.1.2 Installing Xilinx License Server

You can install the FLEXnet license manager on a PC (with Linux operating system) or on a Sun SPARC (with Solaris operating system). The installation process is the same for both operating systems.

The Xilinx license vendor daemon (`xilinxd`) is available in the following directory of the ZeBu software release:

```
<install_path>/vivado/bin/unwrapped/lnx64.0/xilinxd
```

The name of the Xilinx license file is `Xilinx.lic`.

Each delivered license file is specific to one license server (host).

3.7.2 Modifying License Files

This section describes the following:

- [Modifying the ZeBu License File](#)
- [Modifying the Xilinx License File](#)

3.7.2.1 Modifying the ZeBu License File

For more information on the modification of the ZeBu license file, see the [Customizing the License Key File](#) section of the [Synopsys Licensing QuickStart Guide](#), available on the Synopsys website.

If you need to update your ZeBu license file, see the **Procedure to Update a Synopsys License File** section in the [SCL Administration Guide](#) (available in the Documentation section of the [Synopsys Licensing QuickStart Guide](#)).

3.7.2.2 Modifying the Xilinx License File

After you have received the license file, modify it to match your IT configuration. The line `SERVER <hostname> <hostid> <portnumber>` must be modified with the following information:

- Change `<hostname>` to your server hostname by using the result of the `uname -n` command on the license server.
- Change `<portnumber>` to an unused TCP port number. The port number must be higher than 1024 when the license manager is launched with user privileges as recommended by the FLEXnet user documentation).

- The <hostid> value must not be modified manually. If this value is modified, the license file is no longer valid. If it does not match the result of `lmhostid` on the license server, contact Synopsys support.
- It is recommended to copy the license files and the vendor daemons in the same directory as the license manager utilities. However, if the vendor daemons are not stored in the same directory as the license manager daemon (`lmgrd`), modify the `VENDOR` line of the license file with the path to the vendor daemon:

```
VENDOR <vendor_daemon> <path_to_vendor_daemon>
```

Note

If the vendor daemon is in the same directory as the license manager daemon, the `VENDOR` line must be present; the path information is ignored.

Example

A part of the initial content of the license file for the Xilinx software is as follows:

```
SERVER hostname1 12345678 2100
USE_SERVER
VENDOR xilinxd
```

After modification, the license file for the Xilinx software is as follows:

```
SERVER myHost 12345678 1235
USE_SERVER
VENDOR xilinxd /local/licenses/zebu/xilinxd
```

3.7.3 Starting the License Server

This section describes the following:

- [Starting the ZeBu License Server](#)
- [Starting the Xilinx License Server](#)

3.7.3.1 Starting the ZeBu License Server

For more information on how to start the ZeBu license server, see [Starting the License Server](#) section of the [Synopsys Licensing QuickStart Guide](#), available on the Synopsys website.

3.7.3.2 Starting the Xilinx License Server

Execute the following command to start the Xilinx license server:

```
$ <lm_path>/lmgrd -c <Xilinx_lic_file> -l <lm_logpath>/
<Xilinx_lic_logfile>
```

Where:

- **<lm_path>**: Specifies the path to the license manager daemon
- **<Xilinx_lic_file>**: Specifies the path to the Xilinx license file
- **<lm_logpath>**: Specifies the path where you want the license log files to be stored

After the `$LM_LICENSE_FILE` variable is set as described in section [Activating a ZeBu and Xilinx License](#), check the license server is running correctly using the `lmstat -a` command (the host name is `myHost` and the port number is 1235):

```
lmstat - Copyright (c) 1989-2006 Macrovision Europe Ltd. and/or
Macrovision Corporation. All Rights Reserved.
Flexible License Manager status on Mon 11/3/2013 19:21
License server status: 2100@myHost
    License file(s) on myHost: /remote/license/data/Xilinx/
myHost_Xilinx.lic:
    myHost: license server UP (MASTER) v11.6
...
```

```
Vendor daemon status (on myHost):  
  xilinxd: UP v11.6  
Feature usage info:  
Users of Logic_Edition:  (Total of 20 licenses issued;  Total of 0  
licenses in use)  
...  
Users of ISE:  (Total of 40 licenses issued;  Total of 0 licenses  
in use)  
Users of Implementation:  (Total of 20 licenses issued;  Total of 0  
licenses in use)  
...
```

4 Initializing the PCIe Boards

This chapter provides information on how to initialize the PCIe boards located on the host PC. The following figure shows this step in the ZeBu Server installation process.

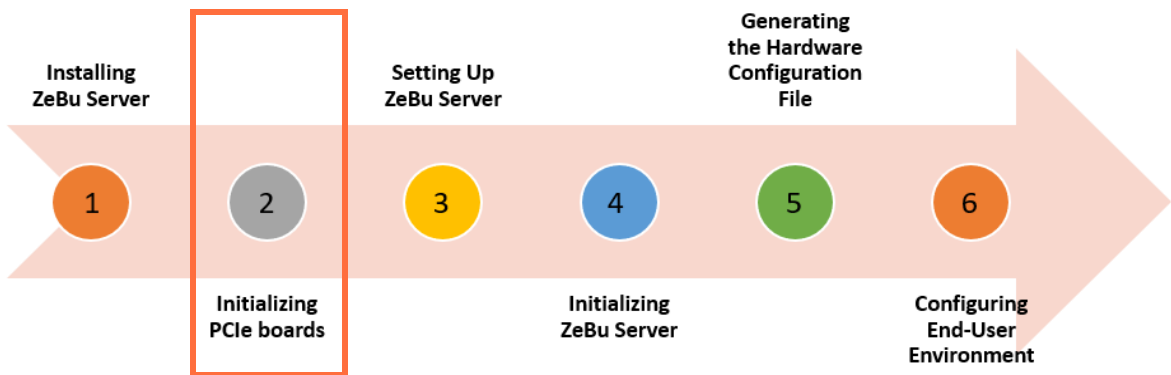


FIGURE 5. Step 2: ZeBu Server Installation Process

This section has the following subsections:

- [Overview of Initialization of PCIe Boards](#)
- [Prerequisites of Initializing PCIe Boards](#)
- [Initializing PCIe Boards on Host PC Running ZeBu 2017.03 or Later Releases](#)
- [Initializing PCIe Boards on Host PC Running 2020.03 and Releases Earlier than 2017.03 \(2015.09 or Earlier\)](#)
- [Installing the ZeBu Server 3 Kernel Module on the Host PC \(Releases Earlier Than 2017.03\)](#)
- [Initializing the PCIe Boards Automatically When Booting Linux](#)
- [PCIe Boards Detection Order](#)
- [Viewing ZeBu Server Installation Log Files](#)
- [Updating PCIe Boards Firmware](#)

4.1 Overview of Initialization of PCIe Boards

For the ZeBu Server system, the initialization of PCIe boards on the host PC requires you to load a specific ZeBu device driver or a ZeBu loadable kernel module.

- **ZeBu device driver:** This ZeBu device driver is part of the ZeBu drivers package and is available on SolvNet. This is the only possibility for ZeBu Server 4.
- **ZeBu loadable kernel module:** The ZeBu loadable kernel module is specific to a ZeBu release (prior to the 2017.03 release) and is provided with the corresponding ZeBu release. It is not possible to use the ZeBu loadable kernel module with ZeBu Server 4.

The procedure for loading the driver or kernel module is similar to loading a driver for any PCIe peripheral running with Linux operating system. The procedure depends on the versions of the ZeBu releases used on the host PC.

- **ZeBu Server 4 and ZeBu Server 3:** Host PC runs only ZeBu Q-2020.03 and releases newer than ZeBu 2017.03 (included). The latest version of the driver must be installed on the host PC, as described in [Initializing PCIe Boards on Host PC Running ZeBu 2017.03 or Later Releases](#).

This is the only applicable scenario for ZeBu Server 4.

- **ZeBu Server 3:** Host PC runs ZeBu Q-2020.03 and releases older than 2017.03 (2015.09 or older). The initialization step of all releases is performed using the Device Driver Switching solution, with its **zSwitchDriver** executable and its associated `zInstallLauncher.sh` script, as described in [Initializing PCIe Boards on Host PC Running 2020.03 and Releases Earlier than 2017.03 \(2015.09 or Earlier\)](#).

This scenario is not applicable to ZeBu Server 4.

In addition to the initialization, the PCIe boards in the host PCs must be occasionally updated when installing a new software version or an intermediate patch. For more information, see [Updating PCIe Boards Firmware](#).

4.2 Prerequisites of Initializing PCIe Boards

The prerequisites of initializing PCIe boards on the Host PC are as follows:

- The ZeBu Device Drivers package must be downloaded and installed. It is recommended to always use the latest ZeBu Device Drivers package.
The instructions to download the latest ZeBu Device Drivers package are available on SolvNet:

- ❑ [Instructions to Download the ZeBu Device Drivers Package for ZeBu Server 4](#)
- ❑ [Instructions to Download the ZeBu Device Drivers' Package for ZeBu Server 1, 2, or 3](#)
- The ZeBu loadable kernel module (zKernel) is specific to a given ZeBu release. It is provided in the ZeBu release package.
- If you have several host PCs in your configuration, the initialization step described in [Initializing PCIe Boards on Host PC Running ZeBu 2017.03 or Later Releases](#) and [Installing the ZeBu Server 3 Kernel Module on the Host PC \(Releases Earlier Than 2017.03\)](#) must be performed on each host PC.

4.3 Initializing PCIe Boards on Host PC Running ZeBu 2017.03 or Later Releases

Before you install or remove drivers from the Host PC, which is running ZeBu 2017.03 or later releases, ensure the following:

- Stop all ZeBu emulation runs during this initialization process.
- Launch all commands described in this section with root permissions.

Installing and removing drivers involves using the `zDriverInstall` script. This script is located in `drivers/zebu_zs` of the directory where the ZeBu driver package is installed.

For more information, see the following subsections:

- [Installing the ZeBu Device Driver on the Host PC](#)
- [Removing the ZeBu Device Driver From the Host PC](#)

Note

This is the only applicable method for ZeBu Server 4.

Installing the ZeBu Device Driver on the Host PC

To install the device driver on the host PC, execute the `zDriverInstall install` script available in the installed package:

```
<driver_package_path>/drivers/zebu_zs/zDriverInstall install
```

Where, `driver_package_path` corresponds to the path where the ZeBu driver package is installed using the Synopsys installer.

Note

Execute this script only when preparing the Host PC when installing the ZeBu Server software for the first time.

The ZeBu device driver, `zebu_zs`, is permanently added as a device driver on the runtime PC using `zDriverInstall install`. The ZeBu device driver is relaunched automatically when the PC is restarted.

The driver installation script creates the following directories, which are required by ZeBu software applications, on the Host PC:

- `/zebu`
- `/zebu/queue`
- `/zebu/zUtils_logs`

Note

The `/zebu` directory must be local to the host PC. It must neither be mounted through NFS nor shared between the host PCs. The `/zebu` directory is specific to each host PC.

Removing the ZeBu Device Driver From the Host PC

To remove the device driver from the host PC, execute the `zDriverInstall remove` script available in the installed package:

```
<driver_package_path>/drivers/zebu_zs/zDriverInstall remove
```

Where, `driver_package_path` corresponds to the path where the ZeBu drivers package is located.

4.4 Initializing PCIe Boards on Host PC Running 2020.03 and Releases Earlier than 2017.03 (2015.09 or Earlier)

In addition to the ZeBu device driver, the Zebu drivers package contains the Device Driver Switching solution, with the `zSwitchDriver` executable and its associated `zInstallLauncher.sh` script. The initialization step of all releases is performed using

the Device Driver Switching solution.

Note

This section is not applicable to ZeBu Server 4.

Before you install or remove drivers from the Host PC, which is running 2020.03 and releases earlier than 2017.03 (2015.09 or earlier), ensure the following:

- Stop all ZeBu emulation runs during this initialization process.
- Execute the `zInstallLauncher.sh` script for each version.
- Launch all commands described in this section with root permissions.
- It is recommended to always use the latest version of `zSwitchDriver`.

For more information, see the following subsections:

- [Preparing the Host PC for ZeBu Releases Installation](#)
- [Installing the ZeBu Server 3 Device Driver on the Host PC \(Q-2020.03, O-2018.09, and M-2017.03\)](#)
- [Installing the ZeBu Server 3 Kernel Module on the Host PC \(Releases Earlier Than 2017.03\)](#)

Preparing the Host PC for ZeBu Releases Installation

You need to prepare the Host PC for ZeBu installation only for the first time. Thereafter, you do not need to perform this step.

To clean your workstation, execute the following command:

```
<driver_package_path>/zSwitchDriver/zInstallLauncher.sh -clean
```

Installing the ZeBu Server 3 Device Driver on the Host PC (Q-2020.03, O-2018.09, and M-2017.03)

To install the ZeBu device driver that is used by Q-2020.03, O-2018.09, and M-2017.03 releases, specify the following:

```
<driver_package_path>/zSwitchDriver/zInstallLauncher.sh
-start -r <zebu_release_path>
[-driverPath <driver_package_path>/drivers/zebu_zs]
[-driverModulePath <path_to_driver_module_file>]
```

Where:

- `<driver_package_path>`: (Optional) Corresponds to the path where the ZeBu drivers package is installed using the Synopsys installer. `-driverPath` is optional.
- `-driverPath`: (Optional) This option is required if the package hierarchy is modified (directories moved from the ZeBu drivers package) to indicate the path of the `zebu_zs` driver.
- `-driverModulePath <path_to_driver_module_file>`: (Optional) Indicates the path to the directory containing the precompiled driver (`zebu_zs.ko`).
- `-r <zebu_release_path>`: Corresponds to the `$ZEBU_ROOT` of the ZeBu release.

Installing the ZeBu Server 3 Kernel Module on the Host PC (Releases Earlier Than 2017.03)

To install the ZeBu loadable kernel module that is used by releases earlier than M-2017.03, use the following:

```
<driver_package_path>/zSwitchDriver/zInstallLauncher.sh
-start -r <zebu_release_path>
```

Where:

- `<driver_package_path>`: Corresponds to the path where the ZeBu drivers package is installed using the Synopsys installer.
- `-r <zebu_release_path>`: Corresponds to the `$ZEBU_ROOT` of the ZeBu release.

Note

Execute `zInstallLauncher.sh` one time for each release earlier than M-2017.03.

By default, `zInstallLauncher.sh` searches for the `zKernel` object file (`zKernel.o`) in the following directories:

- `<zebu_release_path>/bin` (equivalent to `$ZEBU_ROOT/bin`)
- `/zebu`

If the `zKernel` object file cannot be found in these directories, `zInstallLauncher.sh` automatically compiles `zKernel` from the source file provided in the ZeBu software package.

4.5 Initializing the PCIe Boards Automatically When Booting Linux

When Linux starts, launch `zInstallLauncher.sh` by adding the following line in the `/etc/inittab` file. The line needs to be added just before the `ttys` initialization.

```
zInstallLauncher.sh -start -r <zebu_release_1_path>
zInstallLauncher.sh -start -r <zebu_release_2_path>
zInstallLauncher.sh -start -r <zebu_release_N_path>
```

Where, `zebu_release_n_path` corresponds to the `$ZEBU_ROOT` of the ZeBu releases.

Note

You must specify the full path of the software release (equivalent to `$ZEBU_ROOT`) because the environment variables are set after the `inittab` file is interpreted.

4.6 Precompiling the ZeBu Device Driver on Host PC

It is recommended to follow the steps provided in the [Installing the ZeBu Device Driver on the Host PC](#) section. However, the solution described in this section allows you to perform compilation and installation of the Linux Device Driver in different steps.

To compile the device driver on the host PC, perform the following steps:

1. Navigate to the `<driver_package_path>/drivers/zebu_zs/src` directory.
2. Execute the following commands:

```
setenv PATH /usr/bin:$PATH (optional to set the proper gcc version)
<driver_package_path>/drivers/zebu_zs/zDriverInstall compile
```

Make sure to set the `PATH` environment variable to use the proper gcc version.

The `zebu_zs.<kernel_version>.ko` file specific to the version of Linux kernel is

created in the `<driver_package_path>/drivers/zebu_zs/module` directory.

The **zDriverInstall** command automatically uses the appropriate precompiled ZeBu Device Driver.

Note

Execute this script only when the Host PC is prepared for the ZeBu Server installation for the first time.

4.7 PCIe Boards Detection Order

The units must be connected to the PCIe boards in ascending order.

There might be gaps in the unit connection order. For example, you might connect U0, U2 and U4 on three boards of your host PC. However, the connection of the boards must be always in ascending order. You cannot connect the boards in the following order: U0, U2 and U1.

The order for the addressing the PCIe boards in the host PC is defined by the BIOS.

Note

The indications printed on the motherboard of the host PC should not be taken into account.

To detect the order of the PCIe boards, perform the following steps:

1. Install your PCIe boards.
2. Execute the following command to stop a running **zSwitchDriver** and clean the environment:

```
zInstallLauncher.sh -clean
```
3. Set the `ZEBU_DEBUG_BOARD_IN_PC_MASK` environment variable with a mask for each PCIe board as follows:
 - ❑ Bit 0 of the mask is for first PCIe board (index 0). For example, if you want to detect the first PCIe board, you must set the `ZEBU_DEBUG_BOARD_IN_PC_MASK` variable to `0x1`.
 - ❑ Bit 1 of the mask is for second PCIe board (index 1), and so on. For example, if you want to detect all of the 5 PCIe boards, set the `ZEBU_DEBUG_BOARD_IN_PC_MASK` environment variable to `0x1f`, which is equivalent to having no mask at all.

PCIe Boards Detection Order

4. Execute the following command to display all information about the unmasked PCIe boards:

```
zInstallLauncher.sh -start -r <zebu_release_path>
```

5. Use the indexes to connect the units in ascending order. Check the serial number that is physically indicated on each board to identify it.

When the `zUtils -getlabel` command is executed, specific information about the order is displayed and it might be useful to fix the order of units correctly.

- When the detection order is correct, the following information is displayed:

```
-- ZeBu : zUtils : PCIe 0 slot 05:00.0, board 0950 is detected
(remap 128MB).
-- ZeBu : zUtils : PCIe 1 slot 8a:00.0, board 0920 is detected
(remap 128MB).
-- ZeBu : zUtils : PCIe 2 slot 90:00.0, board 0940 is detected
(remap 128MB).
-- ZeBu : zUtils : PCIe 3 slot 87:00.0, board 0910 is detected
(remap 128MB).
-- ZeBu : zUtils : PCIe 4 slot 84:00.0, board 0900 is detected
(remap 128MB).
```

- When the detection order is incorrect, an error message is generated indicating how the PCIe boards must be connected to units:

```
-- ZeBu : zUtils : ERROR : LUI2058E : Bad unit detection order.
-- ZeBu : zUtils : ERROR : PCIe 0 board is normally connected to
unit 0, PCIe 1 board to unit 1, and so on.
-- ZeBu : zUtils : ERROR : You can connect PCIe 0 board to another
unit if, and only if, the other PCIe boards are connected in an
orderly way.
-- ZeBu : zUtils : ERROR : For instance:
-- ZeBu : zUtils : ERROR : - if you connect PCIe 0 to unit 1, then
PCIe 1 must be connected to unit 2, PCIe 2 to unit 3, etc.
-- ZeBu : zUtils : ERROR : - if you connect PCIe 0 to unit 2, then
PCIe 1 must be connected to unit 3, PCIe 2 to unit 4, etc.

-- ZeBu : zUtils : ERROR : The PCIe 0 board 0950 (slot 05:00.0), is
currently connected to unit 3.
-- ZeBu : zUtils : ERROR : The PCIe 1 board 0920 (slot 8a:00.0), is
currently connected to unit 2.
-- ZeBu : zUtils : ERROR : The PCIe 2 board 0940 (slot 90:00.0), is
currently connected to unit 1.
-- ZeBu : zUtils : ERROR : The PCIe 3 board 0910 (slot 87:00.0), is
currently connected to unit 4.
-- ZeBu : zUtils : ERROR : The PCIe 4 board 0900 (slot 84:00.0), is
currently connected to unit 0.
-- ZeBu : zUtils : ERROR : You should swap the cables connected to
the PCIe boards to have the proper order.
```

4.8 Viewing ZeBu Server Installation Log Files

The `zInstallLauncher.sh` uses the `zInstall` tool to generate log files at the following locations:

- `/zebu/zInstall.<i>.log` (log files)

Where, `<i>` is 0, 1, 2, 3, or 4. The most recent file is 0. Older files are shifted for each `zInstall`.

- `/var/log/messages` (system file) (`/bin/dmesg` can be used to read the last lines of `/var/log/messages`).

When the ZeBu Server unit is initialized, `zSwitchDriver` becomes a daemon and continues to run in the background.

The `/zebu/zInstall.<i>.log` files are common to all connections, regardless of the ZeBu release.

Connection or Disconnection Logs

A history file for all connection and disconnection events (for `zInstall`, `zServer`, `zUtils` and customer testbenches) is stored daily as `/zebu/zebu_<year_month_day>.log`. This log file has the details for the last 15 days of use and each tool is listed as:

```
#boardId #userName #tool #pid at #date - #time : #STATUS or
command line
```

Where, tool can be `zInstall`, `zServer`, `zUtils`, or name of a customer testbench.

Example:

Connection log progresses when `user1` uses a ZeBu Server 2 unit (board ID = 00a0) with a C++ testbench (`testbench_prd`).

1. The testbench launches **zServer**:

```
user1 testbench_prd 9733 at <date> - <time> : ** OPEN *****
user1 testbench_prd 9733 at <date> - <time> : run_zebu/
testbench_prd
user1 zServer          9747 at <date> - <time> : ** OPEN *****
user1 zServer          9747 at <date> - <time> : zServer -zebu.work
.../zcui.work/zebu.work
```

2. **zServer** requests the connection with the kernel:

```
user1 zServer          9747 at <date> - <time> : ** CONNECT *****
```

If the board is being used by another process, **zServer** waits for the board to be available (this can last several minutes).

3. **zServer** is connected to the board and the user testbench can use the board, which starts its initialization process:

```
00a0 user1 zServer          9747 at <date> - <time> : ** LOAD
***** using U2.M0
00a0 user1 testbench_prd 9733 at <date> - <time> : ** CONNECT
*****
```

4. The testbench starts running as soon as the connection with **zServer** is established. When the testbench ends, it requests disconnection from **zServer** to close the session:

```
user1 zServer          9747 at <date> - <time> : ** CLOSE *****
user1 zServer          9747 at <date> - <time> : code 0 : Server
closed correctly.
```

5. The testbench terminates correctly:

```
00a0 user1 testbench_prd 9733 at <date> - <time> : ** CLOSE
*****
00a0 user1 testbench_prd 9733 at <date> - <time> : code 0 :
Simulation finished
```

4.9 Updating PCIe Boards Firmware

The PCIe boards in the host PCs sometimes must be updated when installing a new software version or an intermediate patch. A dedicated tool, **zUpdate**, is available for updating the PCIe boards.

The Release Notes contain information on when it is necessary to update the PCIe.

Updating Only One PCIe Board in the Host PC

To update only one PCIe board in the host PC, execute the **zUpdate** command:

```
$ zUpdate
```

After the update is complete, restart the host PC.

Updating Several PCIe Boards in the Host PC

To update several PCIe boards in the host PC, an index can be specified for **zUpdate** by executing the following command:

```
$ zUpdate [<PCIeIndex>]
```

Where, <PCIeIndex> can be any integer between 0 and 4 (If <PCIeIndex> is not set, only the board with index 0 is updated).

Note

*Run the **zUpdate** command as many times as the number of available PCIe boards.*

After all the **zUpdate** commands are run, the update is complete. Make sure you restart the host PC.

5 Setting Up ZeBu Server

For the first installation of the ZeBu Server system and any hardware modification (if a unit or module has been added, changed, removed, or if the cables interconnecting units have been moved), you must launch a setup process for system calibration and tests. A dedicated tool, **zSetupSystem**, supports the complete setup process, as described in [zSetupSystem Tool](#).

The data resulting from diagnostics and calibration process is stored by **zSetupSystem** in a dedicated directory for the entire ZeBu Server system, known as the system directory. To support multiunit and multiuser features, all PCs connected to the ZeBu Server system must have Read/Write accesses to this directory.

After the setup process, the system directory includes the results of diagnostics and calibration processes and the hardware configuration file for ZeBu compilation.

Before proceeding with emulation runtime, this system directory must be defined with the `ZEBU_SYSTEM_DIR` environment variable.

The following figure shows this step in the ZeBu Server installation process.

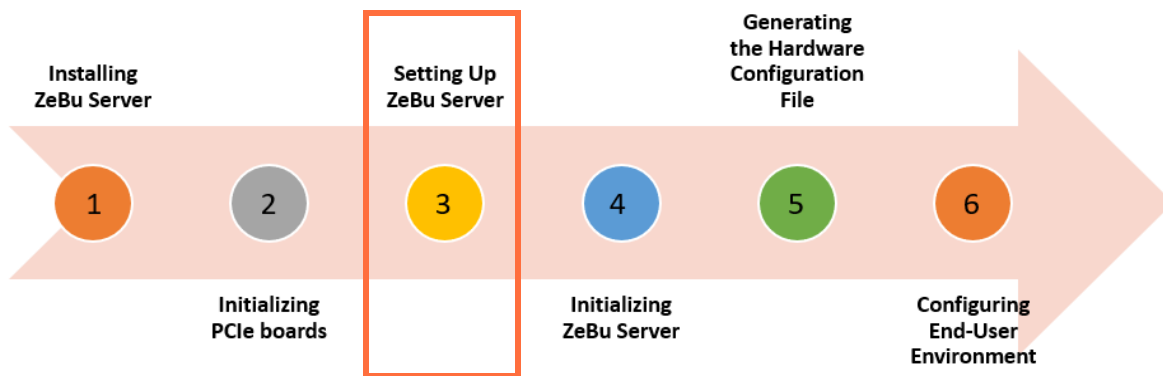


FIGURE 6. Step 3: ZeBu Server Installation Process

This section provides information about **zSetupSystem** tool and describes the following:

- [zSetupSystem Tool](#)
- [Choosing the System Directory](#)
- [Preparing the Setup File \(zini File\)](#)
- [Launching the zSetupSystem Tool](#)
- [Content of the System Directory](#)

5.1 zSetupSystem Tool

The **zSetupSystem** is a dedicated tool for:

- Launching system calibration and diagnostics
- Testing memories
- Generating the target hardware configuration file for compilation

The **zSetupSystem** integrates the following steps:

- Initialization, detection, verification, and calibration of the ZeBu Server system with **zUtils**:
 - ❑ Initializes the ZeBu Server system
 - ❑ Detects the full configuration of all modules in the system
 - ❑ Generates a `config.dff` configuration file with module IDs
 - ❑ Calibrates all inter-FPGA connections at all predefined frequencies and tests all memories
 - ❑ Generates a `status_se.tcl` file containing the test results
- Generation of the target hardware configuration file for compilation with **zConfig**:
 - ❑ Using the `status_se.tcl` file generated in the previous step, **zConfig** generates the target hardware configuration file, `zse_configuration.tcl`, and the FLLP history file (frequency-limited LVDS pairs) for the current ZeBu Server system, as described in [Generating the Target Hardware Configuration File](#).
- Generation of the host topology file for use by the Resource Management API:
 - ❑ This is mandatory for ZeBu Server 4 and optional for the other ZeBu platforms
 - ❑ This file describes the connection between the Host PCs and the ZeBu system

Choosing the System Directory

- ❑ The Resource Management API uses it to determine the possible placements of a compiled design
- Storage of system information:
 - ❑ A compressed archive, <config_directory>.tar.bz2, is stored in the directory from where **zSetupSystem** was launched (for example `my_setup_dir.tar.bz2`). This archive is intended for tracking purposes and should be sent to your Synopsys representative if requested.

5.2 Choosing the System Directory

It is mandatory to create one system directory for each ZeBu system. In addition, it is recommended to use the same system directory (`$ZEBU_SYSTEM_DIR`) for all software versions. Generate this directory using the latest major ZeBu release.

For more information, see the following subsections:

- [One Directory for All PCs](#)
- [Naming Rules](#)

Note

📄 The `ZEBU_SYSTEM_DIR` system directory must be shared between all host PCs, which are connected to the ZeBu system, and it is highly recommended to be on the same local network as the host PCs.

📄 The logs sub-directory of the `ZEBU_SYSTEM_DIR` system directory and its sub-directories must be writable by all users of the ZeBu system.

5.2.1 One Directory for All PCs

When several PCs are connected to the same ZeBu Server 2 system, a symbolic link is created to the system directory of the ZeBu Server 2 system with the name of each PC. Following is an example for PC0:

```
ln -s /usr/share/zebu/setup/203_92_81 /usr/share/zebu/setup/  
my_ZSE_on_PC0
```

With this link, setting the `$ZEBU_SYSTEM_DIR` environment variable for emulation runtime using PC0 is easier because only the name of the PC is required from user:

```
$ export ZEBU_SYSTEM_DIR=/usr/share/zebu/setup/my_ZSE_on_PC0
```

5.2.2 Naming Rules

There is no naming rule for the system directory, any path name can be used. However, it is recommended to choose an understandable name, when several nonidentical systems are available.

Example:

Using the ZeBu Server 3 system, as done by Synopsys for diagnostics patch names, provides a nonambiguous naming convention (see section [Diagnostics for ZeBu Server 3](#)). An index can be added to differentiate systems with identical configurations:

- If the `<sys_config>` information in the diagnostics patch name for your system is 203_92_81, set the following value:

```
$ZEBU_SYSTEM_DIR="/usr/share/zebu/setup/203_92_81";
```

- If you have several systems with the same `<sys_config>` information, for example 203_92_81, use A and B indexes to differentiate the 2 paths:

- For System A:

```
$ZEBU_SYSTEM_DIR="/usr/share/zebu/setup/203_92_81_A";
```

- For System B:

```
$ZEBU_SYSTEM_DIR="/usr/share/zebu/setup/203_92_81_B";
```

5.3 Preparing the Setup File (zini File)

The setup file template, `setup_template.zini`, supplied with this release is available after the software installation in the `$ZEBU_ROOT/etc/configurations/` directory. See [Appendix A: Examples of setup_template.zini](#) for an example of this file. Copy this file locally on the Host PC and then modify it manually as per your actual configuration. The `setup_template.zini` file contains helpful comments for tag modifications.

For more information, see the following subsections:

- [Declaring the System Directory and Diagnostic Patches in the zini file](#)
- [Managing Calibration History](#)
- [Testing the Memories](#)
- [Skipping Memory Tests](#)

5.3.1 Declaring the System Directory and Diagnostic Patches in the zini file

Defining the path of the system directory is mandatory in the `zini` setup file:

```
$ZEBU_SYSTEM_DIR="my_setup_dir";
```

If you have installed only one diagnostics patch corresponding to the configuration of your ZeBu Server system, this is the only mandatory declaration.

Declaring the diagnostics patches (if several are installed)

If several diagnostics patches are installed in your release, you must specify the patch that is used for the present system initialization. Modify the `$patchFile` line with the path to the appropriate patch. For example, for a 5-slot multiunit backplane with 5 9F/ICE modules, specify the following:

```
$patchFile = "/zebu/release/etc/firmwares/ZSE/dt/V6.0/XC7V-C00-00-514_91_91_91_91_91.diag";
```

5.3.2 Managing Calibration History

By default, **zSetupSystem** uses the history file, which is automatically stored in the system directory. To ignore the history, the `$noHistory` variable can be set to 1 in the `zini` setup file:

```
$noHistory = 1;
```

In the template file, this line has a comment mark for easy modification.

When a new system directory is defined in the setup file, no history file is considered.

5.3.3 Testing the Memories

By default, **zSetupSystem** tests all memories at the end of setup, after the calibration of the ZeBu Server system. This is equivalent to:

```
$ zUtils -mem all_mem
```

Information on faulty or available memories is logged in the following two files:

- `status_se.tcl` : See the `ZeBu-SE_memories` section

- `memories.dff` : See the `memories status` section

For runtime errors caused by faulty memories, see [Runtime Errors Caused by Faulty Memories](#).

For examples, see the following subsections:

- [Example of `memories.dff` for a ZeBu Server 3 system](#)
- [Example of `memories.dff` for a ZeBu Server 4 System](#)

Example of `memories.dff` for a ZeBu Server 3 system

```
$U0_M0_F00_DDR_0 = "available";
$U0_M0_F00_DDR_1 = "available";
$U0_M0_F01_DDR_0 = "available";
$U0_M0_F01_DDR_1 = "available";
$U0_M0_F06_DDR_0 = "available";
$U0_M0_F06_DDR_1 = "available";
$U0_M0_F07_DDR_0 = "available";
$U0_M0_F07_DDR_1 = "available";
$U0_M0_F08_DDR_1 = "available";
$U0_M0_FS_DDR_0 = "available";
$U0_M0_FS_DDR_1 = "available";
$U0_M1_F00_DDR_0 = "available";
...
```

Example of `memories.dff` for a ZeBu Server 4 System

```
# ===== Memories status =====

$U0_HM0_F01_DDR_0 "available"
$U0_HM0_F01_DDR_1 "available"
$U0_HM0_F02_DDR_0 "available"
$U0_HM0_F02_DDR_1 "available"
$U0_HM0_F03_DDR_0 "available"
$U0_HM0_F04_DDR_0 "available"
$U0_HM0_F04_DDR_1 "available"
$U0_HM0_FC_DDR_0 "available"
...
```

5.3.4 Skipping Memory Tests

To skip the memory tests performed at the end of setup, the `setup_template.zini` file must be modified before launching **zSetupSystem**. The following line should be uncommented:

```
#$noMemTest = 1;
```

5.4 Launching the zSetupSystem Tool

The **zSetupSystem** is controlled by a setup file, `<my_setup>.zini`, which includes information, such as path to the system directory and the firmware version of the ZeBu Server system. A template for this setup file, `setup_template.zini`, is available in the `$ZEBU_ROOT/etc/configurations` directory and described in [Preparing the Setup File \(zini File\)](#).

To start setup, use **zSetupSystem**. Make sure the license environment is setup before using **zSetupSystem**. The **zSetupSystem** command to launch the setup process is:

```
$ zSetupSystem <my_setup>.zini -platform [zs1|zs2|zs3|zs4 ] [-
topology]
```

For more information, see the following subsections:

- [Multiple Host PCs](#)
- [Creating the Host Topology](#)

Multiple Host PCs

The following configurations are possible when using multiple host PCs:

- **Single unit configuration:** In this configuration, you can run **zSetupSystem** from any host PC.
- **Multiunit configuration:** In this configuration, you must proceed in the following order:
 - a. Connect the hub to connector C4 of unit U0.
 - b. Launch **zSetupSystem** from any host PC connected to U0.

Note

ZeBu Server 3

In a multiuser system configuration (single or multi unit), if one of the Host PCs connected to any unit is switched OFF, you can no longer communicate with the units from any PC connected to the system.

*In most cases, communication resumes as soon as all PCs are switched back ON. If not, you must power OFF/ON all units then launch **zSetupSystem** (described in this section) or **zUtils -initSystem** (described in [Initializing the ZeBu Server System](#)).*

Creating the Host Topology

The **zSetupSystem** tool displays a list of questions about the connection between the Host PCs and the ZeBu Server system. For ZeBu Server 4, it is mandatory to answer all questions. However, for other ZeBu Server platforms, when the `-topology` option is used, it is optional to answer the questions.

The following snippet shows the questions to which you need to respond after **zSetupSystem** is launched.

```
-- ZeBu : zUtils :
=====
```


Content of the System Directory

```
-- ZeBu : zUtils : ==== Creation of (ZEBU_SYSTEM_DIR)/
host_topology.xml ====
-- ZeBu : zUtils :
=====

Registration of a new host:
+ Enter the name of the host pc: zebu_host05
| Registration of a new Pci connection for 'zebu_host05':
| Pci slot: 0
| Pci connector: 0
| Unit ID: 0
| Unit connector: 0
+ Do you want to register another Pci connection for 'zebu_host05'?
(y/n) n
Do you want to register another host machine? (y/n) n
-- ZeBu : zUtils : ==== (ZEBU_SYSTEM_DIR)/host_topology.xml created
=====
```

The following describes the text in bold:

- **Pci slot:** Specifies the number of the PCIe slot inside the Host PC.
- **PCI connector:** Specifies the number of the connector on the ZeBu Server PCIe board.
- **Unit ID:** Specifies the ID of the physical unit linked by this ZeBu Server PCIe board to the Host PC.
- **Unit connector:** Specifies the ID (0-4) of the connector (C0-C4) on the ZeBu Server unit used for the connection.

5.5 Content of the System Directory

The **zSetupSystem** tool generates the system directory, or updates an existing system directory. In addition, the **zSetupSystem** tool generates installation log files.

For more information, see the following subsections:

- [System Directory Files](#)
- [Subdirectories in the System Directory](#)
- [Information Log Files](#)
- [Example of Temperature log file for ZeBu Server 4](#)
- [ZeBu Runtime Log Files](#)

5.5.1 System Directory Files

The **zSetupSystem** tool generates the system directory or updates an existing directory:

TABLE 3 Files in the System Directory

File	Description
config.dff	Complete architecture of the detected ZeBu Server system
memories.dff	Memories status
status_se.tcl	System status for the design compilation process (input file for zConfig)
<my_setup>.zini	Copy of the setup file, with the same name as the zSetupSystem input file
host_topology.xml	Description of the connection between the Host PCs and the ZeBu system

5.5.2 Subdirectories in the System Directory

The system directory also includes the following subdirectories, which are generated by the **zSetupSystem** tool, or updated if they already exist:

TABLE 4 Subdirectories in the System Directory

Directory	Description
dt	Contains all input files for zUtils -calibration and zUtils -DT user tests

TABLE 4 Subdirectories in the System Directory

calibration	Contains all calibration files (except for ZeBu Server 4)
history	Contains Frequency-Limited LVDS Pairs (FLLP) history files
config	Output directory generated by zConfig .
logs/info	Contains information on all the connections and connection attempts to a ZeBu system. These daily log files are named with the current date as follows: sw_connectionInfo_YYYY_MM_DD.log (see sw_connectionInfo_YYYY_MM_DD.log) connections_YYYY_MM_DD.xml (see connections_YYYY_MM_DD.xml)
logs/srdCpu	Contains access reservation logs to hardware resources (does not apply to ZeBu Server 4) Format: srdCpu_<yyyy_mm_dd>.log
logs/connection	Contains temporary connection files
logs/run	Contains runtime log files. For more information, see ZeBu Runtime Log Files .
logs/temperature	Contains temperature log files. For more information, see ZeBu Runtime Log Files . Format: temperature_<yyyy_mm_dd>.log

5.5.3 Information Log Files

For more information, see the following subsections:

- [sw_connectionInfo_YYYY_MM_DD.log](#)
- [connections_YYYY_MM_DD.xml](#)
- [Temperature Log File](#)
- [Automatic Temperature Control Stops Emulation and Controls Overheating](#)

5.5.3.1 sw_connectionInfo_YYYY_MM_DD.log

The content of sw_connectionInfo_YYYY_MM_DD.log files in the logs/info

directory looks like the following example:

```
(1) <date> - <time> : <pid> started (<hostname> - <username> - pid
<pid>)
(2) <date> - <time> : <pid> get connection
(3) <date> - <time> : <pid> reserve : unit <unit_id> <info_reg>
(modules <Mx My>)
(4) <date> - <time> : <pid> free
(5) <date> - <time> : <pid> OK
```

Each line corresponds to a step below:

- (1) The process starts on the Host PC.
- (2) The process accesses the ZeBu Server system.
- (3) The process runs on the ZeBu Server system using the unit and module listed. There are as many such lines as units for the process. <info_reg> is a binary value that can be used by Synopsys for support purposes.
- (4) (5) The process is terminated.

This file can optionally include the size of a process design, which includes the number of FPGAs, LUTs, and registers. To view this optional information, you need a specific license feature. In addition, you need to set the `ZEBU_USE_RESOURCE_LICENSES` environment variable to ON before launching emulation runtime. The information is displayed after line (3) in the log file as follows:

```
<date> - <time> : <pid> resource : nbFpga <nb_FPGA>, nbLut
<nb_LUT>, nbReg <nb_registers>
```

When this size information is added in the `sw_connectionInfo_YYYY_MM_DD.log` file, it is also available in the `srdCpu_<date>.log` file, which is located in the `$ZEBU_SYSTEM_DIR/logs/srdCpu/` directory.

5.5.3.2 connections_YYYY_MM_DD.xml

The `connections_YYYY_MM_DD.xml` files are stored in the `logs/info` directory. These files contain information regarding connections to a specific ZeBu Server system for a specific ZeBu release. XML files are more accessible and contain machine-readable context information that allows data to be read and used by other tools.

The following tags are used in the XML file:

- `<open>`: The process starts on the host PC.
- `<reserve>`: The process accesses the ZeBu system.
- `<load>`: The process runs on the ZeBu system using the listed units and modules. There are as many lines as units for this process.
- `<close>`: The process is terminated.

An example of the `connections_YYYY_MM_DD.xml` files is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<connectionInfo date="20160418">
  <open id="58221839" date="20160418-15:23:51">
    <pc>host</pc>
    <pid>22559</pid>
    <user>username</user>
    <group>usergroup</group>
    <release>/usr/share/zebu/Q-2020.03</release>
    <systemDir>/auto/ZeBu/host</systemDir>
    <currentDir>/auto/design</currentDir>
    <project>projectname</project>
    <team>teamname</team>
  </open>
  <reserve id="58221839" date="20160418-15:23:51"/>
  <load id="58221839" date="20160418-15:23:51">
    <nbUnits>1</nbUnits>
```

```
<unit index="0">
  <modulesList nbModules="2">M0 M1</modulesList>
</unit>
</load>
<close id="58221839" date="20160418-15:24:07">
  <pc>host</pc>
  <pid>22559</pid>
  <user>username</user>
<group>usergroup</group>
  <release>/usr/share/zebu/Q-2020.03</release>
  <systemDir>/auto/ZeBu/host</systemDir>
  <currentDir>/auto/design</currentDir>
  <project>projectname</project>
  <team>teamname</team>
  <nbUnits>1</nbUnits>
  <unit index="0">
    <modulesList nbModules="2">M0 M1</modulesList>
  </unit>
  <openPendingDuration>00:00:00</openPendingDuration>
  <reservePendingDuration>00:00:02</reservePendingDuration>
  <runningDuration>00:00:14</runningDuration>
<driverClk>8.333 MHz 27,934,600</driverClk>
</close>
</connectionInfo>
```

5.5.3.3 Temperature Log File

A temperature log file, `temperature_<yyyy_mm_dd>.log`, is generated in `$ZEBU_SYSTEM_DIR/logs/directory`. The temperature log file includes the following information:

■ System event information

```
zebu_event    <run_Id> <time> <event> <type>
```

Where:

- ❑ <run_Id>: User ID for runtime
- ❑ <time>: Date/time (format: "Weekday DD MM YYYY - HH:MM:SS")
- ❑ <event>: Load or close
- ❑ <type>: TB (user testbench), IS (system init), of DT (diagnostics)

■ Temperature information

❑ Modules temperatures

```
zebu_temp <run_Id> <time> <module_phys_Id> <module_type>
<module_mfg_id> <temps>
```

Where:

- ◆ <module_phys_Id>: Specifies the module physical ID (format: "Ui Mj")
- ◆ <module_type>: Specifies the module board type, for example `zse_4c_1x330_v3`
- ◆ <module_mfg_Id>: Specifies the module manufacturing ID, such as `0x0090`
- ◆ <temps>: Specifies FPGA temperatures in Celsius (°C) (format: {Fx=TT Fy=TT ...})

❑ PCIe temperatures

```
pcie_temp <hostname> S<n> <pcie_mfg_Id> {S<n>.PE.Fx=<temp>}
pcie_max_temp <hostname> {S0.PE.Fx=<max temp> .. Sn.PE.Fx=<max temp>}
pcie_min_temp <hostname> {S0.PE.Fx=<min temp> .. Sn.PE.Fx=<min temp>}
```

Where:

- ◆ `<hostname>`: Specifies the Host PC name of the PCIe boards
- ◆ `S<n>`: Specifies the slot number of the PCIe in the Host PC
- ◆ `<pcie_mfg_Id>`: Specifies the PCIe manufacturing ID
- ◆ `S<n>.PE.FX`: Specifies the FPGA on the `S<n>` PCIe
- ◆ `<[min|max] temp>`: Specifies the temperature in degree Celsius

■ Fan speed information:

```
zebu_fan <run_Id> <time> <unit_phys_Id> <speeds>
```

Where:

- `<unit_Id>`: Specifies the physical ID fo the unit (format: `Ux`)
- `<fan_speeds>`: Specifies the fan speed coefficients for `<unit_Id>` (format: `{SP SP}`)

Example of Temperature log file for ZeBu Server 3

```
version "1.0"
```

Note: The following references are all physical.

```
zebu_event      48 "Mon 15 4 2019 - 00:00:30" load      TB
zebu_temp       48 "Mon 15 4 2019 - 00:00:30" "U0 M3"
zse_xc7v_8c_2000_v1 0x4bd2 {FC=29 FS=33 IF=26 F00=25 F01=23 F02=25 F03=24
F04=25 F05=24 F06=27 F07=26 F08=26}
zebu_fan        48 "Mon 15 4 2019 - 00:00:30" U0 {65 65}
zebu_max_temp   48 "Mon 15 4 2019 - 00:00:30" U0 {33 33}
pcie_temp       48 "Mon 15 4 2019 - 00:00:30" vgfr65zeburt38 S0
0x8552 {S0.PE.FX=41}
pcie_min_temp   48 "Mon 15 4 2019 - 00:00:30" vgfr65zeburt38
{S0.PE.Fx=30 }
pcie_max_temp   48 "Mon 15 4 2019 - 00:00:30" vgfr65zeburt38
{S0.PE.Fx=44 }
zebu_temp       48 "Mon 15 4 2019 - 00:00:31" "U0 M3"
zse_xc7v_8c_2000_v1 0x4bd2 {FC=29 FS=33 IF=26 F00=25 F01=23 F02=25 F03=24
F04=25 F05=24 F06=27 F07=26 F08=26}
zebu_fan        48 "Mon 15 4 2019 - 00:00:31" U0 {65 65}
zebu_max_temp   48 "Mon 15 4 2019 - 00:00:31" U0 {33 33}
pcie_temp       48 "Mon 15 4 2019 - 00:00:31" vgfr65zeburt38 S0
0x8552 {S0.PE.FX=40}
pcie_min_temp   48 "Mon 15 4 2019 - 00:00:31" vgfr65zeburt38
{S0.PE.Fx=30 }
pcie_max_temp   48 "Mon 15 4 2019 - 00:00:31" vgfr65zeburt38
{S0.PE.Fx=44 }
zebu_event      48 "Mon 15 4 2019 - 00:00:38" close      TB
```

Example of Temperature log file for ZeBu Server 4

```

version "1.0"

Note: The following references are all physical.

zebu_event          3 "Mon 15 4 2019 - 00:02:41" load          TB
pcie_temp            3 "Mon 15 4 2019 - 00:02:41" vgfr65zeburt67 S0
0x1055 {S0.PE.FX=47}
pcie_max_temp        3 "Mon 15 4 2019 - 00:02:41" vgfr65zeburt67
{S0.PE.Fx=52 }
zebu_temp            3 "Mon 15 4 2019 - 00:02:41" "U0 M7" orion_xcvu_6c_b_v2
0x151 {FC =28 F00=26 F01=23 F02=25 F03=22 F04=23 F05=22}
zebu_max_temp        3 "Mon 15 4 2019 - 00:02:41" "U0 M7" orion_xcvu_6c_b_v2
0x151 {FC =35 F00=27 F01=24 F02=26 F03=23 F04=24 F05=23}
pcie_temp            3 "Mon 15 4 2019 - 00:02:41" vgfr65zeburt67 S0
0x1055 {S0.PE.FX=47}
pcie_max_temp        3 "Mon 15 4 2019 - 00:02:41" vgfr65zeburt67
{S0.PE.Fx=52 }
zebu_temp            3 "Mon 15 4 2019 - 00:02:41" "U0 M7" orion_xcvu_6c_b_v2
0x151 {FC =28 F00=26 F01=23 F02=25 F03=22 F04=23 F05=22}
zebu_max_temp        3 "Mon 15 4 2019 - 00:02:41" "U0 M7" orion_xcvu_6c_b_v2
0x151 {FC =35 F00=27 F01=24 F02=26 F03=23 F04=24 F05=23}
zebu_event          3 "Mon 15 4 2019 - 00:02:43" close          TB

```

5.5.3.4 Automatic Temperature Control Stops Emulation and Controls Overheating

If the temperature of an FPGA on which the design is run exceeds 85°C (185°F), the emulation stops and the temperatures of all the FPGAs belonging to the module are displayed in Celsius degrees.

The following snippet shows an overheating error. You cannot connect to module M0 if the temperature of IF FPGA has not dropped below 85 °C (185 °F). 0x00000010 is an internal register and 0x0060 is the ID of the overheated module.

Content of the System Directory

```

Error : ===== Overheat detected (0x00000004) =====
Error : --- U0_M0 ---
Error : Type      : zse_xc7v_8c_2000_v1
Error : ID       : 0x0092
Error : M0 (physical M0) FC : 71 C
Error : M0 (physical M0) FS : 63 C
Error : M0 (physical M0) IF : 93 C
Error : M0 (physical M0) F00 : 72 C
Error : M0 (physical M0) F01 : 71 C
Error : M0 (physical M0) F02 : 75 C
Error : M0 (physical M0) F03 : 45 C
Error : M0 (physical M0) F04 : 43 C
Error : M0 (physical M0) F05 : 62 C
Error : M0 (physical M0) F06 : 68 C
Error : M0 (physical M0) F07 : 43 C
Error : M0 (physical M0) F08 : 45 C

```

Note

A temperature log file, `temperature_<yyy_mm_dd>.log`, is described in detail in [Temperature Log File](#).

5.5.4 ZeBu Runtime Log Files

The log files generated by the **zUtils** and **zInstall** tools are saved in `/zebu` local subdirectories. The runtime logs generated by testbenches and **zServer** are by default stored in the `$ZEBU_SYSTEM_DIR/logs/run` directory as listed in [Table 4](#).

The log files are named as follows:

```
#processName.#hostName.#userName.#date.log
```

Symbolic links are also available to the five most recent logs in this directory

(#processName.#hostName.#index.log).

For information pertaining to administrative tasks for ZeBu Runtime Log Files, see the following subsections:

- [Modifying the Storage Directory and Names of Runtime Log Files](#)
- [Logs for Runtime Jobs That Last Multiple Days](#)
- [Compressing and Deleting Old Runtime Log Files](#)
- [Links to Logs in the Run Directory](#)

5.5.4.1 Modifying the Storage Directory and Names of Runtime Log Files

You can set the runtime logs directory and the runtime log file names, as follows:

- Default name of the runtime logs directory (`$ZEBU_SYSTEM_DIR/logs/run/`) using the `ZEBU_LOGS_DIRECTORY` environment variable.
- Default runtime log file names using the `ZEBU_LOGS_NAME_SUFFIX` environment variable.

With both environment variables set, the path of the log file is as follows:

`$ZEBU_LOGS_DIRECTORY/#processName.$ZEBU_LOGS_NAME_SUFFIX.log`

Note

📄 When the `ZEBU_LOGS_NAME_SUFFIX` variable is set, the date in the name of the file is removed.

📄 When the `ZEBU_LOGS_DIRECTORY` and the `ZEBU_LOGS_NAME_SUFFIX` environment variables are set, the `designFeatures.help` file is also named like the other runtime log files: `$ZEBU_LOGS_DIRECTORY/designFeatures.$ZEBU_LOGS_NAME_SUFFIX.help`

5.5.4.2 Logs for Runtime Jobs That Last Multiple Days

When a runtime job lasts several days, all the information about this job is registered into several runtime log files across the log directories. These log files have a different date, which makes all the information available as expected in the archives.

Example:

A runtime job starts on Day1 and ends on Day3. Runtime logs for this job is created:

- **Day 1:** At the start of the job, the log name is
`sw_connectionInfo_YYYY_MM_Day1.log`
- **Day 2:** If information must be logged, the log name is
`sw_connectionInfo_YYYY_MM_Day1.log`
- **Day 3:** At the end of the job, the log name is
`sw_connectionInfo_YYYY_MM_Day3.log`

5.5.4.3 Compressing and Deleting Old Runtime Log Files

Old files are never deleted by the ZeBu software. They are compressed and it is your responsibility to decide whether to retain them or not.

The storage of all runtime log files can be controlled using the following environment variables:

- `ZEBU_LOGS_MAX_AGE`: Specifies the maximum age of the compressed log files. Default is 15 days.
- `ZEBU_LOGS_GZIP_CMD`: Specifies the command used for file compression. By default, `gzip` is used.
- `ZEBU_LOGS_GZIP_OPTIONS`: Specifies the `gzip` command option. By default, the `-f` option is used.

5.5.4.4 Links to Logs in the Run Directory

In the directory where the ZeBu Server run is performed, links point to the five most recent files.

6 Initializing the ZeBu Server System

You must use the **zUtils** tools to initialize the ZeBu Server system every time any unit in the system is switched OFF or ON (without any modification of the interconnection between units). To initialize the ZeBu Server system, execute the following command from any PC connected to unit U0:

```
$ export ZEBU_SYSTEM_DIR=<my_system_dir>
$ zUtils -initSystem
```

This loads the backplanes and the system FPGAs of the module.

The following figure shows the *Initializing the ZeBu Server System* step in the ZeBu Server installation process.

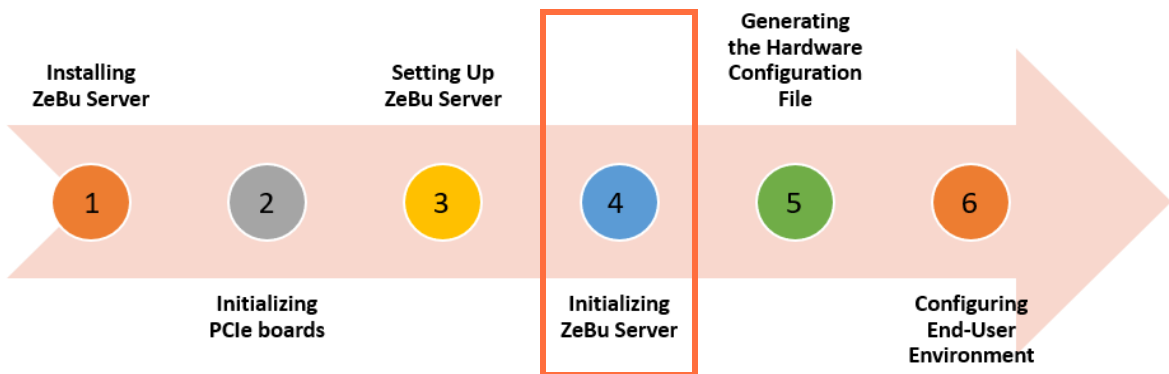


FIGURE 7. Step 4: ZeBu Server Installation Process

You must initialize the ZeBu Server system using the **zSetupSystem** tool every time a configuration change occurs in your ZeBu Server system. Modification of the interconnection between units is considered as a modification of the hardware configuration of the system. In such cases, do not use `zUtils -initSystem`. Instead you must use the **zSetupSystem** tool to consider configuration changes. By using the **zSetupSystem** tool, you can avoid runtime failure and possible hardware damages. For more information on the **zSetupSystem** tool, see [Setting Up ZeBu Server](#).

7 Generating the Target Hardware Configuration File

This section describes how to create the target hardware configuration file of a ZeBu Server system. The target hardware configuration file is a Tcl script describing the modules plugged in each unit of a ZeBu Server system. This description file is mandatory for the ZeBu Server compiler and is declared in the compilation project.

The following figure shows this step in the ZeBu Server installation process.

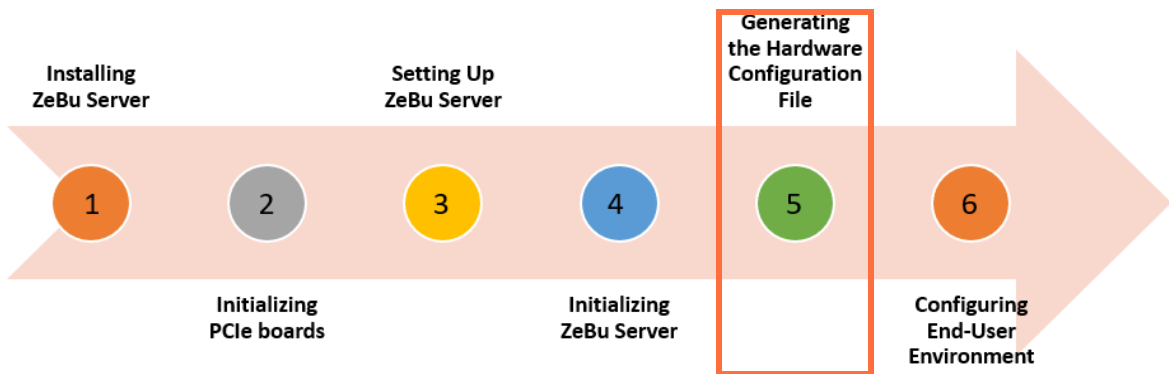


FIGURE 8. Step 5: ZeBu Server Installation Process

Overview

The following possibilities exist when generating a target hardware configuration file:

■ Generated hardware configuration file for the actual system

This file is either generated during the setup of the system, as described in [Setting Up ZeBu Server](#), or generated manually with the **zConfig** tool, as described in [Generating a Configuration File for One ZeBu Server System](#) to [Default Configuration File for a Specific Interconnection Topology](#) for specific configurations. It takes into account the diagnostics of the actual ZeBu Server system and you can use it to compile when targeting runtime emulation.

■ Example configuration file

The example configuration files listed in [Using an Example Configuration File](#) are delivered with the software release for evaluation purposes. You can compile for

tem

most of the available ZeBu Server system configurations without the actual hardware installed.

These example files can only be used for compilation.

■ **Generated default configuration file**

This file is generated manually with the **zConfig** tool to compile designs and estimate the performance in a single or multiunit environment without the actual ZeBu Server units (the diagnostics are not considered). See [Default Configuration File for a Specific Interconnection Topology](#).

This is useful for manually generating a configuration file to compile when no example hardware configuration file matches the system you want to evaluate.

These generated files can only be used for compilation.

Compiling for Several ZeBu Server Systems With zConfig

If you must compile for several ZeBu Server systems, the **zConfig** tool has some options to generate a hardware configuration file that is compatible with your systems in the following cases:

- Several ZeBu Server systems with the same units and modules, in terms of type and position (see [Hardware Configuration File for Nonidentical ZeBu Server Systems](#)).
- Several nonidentical ZeBu Server systems (see [Generating a Hardware Configuration File for Relocation](#)):

The generated hardware configuration file supports only the modules that are similar in all systems.

The hardware configuration file of a ZeBu Server system is always generated along with a frequency-limited LVDS pairs (FLLP) history file, `status_U<N>_<XXXX>.pin` (where <N> is the ID of a ZeBu Server unit and <XXXX> is the serial number of the same unit in hexadecimal format).

7.1 Generating a Configuration File for One ZeBu Server System

The target hardware configuration file for one ZeBu Server system is automatically generated during the setup of the system with the **zSetupSystem** tool, as described in [Setting Up ZeBu Server](#). In such a case, the generated file considers the diagnostics of the ZeBu Server system.

The target hardware configuration file is named `zse_configuration.tcl` and is automatically stored in the system directory (`$ZEBU_SYSTEM_DIR`). This information is an input for the ZeBu compiler in the **zCui** tool or in the Unified Compile UTF file.

7.2 Generating a Configuration File for Several ZeBu Server Systems

When targeting to use the compiled design with several ZeBu Server systems, you must manually create a single hardware configuration file with the **zConfig** tool.

Generating such a file requires that each targeted ZeBu Server system has been set up previously with the **zSetupSystem** tool as described in [Setting Up ZeBu Server](#) because the `status_se.tcl` files are required.

For more information, see the following subsections:

- [Hardware Configuration File for Identical ZeBu Server Systems](#)
- [Hardware Configuration File for Nonidentical ZeBu Server Systems](#)

Hardware Configuration File for Identical ZeBu Server Systems

If you intend to use ZeBu Server systems, which are identical in terms of units and modules (type and position in the system), the **zConfig** tool must be used with the `status_se.tcl` file generated in the system directory for each system (the command has been split on several lines for legibility):

```
$ zConfig -cfg <$ZEBU_SYSTEM_DIR_1>/status_se.tcl
          -cfg <$ZEBU_SYSTEM_DIR_2>/status_se.tcl
          -cfg <$ZEBU_SYSTEM_DIR_n>/status_se.tcl
          -o <output_path>
```

The generated hardware configuration file, `zse_configuration.tcl`, makes compilation results compatible between all `<n>` ZeBu Server systems.

Note

zConfig also generates an FLLP file for each ZeBu Server unit present in each of the `<n>` ZeBu Server systems.

Hardware Configuration File for Nonidentical ZeBu Server Systems

For ZeBu Server systems that are not identical in terms of units and modules, if you use the same syntax as in the previous section, the **zConfig** tool reports an error indicating that the merging of your systems is not possible.

In this case, use the `-mi` option (module intersection) to generate the target hardware configuration file, which is common to all of the ZeBu Server systems:

```
$ zConfig -mi
    -cfg <$ZEBU_SYSTEM_DIR_1>/status_se.tcl
    -cfg <$ZEBU_SYSTEM_DIR_2>/status_se.tcl
    -cfg <$ZEBU_SYSTEM_DIR_n>/status_se.tcl
    -o <output_path>
```

Note

When the `-mi` option is used, `zConfig` keeps in `zse_configuration.tcl` only the units and modules that are identical in all ZeBu Server systems.

7.3 Generating a Hardware Configuration File for Relocation

In a multiuser environment, relocation is possible only if the hardware configuration file considers simultaneously the diagnostics of all the FPGA modules in all systems, all the detected FLLPs.

For that purpose, you can run the **zConfig** tool with the `-merge` option:

```
$ zConfig -merge -cfg <path_0>/status_se.tcl -cfg <path_1>/
status_se.tcl -o my_config
```

Note

You can specify the `-merge` option anywhere in the command line.

7.4 Generating a Default Configuration File

To compile designs and estimate the performance in a single or multiunit environment without any diagnostics file (without the actual ZeBu Server units), you can generate a default target hardware configuration file with **zConfig**.

You can use the following options:

- **-default**: Use this option to generate a default configuration file with the list of modules in the ZeBu Server system that you want to compile.
- **-nbfpga**: Use this option to generate a default configuration file with the number of FPGAs in the ZeBu Server system that you want to compile.

In a multiunit environment, it might be useful to generate a default configuration file that matches a specific interconnection topology between the units. This is applicable for the specific topology that provides a scalable system and does not require the modification of existing cables.

For more information, see the following subsections:

- [Default Configuration File Using a List of Modules](#)
- [Default Configuration File Using an FPGA Count](#)
- [Default Configuration File for a Specific Interconnection Topology](#)

7.4.1 Default Configuration File Using a List of Modules

To generate a default hardware configuration file with the list of modules in the ZeBu Server system that you want to compile, specify the following **zConfig** command:

```
$ zConfig -default <system_config> [-mu] -o <output_path>
```

Where:

- **<output_path>**: Specifies the directory where `zse_configuration.tcl` is generated.
- **<system_config>**: Specifies the description of the ZeBu Server system:
 - ❑ 2-slot single unit system: `ab`
 - ❑ 5-slot single unit system: `abcde`
 - ❑ Multiunit system (5-slot only): `abcde_fghij_klmno_pqrst_uvwxy`

- `-mu`: (Optional) Specifies toggle to create a multiunit based file when only one 5-slot system is declared.

Each of the characters has one the following values:

TABLE 5 ZeBu Server 3 Syntax Rules

Character in Syntax	Value	Type of Module
a through y	0	No module
	8	9F
	i	9F/ICE

TABLE 6 ZeBu Server 4 Syntax Rules

Character in Syntax	Value	Type of Module
a through y	0	No module
	M	12F
	H	Hub

See the following examples:

- [Example Common to All ZeBu Server Hardware](#)
- [ZeBu Server 3 Examples](#)
- [ZeBu Server 4 Examples](#)

7.4.1.1 Example Common to All ZeBu Server Hardware

The 2-slot single-unit system with an ICE module in M0 and no module in M1:

```
$ zConfig -default i0 -o path_to_myconfig
```

7.4.1.2 ZeBu Server 3 Examples

Following are the Zebu Server 3 examples for a 5-slot system and a multiunit system:

- 5-slot system:

Generating a Default Configuration File

- ❑ To create a default configuration file for a 5-slot unit with a 9F/ICE module in M0, no module in M1 and 9F modules in M2, M3 and M4:

```
$ zConfig -zs3 -default i0888 -o path_to_myconfig
```

- ❑ To create a default configuration file for the same 5-slot unit but for a multiunit environment:

```
$ zConfig -zs3 -mu -default i0888 -o path_to_myconfig
```

- Multiunit system (4 units in these examples):

```
$ zConfig -zs3 -default i0888_iiii_i8i08_00888 -o  
path_to_myconfig
```

Where, 00888 indicates that there is no module in U3.M0 and U3.M1.

7.4.1.3 ZeBu Server 4 Examples

To create a default configuration file with 4 FPGA modules:

```
$ zConfig -zs4 -default MMMM -o path_to_myconfig
```

7.4.2 Default Configuration File Using an FPGA Count

To generate a default target hardware configuration file with the number of FPGAs in the ZeBu Server system that you want to compile, specify the following command:

```
$ zConfig -nbfpga <N> -o <output_path>
```

Where, <N> is the number of FPGAs that is generated (from 1 to 800).

Examples

- Default configuration file for 80 FPGAs:

```
$ zConfig -nbfpga 80 -o path_to_myconfig
```

This generates a configuration file for one-unit with five 17F modules (FFFFF).

- Default configuration file for 81 FPGAs:

```
$ zConfig -nbfpga 81 -o path_to_myconfig
```

This generates a configuration file for a two-unit system:

- ❑ First unit (U0) with 5 x 17f modules (FFFFFF)
- ❑ Second unit (U1) with 1 x 5F/ICE module in U1.M0 (40000)

7.4.3 Default Configuration File for a Specific Interconnection Topology

You can generate a default target hardware configuration file for multiunit systems with a specific interconnection topology. This is applicable for the specific topology that provides a scalable system not requiring the modification of existing cables.

To generate such a configuration file, `zConfig -topo` must be used simultaneously with either the `-default` option or the `-nbfgpa` option, as shown in the following:

```
$ zConfig -topo 84481-3 -default <system_config> -o <output_path>
```

or

```
$ zConfig -topo 84481-3 -nbfgpa <N> -o <output_path>
```

Where:

- `<system_config>`: Describes the ZeBu Server system with same syntax as in [Default Configuration File Using a List of Modules](#).
- `<N>`: Specifies the number of FPGAs of the ZeBu Server system (from 4 to 800 to match all possible single unit or multiunit systems).
- `<output_path>`: Specifies the directory where the `zse_configuration.tcl` file is generated.

In the present software version, only one additional interconnection topology (84481-3) is available that matches a scalable configuration. Synopsys can only update the list of supported values.

zConfig for ZeBu Server 4

To create a configuration for a single unit of four modules:

```
zConfig -zs4 -default MMMM -o cfg_single_unit
```

To create a multiunit configuration:

```
zConfig -zs4 -default MMMM_MMMM_MMMM_HH -cabling <tone-file-wiring-3U> -o cfg_3unit
```


Note

You must provide your own wiring for multiunit configurations.

An example cabling file is present at the following location:

```
${ZEBU_ROOT}/etc/sys/ORION/configs/orion_cabling*.
```

Example of Cabling Command

```
zConfig -zs4 -default MMMM_MMMM_HH -cabling ${ZEBU_ROOT}/etc/sys/ORION/configs/orion_cabling_16U_8H.tcl -o <output_dir>
```

7.5 Using an Example Configuration File

After installing the ZeBu package, some example target hardware configuration files are available in the ZeBu package. These files are located in the `$ZEBU_ROOT/etc/configurations` directory. These files can only be used for compilation.

This section describes the following:

- [ZeBu Server 3 Example Configuration Files](#)
- [ZeBu Server 4 Example Configuration Files](#)

Note

For runtime, you must generate a specific target hardware configuration file for your actual ZeBu Server system before compilation (see [Generating a Configuration File for Several ZeBu Server Systems](#) and [Generating a Default Configuration File](#)).

ZeBu Server 3 Example Configuration Files

Following is an example of ZeBu Server 3 configuration files:

TABLE 7 Available Example Configuration Files

ZeBu Server 3 System	FPGA Modules	# of FPGAs	Example Configuration File
2-slot single-unit	1x9F	9	sample_ZS3_2S_8C_2000T.tcl
"	1x9F/ICE	9	sample_ZS3_2S_ICE_2000T.tcl
"	2x9F/ICE	18	sample_ZS3_2S_ICE_ICE_2000T.tcl
	2x9F	18	sample_ZS3_2S_8C_8C_2000T.tcl
5-slot single-unit	3x9F	27	sample_ZS3_5S_8C_8C_8C_2000T.tcl
"	5x9F	45	sample_ZS3_5S_8C_8C_8C_8C_2000T.tcl
"	5x9F/ICE	45	sample_ZS3_5S_ICE_ICE_ICE_ICE_ICE_ICE_2000T.tcl
5-slot multi unit	5x9F	45	sample_ZS3_5SM_8C_8C_8C_8C_8C_2000T.tcl

ZeBu Server 4 Example Configuration Files

Following are some ZeBu Server 4 configuration files:

```
sample_ORION_16U_8H_64x12C_440.tcl
sample_ORION_2U_1H_8x12C_440.tcl
sample_ORION_4S_12C_12C_12C_12C_440.tcl
sample_ORION_4S_12C_440.tcl
sample_ORION_4S_H6C_440.tcl
```

8 Configuring the End-User Environment

This chapter describes how to configure the end-user environment. The following figure shows this step in the ZeBu Server installation process.

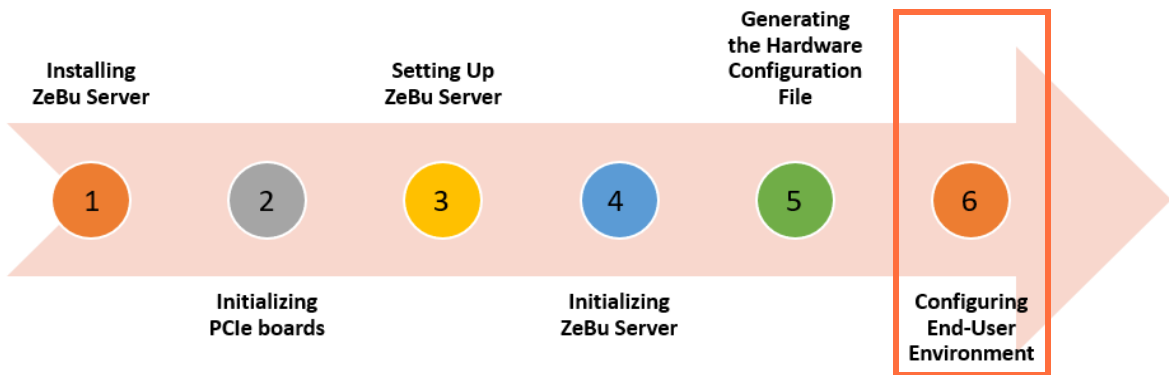


FIGURE 9. Step 5: ZeBu Server Installation Process

The administration tasks described in this section are required for every ZeBu user and consists of the following:

- [Setting Mandatory Environment Variables for ZeBu Software](#)
- [Activating a ZeBu and Xilinx License](#)
- [Declaring the System Directory for Emulation Runtime](#)
- [Runtime Settings for Relocation](#)

8.1 Setting Mandatory Environment Variables for ZeBu Software

Based on the shell type you use, you must source the `zebu_env.<shell>` file that is in the installation directory to automatically set the mandatory environment variables for the ZeBu Server compilation and runtime software.

Sourcing the `zebu_env.<shell>` file also sets the required environment variables

for Xilinx ISE and Vivado Place & Route software.

Note

It is not recommended to manually set mandatory environment variables for ZeBu.

For other third-party tools, refer to the supplier's recommendations.

8.2 Activating a ZeBu and Xilinx License

For more information on how to activate a ZeBu license, see the [Setting Up the User Environment to Access the Key File](#) section of the [Synopsys Licensing QuickStart Guide](#), available on the Synopsys website.

To activate a Xilinx license, update the `LM_LICENSE_FILE` environment variable for each end-user with appropriate information.

Following is the syntax:

■ **In bash shell:**

```
$ export LM_LICENSE_FILE=<Xilinx_portnb>@<hostname>:$LM_LICENSE_FILE
```

■ **In C shell:**

```
setenv LM_LICENSE_FILE <Xilinx_portnb>@<hostname>:$LM_LICENSE_FILE
```

Where, `<...portnb>` and `<hostname>` depend on the content of the corresponding license files.

Instead of setting the `LM_LICENSE_FILE` environment variable, it is possible to use a specific environment variable for Xilinx licenses.

Example of the syntax:

■ **In bash shell:**

```
$ export  
XILINXD_LICENSE_FILE=<Xilinx_portnb>@<hostname>:$XILINXD_LICENSE_FILE
```

■ **In C shell:**

```
setenv XILINXD_LICENSE_FILE  
<Xilinx_portnb>@<hostname>:$XILINXD_LICENSE_FILE
```

See the FLEXnet documentation for more information about setting these environment variables.

8.3 Declaring the System Directory for Emulation Runtime

Before proceeding with emulation, each user must define the system directory by setting the `ZEBU_SYSTEM_DIR` environment variable.

This variable is mandatory in the same way as `ZEBU_ROOT` is mandatory to each ZeBu Server system. Following is an example of a path setting for ZeBu Server:

```
$ export ZEBU_SYSTEM_DIR=<my_setup_directory>
```

Where, `<my_setup_directory>` is the path declared in the setup file (`<my_setup>.zini`), as described in [Declaring the System Directory and Diagnostic Patches in the zini file](#).

8.4 Enabling ZeBu Server 4 Bitstream Cache

To reduce the loading time of a design when it is run several times on the same modules, set `ZEBU_DEBUG_CACHE_BITSTREAM_DISABLE` to `false`. Therefore, the time required to initialize a design is reduced.

You can find the relevant information in the log file as follows:

```
-- ZeBu : 18:44:44.933937    39 : 269108 : zServer : NOTE : 480
Bitstreams have been programmed from cache.
-- ZeBu : 18:44:44.934998    39 : 269108 : zServer : Ux_HMx_Fxx
loading, all threads statistics : real time = 5 s, user time = 2 s,
system time = 2 s
```

For example, a design uses all modules of a 10 unit system. When the design is run without setting the preceding environmental variable, it takes 128s to load the FPGAs. When the design is run after setting `ZEBU_DEBUG_CACHE_BITSTREAM_DISABLE` to `false`, it consumes 134s in the first run and 38s for the subsequent runs.

8.5 Runtime Settings for Relocation

In a multiuser environment, you can run a design that was compiled for relocation on different subsets of FPGA modules of the ZeBu system. It is not possible to run a design on a subset of FPGA modules if the `use_module` command has not been declared during compilation.

For more information, see the following subsections:

- [Runtime Settings for Relocation on ZeBu Server 3](#)
- [Runtime Settings for Relocation With a Multiunit or Multi-PCIe System](#)
- [Runtime Settings for Relocation on ZeBu Server 4](#)
- [Runtime Settings for Speed Adapters](#)

8.5.1 Runtime Settings for Relocation on ZeBu Server 3

By default, the design runs on the subset of FPGA modules declared for compilation with `use_module` commands, generally starting from M0. To run the design on other modules of the same unit, set the `ZEBU_M0_PHYSICAL_LOCATION` environment variable before launching emulation runtime (note that 0 in M0 is character 'zero', not capital 'O'). This environment variable redefines at runtime the new physical location of the first module logically assigned during compilation.

In case of a multiunit system, the `ZEBU_M0_PHYSICAL_LOCATION` environment variable has additional constraints. You must be connected on the appropriate ZeBu Server unit, as shown in the following example. Specific recommendations for relocation with a multiunit or multi-PCIe system are available in [Runtime Settings for Relocation With a Multiunit or Multi-PCIe System](#).

Information for the possible physical locations of M0 after compilation is available in the `loc.xref` file located in the `zcui.work/zebu.work/tools/zPar/` directory. The modules that you can declare are the ones listed in the `$U0.M0.physicalLoc` line of the file.

Note

When the design has been compiled for a single module with a hardware configuration file for a 5S unit, it can be used indifferently on a 2S or 5S unit. When the design has been compiled for a single module with a hardware configuration file for a 2S unit, it can be used on a 2S or on modules M0 or M1 of a 5S unit.

Runtime Settings for Relocation

When initializing the emulation, the module declared in the `ZEBU_M0_PHYSICAL_LOCATION` environment variable is checked for coherence with the information resulting from the compilation of the design and with the actual ZeBu system. The **zServer** tool provides the following message to determine on which module the design is run:

```
-- ZeBu : zServer : The design will be relocated on physical Unit 1
Module 0.
```

Note

If the expected resources are currently used for another design, the emulation is queued by zServer (except if the `ZEBU_EXIT_IF_NO_RESOURCE_IS_AVAIL` environment variable has been previously set to "OK": the emulation stops instead of being queued).

Example

For a design that uses one module compiled on a 2-unit system with 3 modules, the following line appears in the `loc.xref` file:

```
$U0.M0.physicalLoc=" U0.M0 U0.M1 U1.M0 ";
```

This means that logical module M0 of the design can be assigned to `U0.M0`, `U0.M1` or `U1.M0`.

When emulation is launched from a PC connected to U0, you can specify either of the following:

```
$ export ZEBU_M0_PHYSICAL_LOCATION=M0
```

or

```
$ export ZEBU_M0_PHYSICAL_LOCATION=M1
```

When emulation is launched from a PC connected to U1, you can declare the following:

```
$ export ZEBU_M0_PHYSICAL_LOCATION=M0
```

8.5.2 Runtime Settings for Relocation With a Multiunit or Multi-PCIe System

When using a multiunit system with several PCIe boards in the Host PCs, you can set the `ZEBU_U0_PHYSICAL_LOCATION` environment variable to declare the ZeBu Server physical unit, which is connected as logical U0.

If this environment variable is not set, logical U0 is the unit connected to the first PCIe board detected by the BIOS of the Host PC. For more information on the BIOS detection of PCIe boards, see [PCIe Boards Detection Order](#).

Example

For a design that uses 2 modules, compiled on a 3-unit system, the following line appears in the `loc.xref` file:

```
$U0.M0.physicalLoc=" U0.M0 U1.M0 U2.M0 ";  
$U0.M1.physicalLoc=" U1.M0 U2.M0 U0.M0 ";
```

This means that the design can be assigned to M0 modules of U0 and U1, or to M0 modules of U1 and U2 or to M0 modules of U2 and U0.

Using a single PC with three PCIe boards, the design is launched by default on U0 and U1, equivalent to:

```
$ export ZEBU_U0_PHYSICAL_LOCATION=U0
```

To launch the design on U1 and U2, specify the following:

```
$ export ZEBU_U0_PHYSICAL_LOCATION=U1
```

To launch the design on U2 and U0, declare the following:

```
$ export ZEBU_U0_PHYSICAL_LOCATION=U2
```

If another host PC with only one PCIe board is connected to U0, U1 or U2, there is no need to declare `ZEBU_U0_PHYSICAL_LOCATION`. There is only one possible location in such a case; the unit connected to the particular PC.

8.5.3 Runtime Settings for Relocation on ZeBu Server 4

When using a ZeBu Server 4 to run the design on other modules of the same unit, set the `ZEBU_PHYSICAL_LOCATION` environment variable before launching emulation

Runtime Settings for Relocation

runtime. This environment variable defines the new physical location of the first module logically assigned during compilation. The location of the first module of each unit must be specified using the `ZEBU_PHYSICAL_LOCATION` environment variable as follows:

```
export ZEBU_PHYSICAL_LOCATION="U2.M0,U3.M0"
```

When a design uses only one half-module, use the following syntax:

```
export ZEBU_PHYSICAL_LOCATION="U2.HM1"
```

When initializing the emulation, the module declared in the `ZEBU_PHYSICAL_LOCATION` environment variable is checked for consistency with the information resulting from the compilation of the design and with the actual ZeBu system. The **zServer** tool provides the following message to indicate on which module the design is run:

```
-- ZeBu : zServer : Remapping required:
-- ZeBu : zServer :   U0.M0 -> U0.M1
```

Note

If the expected resources are not available, the emulation stops and is not queued.

Example:

To run your design on modules starting with M2, declare the following:

```
$ export ZEBU_PHYSICAL_LOCATION=U0.M2
```

If the design fits on one half-module, specify the following:

```
$ export ZEBU_PHYSICAL_LOCATION=U0.HM3
```

8.5.4 Runtime Settings for Speed Adapters

When using any of the following Speed Adapters with ZeBu Server 4, you need to activate the ZeBu queue during runtime.

- Ethernet Speed Adapter
- PCIe Speed Adapter

- Level Shifter Board Speed Adapter
- SMART-ZICE Speed Adapter

Otherwise, **zServer** exits with an error if the required Speed Adapter is used.

To activate the ZeBu queue during runtime, set the following environment variable:

```
export ZEBU_ACTIVATE_EMU_QUEUE=OK
```

8.5.5 Checking the Position of a ZeBu System

To check the position of a ZeBu system, use the `zRscManager` command as follows:

```
$ zRscManager -nc -sysstat $ZEBU_SYSTEM_DIR
U0.M0 used 4ef72a3f user1 host1
U0.M1 free
U0.smartZICE.C0 free
U0.smartZICE.C1 free
U0.smartZICE.C2 free
U0.smartZICE.C3 free
U0.smartZICE.C4 free
```

For more information on this command, use the following:

```
zRscManager -nc -help sysstat
```

9 Troubleshooting ZeBu Server Installation

This section provides information on installation errors that might be reported during the ZeBu Server installation. The following troubleshooting scenarios are described:

- *Missing Kernel Source Files for the Linux Kernel*
- *ZeBu Server Unit is Not Detected by the Host PC*
- *ZeBu Kernel Module Version Mismatch*
- *ZeBu Software Does Not Recognize the Hardware*
- *zInstall Accidentally Called*
- *zServer or zUtils is Running While Trying to Install a Driver*
- *ZeBu Driver is Not Installed (Versions before L-2016.06)*
- *Problems When Compiling With gcc*
- *Cannot Communicate With Units When a PC is OFF*
- *Runtime Errors Caused by Faulty Memories*
- *Synopsys Installer Does Not Install Packages*

9.1 Missing Kernel Source Files for the Linux Kernel

When launching the `zInstallLauncher.sh` script with automatic compilation of the `zKernel` module you might not have the appropriate source files of the Linux kernel. This results in the following message during installation:

```
-- ZeBu : zInstall : Extracting "zKernel" sources from " /home/
<user>/installation_directory/bin" into the local "/zebu/L-
2016.06" directory.
-- ZeBu : zInstall : Compiling "zKernel".
-- ZeBu : zInstall : ERROR : ZINS0101E : Cannot compile sources :
-- ZeBu : zInstall : ERROR : ZINS0100E : make: Entering directory
'/zebu/L-2016.06/zKernel-src'
In directory /zebu/L-2016.06/zKernel-src
Generating config file ... OK (./config-2.6.18-164.el5-x86_64.mk)
Looking for gcc ... OK
Using /usr/bin/gcc: ELF 64-bit LSB executable, AMD x86-64, version
1 (SYSV), for GNU/Linux 2.6.9, dynamically linked (uses shared
libs), for GNU/Linux 2.6.9, stripped
Using built-in specs.
Target: x86_64-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/
man --infodir=/usr/share/info --enable-shared --enable-
threads=posix --enable-checking=release --with-system-zlib --
enable-__cxa_atexit --disable-libunwind-exceptions --enable-
libgcj-multifile --enable-languages=c,c++,objc,obj-
c++,java,fortran,ada --enable-java-awt=gtk --disable-dssi --
enable-plugin --with-java-home=/usr/lib/jvm/java-1.4.2-gcj-
1.4.2.0/jre --with-cpu=generic --host=x86_64-redhat-linux
Thread model: posix
gcc version 4.4.7 Red Hat 6.6)
```

Missing Kernel Source Files for the Linux Kernel

```
Computing sources directory name ... OK
Getting the included files list ... OK
Testing remap_page_range ... FAILED (not mandatory)
Testing remap_pfn_range ... FAILED (not mandatory)
There must be either "remap_page_range" or "remap_pfn_range".
KGCC := /usr/bin/gcc
KERNEL_RELEASE := 2.6.18-164.el5
TGT_SUFFIX := -2.6.18-164.el5-x86_64
KERNEL_FAMILY_2_6 := true
KERNEL_ARCH_DIR := x86_64
make: *** [zKernel.o] Error 255
make: Leaving directory '/zebu/Q-2020.03/zKernel-src'
-- ZeBu : zInstall : 16349 close at Mon 6 1 2015 - 14:15:10.
-- ZeBu : zInstall : The zInstall daemon process pid 16349 finished
with exit code 255.
In the install.log file, you can find the following information:
Testing remap_page_range ... FAILED (not mandatory)
Testing remap_pfn_range ... FAILED (not mandatory)
There must be either "remap_page_range" or "remap_pfn_range".
[...]
make: *** [zKernel.o] Error 255
```

Solution

In such a case, you must install the Linux package with the source files of your version of the Linux kernel (typical name of package: `kernel-devel-<version>`) and relaunch `zInstallLauncher.sh` to compile the `zKernel` module (see [Initializing PCIe Boards on Host PC Running ZeBu 2017.03 or Later Releases](#)).

9.2 ZeBu Server Unit is Not Detected by the Host PC

When launching `zInstallLauncher.sh`, the following error message indicates that the PCIe interconnection board of the Host PC has not been identified:

```
-- ZeBu : zInstall : ERROR : No ZeBu board detected on system!
```

Solution

Check that the interconnection board is plugged correctly. If the error persists, send the result of the following command to Synopsys support (zebu_support@synopsys.com):

```
$ zUtils -getPCInfo
```

9.3 ZeBu Kernel Module Version Mismatch

When loading `zKernel` in the Linux kernel, you might get the following error message:

```
zKernel.yyy.o: kernel-module version mismatch
zKernel.yyy.o was compiled for kernel version yyy while this kernel
is version xxx
```

This message is generated when `zKernel` was not compiled for the right Linux kernel version. This happens, for example, when you reuse a module from an installation on another PC that has a different version of the Linux kernel.

Solution

Check the kernel version by using `'uname -r'` that reports the kernel version. The version must be equal to the `'yyy'` of the `zKernel.yyy.o` file.

You should check the availability of the Linux kernel source files and rerun `zInstallLauncher.sh`, as described in [Initializing PCIe Boards on Host PC Running ZeBu](#)

ZeBu Software Does Not Recognize the Hardware

2017.03 or Later Releases.

Note

When zKernel was compiled automatically by zInstallLauncher.sh, this error does not occur.

9.4 ZeBu Software Does Not Recognize the Hardware

If you are not using the **zSetupSystem** tool from the latest ZeBu release, the following messages can be displayed:

```
-- ZeBu : zUtils : ERROR : LUI2081E : Unit 0 is not detected.  
-- ZeBu : zUtils : ERROR : LUI2338E : Almost 1 PCIe board must be  
connected to an unit.
```

Solution

One of the reasons for this issue is that the software does not recognize the newer version of the hardware.

To resolve this mismatch, you must always use **zSetupSystem** and `zUtils - initSystem` from the latest ZeBu release.

If the error persists, contact Synopsys support.

9.5 zInstall Accidentally Called

The **zSwitchDriver** executable detects manual **zInstall** calls and rejects them with the following message:

```
zSwitchDriver ERROR : zInstall has been used to install release  
<the_release>  
zSwitchDriver ERROR : Removing release <the_release>  
zSwitchDriver ERROR : Please use zInstallLauncher.sh script to  
install ZeBu releases.
```

This message is displayed both on `stderr` and in `zSwitchDriver.0.log`.

Solution

When manually running **zInstall** on the workstation, the Device Driver Switching solution errors out. **zInstall** must not be called manually.

ZeBu releases installed on the workstation must be first cleaned and then installed by running the `zInstallLauncher.sh` script as described in [Installing ZeBu Device Drivers](#).

9.6 zServer or zUtils is Running While Trying to Install a Driver

The following error message is displayed:

```
ERROR : zServer (pid 3165) detected as running.  
ERROR : /my_path/my_release INSTALLATION ABORTED.
```

Solution

It means that a **zUtils** or **zServer** process is running while calling the following command:

```
zInstallLauncher.sh -start -r <zebu_release_path>
```

Wait for the end of **zUtils** or **zServer** execution and retry the installation command.

9.7 ZeBu Driver is Not Installed (Versions before L-2016.06)

If the driver for a ZeBu release is not installed when starting a run, you receive the following error message:

```
-- ZeBu : zServer : ERROR : LUI0962E : The daemon for your release  
is not installed, please restart "zInstall".  
  
-- ZeBu : theTest : ERROR : The connection is closed, signal 15  
(TERM : "Terminated") has been trapped..
```

Solution

Install the required driver for the release with the `zInstallLauncher.sh` script as described in section [Installing ZeBu Device Drivers](#).

Note

*Do not use **zInstall** as prompted in the error message. **zInstall** must not be called manually.*

9.8 Problems When Compiling With gcc

The `gcc` compiler is required for correct installation of the ZeBu software. If you cannot run `gcc`, first verify that a `gcc` is installed on the PC (usually, `gcc` can be found in `/usr/local/bin` or in `/usr/bin`):

```
$ ls /usr/bin/gcc* /usr/local/bin/gcc*
```

If no `gcc` can be found in the previous locations, your Linux installation is not a developer installation. Install the `gcc` from your Linux distribution.

For example, using Red Hat Linux, you can perform this operation using `gnorpm` or the `rpm` command. See the Development/Language for the required packages for `gcc`.

When `gcc` is installed, verify that your environment recognizes it. For that, you can

verify the version of gcc:

```
$ gcc -v  
gcc version 4.4.7 Red Hat 6.6
```

Note

Check the Release Notes for the gcc version that is required for your release.

If no gcc is found, verify that the \$PATH environment variable contains the directory where gcc is installed, and if it does not contain it, add it as follows:

```
$ gcc -v  
bash: gcc: command not found  
$ echo $PATH  
/usr/bin:/bin:/sbin:/usr/local/tools/simulators
```

9.9 Cannot Communicate With Units When a PC is OFF

In a multiuser system configuration (single or multi unit), if one of the Host PCs connected to any unit is switched OFF, communication might be lost with the units from any PC connected to the system. In such a case, the following error is displayed:

```
-- ZeBu : zUtils : Loading U0_BP_FC0 with          "/home/<user>/  
installation_directory/etc/firmwares/ZSE/default/  
zse_sbp_2s_fc.bit" (using CPU).  
  
..... LAU0331E : Cannot check if U0_BP_FC0 is  
done (cannot read status).  
  
-- ZeBu : zUtils : ERROR : FAILED.  
  
  
-- ZeBu : zUtils : WARNING : Retrying to load U0_BP_FC0 (1/10).  
-- ZeBu : zUtils : Loading U0_BP_FC0 with "/home/<user>/  
installation_directory/etc/firmwares/ZSE/default/  
zse_sbp_2s_fc.bit" (using CPU).  
LAU0330E : Cannot reset U0_BP_FC0 (cannot check cmd 0x00000002).  
-- ZeBu : zUtils : ERROR : FAILED.
```

Solution

In most cases, communication resumes as soon as ALL the PCs are switched ON. If not, you should power OFF/ON all units and launch **zSetupSystem** (described in [zSetupSystem Tool](#)) or `zUtils -initSystem` (described in [Initializing the ZeBu Server System](#)).

9.10 Runtime Errors Caused by Faulty Memories

A design using a memory declared as `faulty` during setup (see [Testing the Memories](#)) causes the runtime software to error out with one of the following messages, based on the memory type.

For information on these runtime errors, see the following subsections:

- [Faulty DDR2 Memory for the SRAM Trace Memory](#)
- [Faulty DDR2 and RLDRAM Memories](#)
- [Faulty Mx_FS Memory](#)
- [Faulty DDR3 Memories](#)
- [Faulty Mx_FS DDR3 Memory](#)

Faulty DDR2 Memory for the SRAM Trace Memory

```
LUI1971E: Trace Memory U0_M0_FS_DDR2_0 has been detected as  
"faulty" during setup and the design is trying to use it
```

Faulty DDR2 and RLDRAM Memories

```
LUI1969E: Memory U0_M0_F04_DDR2_0 has been detected as "faulty"  
during setup and the design is trying to use it  
LUI1970E: Memory U0_M0_F00_RLDRAM_0 has been detected as "faulty"  
during setup and the design is trying to use it
```

Faulty Mx_FS Memory

If the `Mx_FS` memory is found to be faulty when the design is being loaded, the following error messages are reported:

Runtime Errors Caused by Faulty Memories

```
Warning : Memory U0_M0_FS has been detected as "faulty":
Warning : No bitstream cache will be available for this module
Warning : No SRAM_TRACE will be available for this module

# 5- TRACE_SIZE must be modulo 1024; else, it will be rounded down

Warning : Specified TRACE_SIZE (0x.....) is not modulo 1024
Warning : TRACE_SIZE is rounded down to 0x.....
```

If the specified trace size is less than 1024, the following message is reported:

```
Warning : Specified TRACE_SIZE (0x....) is too small (MIN = 0x400)
Warning : TRACE is disabled
```

If the trace size exceeds that of the module's physical memory, the following message is reported:

```
Warning : Specified TRACE_SIZE (0x....) exceeded MAX TRACE_SIZE for
Ux_Mx module types
Warning : TRACE Size is forced to 0x.... : No bitstream cache will
be available for this Module
```

Faulty DDR3 Memories

```
LUI1969E : Memory Unit 0 Module 0 (physical Unit 0 Module 1)
(zse_xc7v_8c_2000_v1) F00 DDR 0 has been detected as "faulty"
during setup and the design trying to use it
```

Faulty Mx_FS DDR3 Memory

If the Mx_FS DDR3 memory is found to be faulty when the design is being loaded:

- Mx_FS DDR3 memory for SRAM trace (static-probes)

```
WARNING : U0_M0_FS_DDR_0 memory has been detected as "faulty" :  
WARNING : - No SRAM_TRACE will be available for this module
```

- Mx_FS DDR3 memory for bitstream cache

```
WARNING : U0_M0_FS_DDR_1 memory has been detected as "faulty" :  
WARNING : - No Cache bitstream will be available for this module
```

9.11 Synopsys Installer Does Not Install Packages

Synopsys installer does not install the packages if the following warning appears:

```
Installer: Warning Executing zebu_xilinx_vivado_2019_1_patched_vQ-2020.03_common.spf.csh failed, EST file zebu_xilinx_vivado_2019_1_patched_vQ-2020.03_common.spf is not created

Installer: Creating EST file zebu_vQ-2020.03_common.spf ...
Installer: Warning Executing zebu_vQ-2020.03_common.spf.csh failed, EST file zebu_vQ-2020.03_common.spf is not created

Installer: Creating EST file zebu_xilinx_vivado_2019_1_vQ-2020.03_common.spf ...
Installer: Warning Executing zebu_xilinx_vivado_2019_1_vQ-2020.03_common.spf.csh failed, EST file zebu_xilinx_vivado_2019_1_vQ-2020.03_common.spf is not created

Installer: Following EST file(s) have been ignored due to previous errors or invalid characters in filename

Installer:      zebu_xilinx_vivado_2019_1_patched_vQ-2020.03_common.spf
Installer:      zebu_vQ-2020.03_common.spf
Installer:      zebu_xilinx_vivado_2019_1_vQ-2020.03_common.spf
```

This error occurs because the .csh files do not exist or Unix execution rights has been set in the .csh file.

Solution

- Check if the <package_name>.csh file exists.
- Ensure that the .csh file has execution rights permissions.

10 Appendix A: Examples of setup_template.zini

This chapter provides an example of the `setup_template_zini` file for ZeBu Server 3 and ZeBu Server 4. See the following subsections:

- [Example of setup_template.zini File for ZeBu Server 3](#)
- [Example of setup_template.zini File for ZeBu Server 4](#)

Example of setup_template.zini File for ZeBu Server 3

```
# This file will be parsed with one of the following tools and
options:
# - zSetupSystem <my_setup>.zini
# - zUtils -initSystem <my_setup>.zini

#####
##### MANDATORY DECLARATIONS #####
#####

##### ZEBU_SYSTEM_DIR: output directory for the system files
generated by zUtils #####
# All these system files will be used by the ZeBu Server system at
runtime.
# The following directories will be generated (or updated if they
already exist):
#   - dt          : contains all input files for 'zUtils -
calibration' and 'zUtils -dt' user tests
#   - calibration : contains all calibration files
#   - history      : contains FLLP (Frequency-Limited LVDS Pairs) or
faulty pins history files
#   - config       : output directory generated by zConfig
# The following files will be generated (or updated if they already
exist):
#   - config.dff    : complete architecture of the detected ZeBu
Server system
#   - memories.dff  : status of each memory in the ZeBu Server
system
```

```

# - status_se.tcl : system status for the design compilation
process (input file for zConfig)

# - setup.zini      : a copy of my_setup.zini
$ZEBU_SYSTEM_DIR = "my_ZeBu_system";

#####
##### OPTIONAL DECLARATIONS #####
#####

##### Input configuration diagnostics file #####
# "patchFile": input file which defines the ZeBu Server system
configuration
# "patchPath": input path of the diagnostics directory bitstreams
# This file/directory is located right where you installed your
system's configuration patch
# A copy of patchFile will be made in $ZEBU_SYSTEM_DIR/dt/V6.0 (see
above)
# A link of the patchPath/bitstreams will be made in
$ZEBU_SYSTEM_DIR/dt/V6.0 (see above)
# Use $copyBitstreams to request a copy instead of a link (see
below)

# Example:
#$patchFile = "/zebu/release/etc/firmwares/ZSE/dt/V6.0/
patch_file.diag";
#$patchPath = "/zebu/release/etc/firmwares/ZSE/dt/V6.0/";
# NOTE 1: if $patchFile is not specified:
# zUtils will look for patchFile in the $ZEBU_ROOT/etc/firmwares/
ZSE/dt/V6.0 directory.
# This will work with ONLY ONE patchFile present in this directory
(i.e. ONLY ONE diagnostics patch was installed)

```

```
# NOTE 2: if $patchPath is not specified:
# zUtils will look for diagnostics directory bitstreams in the
$ZEBU_ROOT/etc/firmwares/ZSE/dt/V6.0 directory.

##### Input system cables configuration file #####
# This input file will only be taken into account with a 5Sx multi-
unit system configuration.
# "cables_xU.dff" is the input file which defines inter-unit system
cables connections.
# 'xU' is the number of units in the ZeBu Server system.
# This file is present in the release $ZEBU_ROOT/etc/
configurations.
# The generated config.dff file (see above) will contain a copy of
this file.
# Example for a '2-unit' ZeBu Server system configuration
#$systemCableFile = "/zebu/release/etc/configurations/
cables_2U.dff";

# NOTE: if $systemCableFile is not specified:
# zUtils will look for the input system cables configuration file
in $ZEBU_ROOT/etc/configurations/cables_xU.dff which is compatible
with patchFile

#####
##### MISC. DECLARATIONS #####
#####

# Do not launch memories tests at the end of setup:
#$noMemTest = 1;
```

```
# Copy the diagnostics bitstreams directory instead of the link:
#$copyBitstreams = 1;

# Do not launch zConfig at the end of setup:
#$zConfig = 0;

# Do not use $ZEBU_SYSTEM_DIR/history (faulty pins history)
#$noHistory = 1;

# Specify default Unit #num Smarti Z-ICE Vcc (1.5 V by default)
# See below for available power supply voltages (e.g. Unit 0)
# Select one of the following lines:
#$U0.SmartZice.Vcc = "1.5 V";
#$U0.SmartZice.Vcc = "1.8 V";
#$U0.SmartZice.Vcc = "2.5 V";
#$U0.SmartZice.Vcc = "3.3 V";

# Specify default Unit #num Module #num 8C-ICE Vcc Ios #num, 1.2 V
by default on ZS3 (Virtex7)
# See below for available power supply voltages (e.g. Unit 0 Module
0 VccIo 1 and 2)
# Select one lines for each Vcc per 8C-ICE Module:
#$U0.M0.DirectIce.VccIo_1 = "1.2 V";
#$U0.M0.DirectIce.VccIo_1 = "1.35 V";
```

```
#U0.M0.DirectIce.VccIo_1 = "1.5 V";  
#U0.M0.Directice.VccIo_1 = "1.8 V";  
#U0.M0.DirectIce.VccIo_2 = "1.2 V";  
#U0.M0.DirectIce.VccIo_2 = "1.5 V";  
#U0.M0.DirectIce.VccIo_2 = "1.35 V";  
#U0.M0.Directice.VccIo_2 = "1.8 V";
```

Example of setup_template.zini File for ZeBu Server 4

```
# This file will be parsed with the following tool and option:
# zUtils -initSystem <my_setup.ini>

#####
##### MANDATORY DECLARATIONS #####
#####

##### ZEBU_SYSTEM_DIR: output directory for the system files
generated by zUtils #####
# All these system files will be used by the ZeBu-Server system at
runtime.
# The following directories will be generated (or updated if they
already exist):
#   - dt                : contains all input files for 'zUtils -
calibration' and 'zUtils -dt' user tests
#   - calibration       : contains all calibration files
#   - history           : contains faulty pins history files
#   - config            : output directory generated by zConfig
# The following files will be generated (or updated if they already
exist):
#   - config.dff        : complete architecture of the detected ZeBu-
Server system
#   - status_se.tcl     : system status for the design compilation
process (input file for zConfig)
#   - setup.ini         : a copy of my_setup.ini

$ZEBU_SYSTEM_DIR = "/remote/fr65zebu_config/CONFIG.TD.ZEBU/ZSE/
XCVU-440_12C_0x00106";
```

```
#####  
##### OPTIONAL DECLARATIONS #####  
#####  
  
##### Input configuration diagnostics file #####  
# "patchFile": input file which defines the ZeBu-Server system  
configuration  
# "patchPath": input path of the diagnostics directory bitstreams  
# This file/directory is located right where you installed your  
system's configuration patch  
# A copy of patchFile will be made in $ZEBU_SYSTEM_DIR/dt/V2.0 (see  
above)  
# A link of the patchPath/bitstreams will be made in  
$ZEBU_SYSTEM_DIR/dt/V2.0 (see above)  
# Use $copyBitstreams to request a copy instead of a link (see  
below)  
  
# Example:  
#$patchFile = "$ZEBU_ROOT/etc/firmwares/ZSE/dt/V5.6/XC6V-  
103_53.diag";  
  
# NOTE 1: if $patchFile is not specified:  
# zUtils will look for patchFile in the $ZEBU_ROOT/etc/firmwares/  
ZSE/dt/V2.0 directory.  
# This will work with ONLY ONE patchFile present in this directory  
(i.e. ONLY ONE diagnostics patch was installed)
```



```

# NOTE 2: if $patchPath is not specified:
# zUtils will look for diagnostics directory bitstreams in the
$ZEBU_ROOT/etc/firmwares/ZSE/dt/V2.0 directory.

##### Input system cables configuration file #####
# This input file will only be taken into account with a 5Sx multi-
unit system configuration.
# "cables_xU.dff" is the input file which defines inter-unit system
cables connections.
# 'xU' is the number of units in the ZeBu-Server system.
# This file is present in the release $ZEBU_ROOT/etc/
configurations.
# The generated config.dff file (see above) will contain a copy of
this file.

# Example for a '2-unit' ZeBu-Server system configuration
# $systemCableFile = "/zebu/release/etc/configurations/
cables_2U.dff";

# NOTE: if $systemCableFile is not specified:
# zUtils will look for the input system cables configuration file
in $ZEBU_ROOT/etc/configurations/cables_xU.dff which is compatible
with patchFile

#####
##### MISC. DECLARATIONS #####
#####
# Copy the diagnostics bitstreams directory instead of the link:
#$copyBitstreams = 1;
# Do not launch zConfig at the end of setup:
#$zConfig = 0;

```

```
$dt = 0;  
$calibration = 0;  
$noMemTest = 1;  
  
# Do not use $ZEBU_SYSTEM_DIR/history (faulty pins history)  
#$noHistory = 1;  
  
$TRACE_SIZE = 0x4000000;
```

11 Appendix B: Getting Board Labels

If a Synopsys representative requests the board label information, for maintenance purposes, you can obtain the board labels of your ZeBu Server system:

```
zUtils -getLabel <labelBoardName> <labelFileName>
```

Where, <labelBoardName> is any of the following boards:

PC	Hub	Unit U0	Unit U1	Unit U2	Unit U3	Unit U4
S0_PCIe	HUB	U0_BP	U1_BP	U2_BP	U3_BP	U4_BP
S1_PCIe		U0_FP	U1_FP	U2_FP	U3_FP	U4_FP
S2_PCIe		U0_M0	U1_M0	U2_M0	U3_M0	U4_M0
S3_PCIe		U0_M1	U1_M1	U2_M1	U3_M1	U4_M1
S4_PCIe		U0_M2	U1_M2	U2_M2	U3_M2	U4_M2
		U0_M3	U1_M3	U2_M3	U3_M3	U4_M3
		U0_M4	U1_M4	U2_M4	U3_M4	U4_M4

- Where BP stands for backplane and FP for front panel.
If <labelBoardName> is not specified, all labels are displayed on screen.
- <labelFileName> is a file where the E2PROM content of the specified board is stored. If <labelFileName> is not specified, the board label is displayed on screen.

Note

To get the label of a PCIe board plugged in a given host PC, you must be connected to the same host PC. You cannot get the labels of other PCIe boards or even know what other PCs are connected to the system.

The output depends on whether the host PC from which the command was issued is connected to unit U0:

- If the host PC is connected to U0:
- Output = Full label (for any board). Example for ZeBu Server 1:

```
-- ZeBu : zUtils : ==== Unit 0 Module 0 label ====  
$labelVersion = 2;  
$boardType    = "zse_xc6v_4c_ice_lx760_v3";  
$initDay      = "2-2-2015";  
$id           = 0x0102;  
$revPCB       = "3.0";  
$revBOM       = "2.0";  
$features     = 0xffffffff;
```

- If the host PC is not connected to U0:
 - ❑ Output = Full label (for the PCIe board plugged in the current host PC).
 - ❑ Output = Short label (for any other board). Example for ZeBu Server 1:

```
-- ZeBu : zUtils : ==== Unit 4 Module 0 (physical Unit 0 Module 0)  
label ====  
$boardType    = "zse_xc6v_4c_ice_lx760_v3";  
$id           = 0x0102;
```