ZeBu® Server Administration Guide

Version V-2024.03-1, July 2024

SYNOPSYS®

Copyright and Proprietary Information Notice

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 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

Feedback

Contents

Preface

This chapter has the following sections:

About This Book

This guide contains information on administration tasks for ZeBu Server 4. It describes the procedure 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 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 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.

Feedback

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	The*zSetupSystem* tool for completing the ZeBu Server setup process
Initializing the ZeBu Server System	Initialization of ZeBu Server with the *zSetupSystem* 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 setup_template_zs4.zini	The *setup_template_zs4.zini* file for each ZeBu Server
Appendix B: Getting Board Labels	Board labels of ZeBu Server

Hardware Documentation

Document Name	Description
ZeBu Getting Started Guide	Provides brief information about Synopsys' emulation system - ZeBu.
ZeBu Power Estimation User Guide	Provides the power estimation flow and the tools required to estimate the power on a System on a Chip (SoC) in emulation.

Feedback

Document Name	Description
ZeBu Server 4 Smart Z-ICE Interface User Guide	Provides physical description of the Smart Z-ICE interface and the steps to instantiate and use it on ZeBu Server 4.
ZeBu Server 4 Release Notes	Provides enhancements and limitations for new ZeBu Server 4 releases.

Related Documentation

Document Name	Description
ZeBu Debug Guide	Provides information on tools you can use for debugging.
ZeBu Debug Methodology Guide	Provides debug methodologies that you can use for debugging.
ZeBu Unified Command-Line User Guide	Provides the usage of Unified Command-Line Interface (UCLI) for debugging your design.
ZeBu UTF Reference Guide	Describes Unified Tcl Format (UTF) commands used with ZeBu.
ZeBu Power Aware Verification User Guide	Describes how to use Power Aware verification in ZeBu environment, from the source files to runtime.
ZeBu Functional Coverage User Guide	Describes collecting functional coverage in emulation.
Simulation Acceleration User Guide	Provides information on how to use Simulation Acceleration to enable cosimulating SystemVerilog testbenches with the DUT
ZeBu Verdi Integration Guide	Provides Verdi features that you can use with ZeBu. This document is available in the Verdi documentation set.
ZeBu Runtime Performance Analysis With zTune User Guide	Provides information about runtime emulation performance analysis with zTune.
ZeBu Custom DPI Based Transactors User Guide	Describes ZEMI-3 that enables writing transactors for functional testing of a design.
ZeBu LCA Features Guide	Provides a list of Limited Customer Availability (LCA) features available with ZeBu.
ZeBu Transactors Compilation Application Note	Provides detailed steps to instantiate and compile a ZeBu transactor.
ZeBu zManualPartitioner Application Note	Describes the zManualPartitioner feature for ZeBu. It is a graphical interface to manually partition a design.

Document Name	Description
ZeBu Hybrid Emulation Application Note	Provides an overview of the hybrid emulation solution and its components.

Other Useful References

Document Name	Description
SpyGlass Power Estimation and Rules Reference Guide and SpyGlass Power Estimation Methodology Guide	Provides information about performing low-power design implementation and verification with SpyGlass.

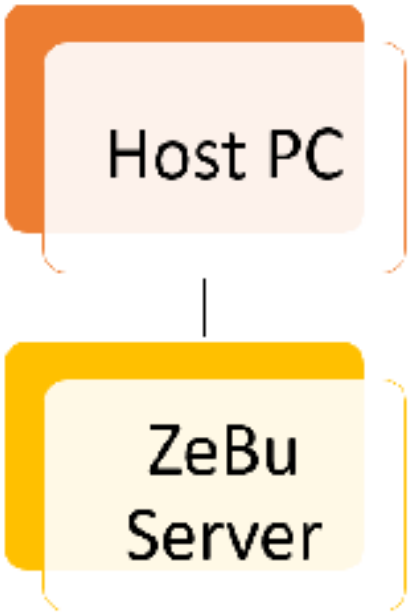
Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1

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.



Administration tasks are performed on the Host PC. The ZeBu Server software is installed here.

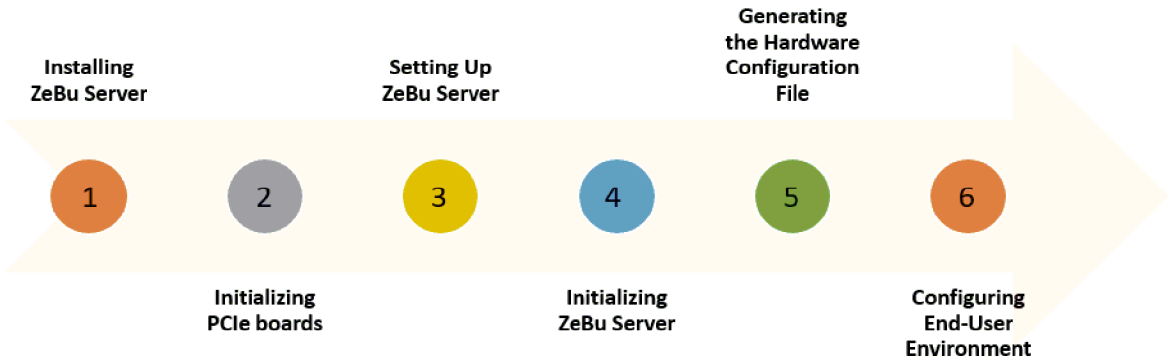
The ZeBu Server hardware runs the emulation.

For more information, see the following sections:

- ZeBu Server Installation Process
- Hardware and Software Requirements for the Host PC
- Hardware and Software Requirements for the Compilation PC

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, seeInstalling 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.
- *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.

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 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:

This chapter has the following sections:

- License Software Compatibility
- Infrastructure Requirement for ZeBu Runtime
- Synopsys Interoperable Technologies
- Third-Party Tools
- Managing Emulation Resources
- Hardware and Software Requirements for the Compilation PC

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.

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 Vivado (see Table in *Third-Party Tools*) 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 Vivado supported by this ZeBu Server release, see *ZeBu Release Notes*.

Xilinx License Software Compatibility

The Xilinx license software runs on the following operating systems:

- 32-bit Linux
- 64-bit Linux

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.

Infrastructure Requirement for ZeBu Runtime

All host servers connected to ZeBu emulators that are used as runtime hosts must have their local time synchronized on the network using the Network Time Protocol (NTP) or equivalent service.

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 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
Verdi3	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.

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 4	Xilinx tools for FPGA Place & Route during ZeBu compilation.*http://www.xilinx.com*

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

FPGA Synthesizers

ZeBu includes its own FPGA synthesizer.

Xilinx Place & Route Software

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

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

Managing Emulation Resources

ZeBu runtime software and `zRscManager` depend on the host machine's time and date for managing emulation resources, logging reliable data, and providing capabilities, such as multiuser and multihost. Therefore, for all runtime host machines accessing ZeBu emulators, you must synchronize the runtime host machine with the same Network Time Protocol (NTP) server and NTP services.

Hardware and Software Requirements for the Compilation PC

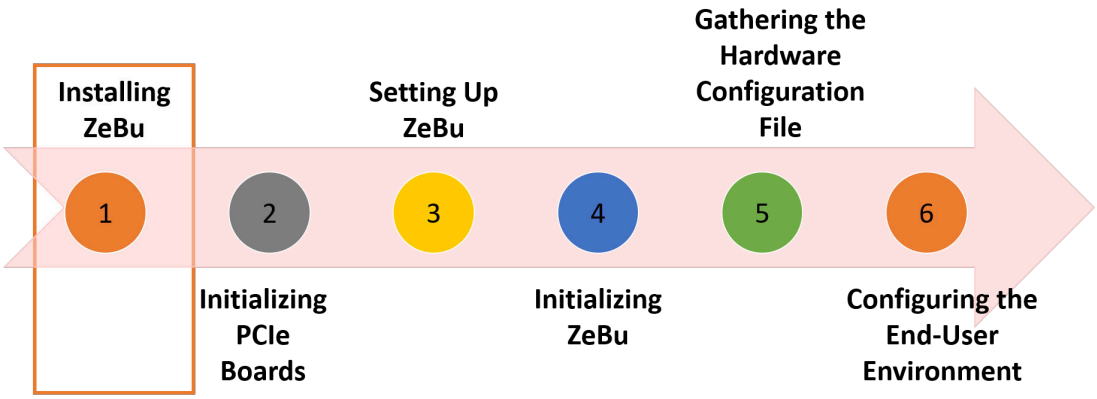
To avoid random crashes, make sure you set the stack size to unlimited when compiling for ZeBu using zCui. This applies to all hosts of the grid that are used to compile. Typically, your IT department sets the stack size to unlimited. However, in case it is not set, you can use the `ulimit` Bash built-in command or the `limit` Csh built-in command. For more details, see Setting the Stack Size for Compilation.

2

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.



This section provides information on the following:

- Downloading Packages for Installation
- Installing the ZeBu Server Software
- Downloading Required Interoperable Technologies
- 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.

Downloading Packages for Installation

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

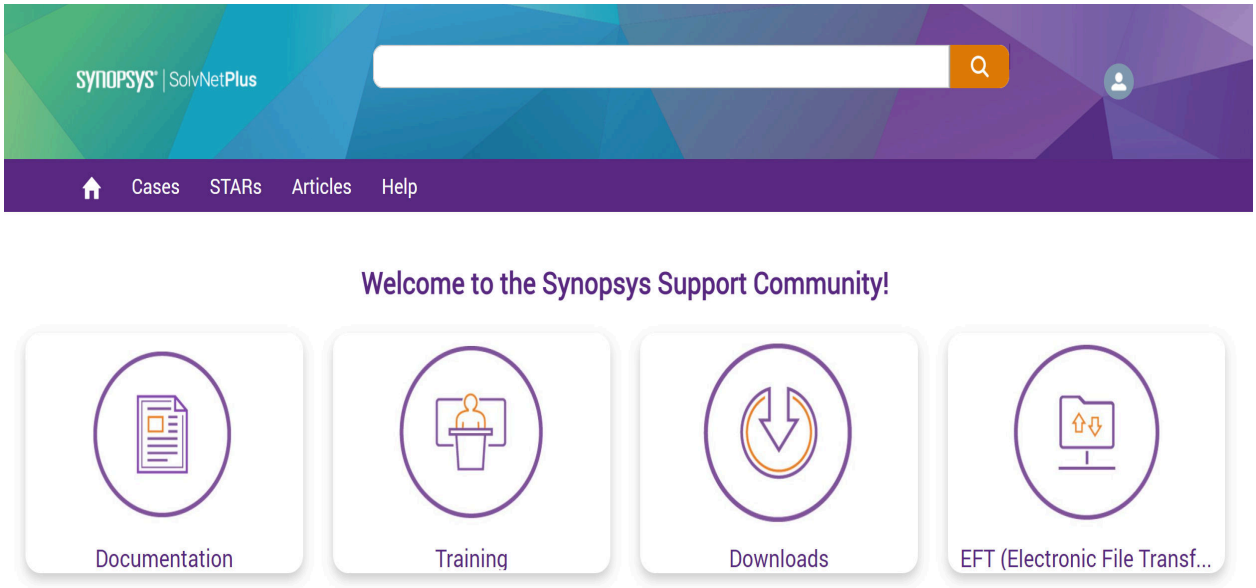
For more information, see the following subsections:

- Downloading and Extracting the Synopsys Installer
- Downloading the ZeBu Software Package

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 SolvNetPlus home page.



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_v5.7.run` containing the **Synopsys Installer** software to a dedicated directory. For example, `/home/<user>/install_files`. For more details, see Synopsys Installer on SolvNetPlus.
7. After the file is downloaded, execute the following command to extract its content:

   ```
   $ cd /home/<user>/install_files
   $ SynopsysInstaller_v5.7.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.

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.
3. From the **Downloads** list, choose **ZeBu Server 4 > V-2024.03-1**.
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 all files required for the installation of the ZeBu software release:
   - `zebu_INSTALL_README.txt`
   - `zebu_vV-2024.03-1_linux64.spf`
   - `zebu_vV-2024.03-1_common.spf` (2 parts)

- `zebu_vV-2024.03-1_zs4.spf` (5 parts)

  Note:

  `zebu_vV-2024.03-1_common.spf`, and `zebu_vV-2024.03-1_zs4.spf` files are split due to size limitations.

7. Download the following files containing the Xilinx Place & Route software based on your hardware:
   - Xilinx Vivado Place & Route for ZeBu Server 4:
     `zebu_xilinx_vivado_2022_1_patched18_vV-2024.03-1_common.spf`

     Note:

     This file has been split into 7 parts due to size limitations.

   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).
8. Download the **ZeBu Device Driver Switching Solution** package that contains the following drivers:
   - For ZeBu Server 4, see SolvNetPlus article *Instructions to Download the ZeBu Device Drivers Package for ZeBu Server 4*

For more information, see Initializing the PCIe Boards.

Downloading Required Interoperable Technologies

ZeBu requires specific Synopsys interoperable technologies.

VCS

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

- `vcs_vV-2023.12-<patch>_common.spf`
- `vcs_vV-2023.12-<patch>_linux.spf`
- `vcs_vV-2023.12-<patch>_linux64.spf`

For supported VCS version, see *ZeBu Release Notes*.

Verdi

To use enhanced debugging with ZeBu, 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_vV-2023.12-1_common.spf`
- `verdi_vV-2023.12-1_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.

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

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-4 Emulation** product (marked as **STAND-ALONE**).
   b. Select the Xilinx Place & Route software based on your hardware (marked as **OVERLAY**):

      For ZeBu Server 4: **Xilinx Vivado 2022.1_patched18 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.

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).

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[]:
```

Feedback

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

```
Select Synopsys product(s) to install:
          [ 1] zebu - ZeBu (R) Emulation
           [ 2] zebu_xilinx_vivado_2022_1 - Xilinx Vivado 2022.1 for
 ZeBu (R)  (OVERLAY)
...
                     [ b] back - Back to Select Another Version
```

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) Emulation
...
```

Post Installation Activities

The installation directory (`/usr/share/zebu/V-2024.03-1/` 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.

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.

Installing Licenses

ZeBu requires the following FLEXnet licenses:

- For the ZeBu software license:
  - The FLEXnet license daemon (`lmgrd`)
  - The Synopsys vendor license daemon (`snpslmd`)
  - The license file
- For the Xilinx Vivado Place & Route software license:
  - The FLEXnet license daemon (`lmgrd`)
  - The Xilinx vendor license daemon (`xilinxd`)
  - 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

Installing ZeBu and Xilinx Licenses

This section describes the following:

- Installing ZeBu License Server
- Installing Xilinx License Server

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.

Feedback

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_p2/bin/unwrapped/lnx64.o/xilinxd
```

If you use vivado during FPGA P&R, the following warning message appears:

```
WARNING: [Common 17-348] Failed to get the license for feature
 'Implementation' and/or device 'xcvu440'. Explanation: Your license
 server is running an obsolete version of the xilinxd licensing daemon
 and needs to be upgraded to the current version.
Resolution: Check the status of your licenses in the Vivado License
 Manager. For debug help search Xilinx Support for "Licensing FAQ".
12 Infos, 19 Warnings, 100 Critical Warnings and 1 Errors encountered.
opt_design failed
ERROR: [Common 17-345] A valid license was not found for feature
 'Implementation' and/or device 'xcvu440'. Please run the Vivado License
 Manager for assistance in determining which features and devices are
 licensed for your system.
```

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

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

Modifying License Files

This section describes the following:

- Modifying the ZeBu License File
- Modifying the Xilinx License File

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*).

Feedback

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
```

Starting the License Server

This section describes the following:

- Starting the ZeBu License Server
- Starting the Xilinx License Server

Feedback

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.

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`):

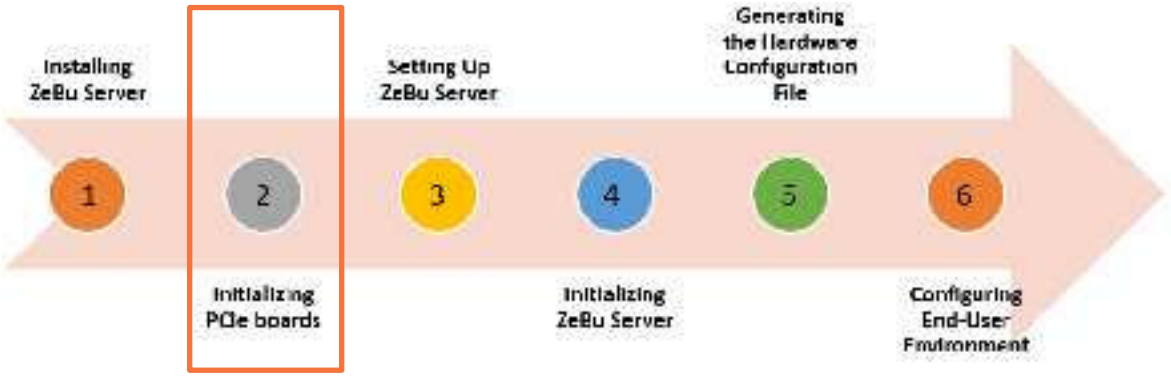
```
lmstat - Copyright (c) 1989-2006 Macrovision Europe Ltd. and/or
 Macrovision Corporation. All Rights Reserved.
Flexible License Manager status on Mon 11/3/2013 19:21
License server status: 2100@myHost
    License file(s) on
 myHost: /remote/license/data/Xilinx/myHost_Xilinx.lic:
      myHost: license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

3

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 1 Step 2: ZeBu Server Installation Process



This section has the following subsections:

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. This ZeBu device driver is part of the ZeBu drivers package and is available on SolvNetPlus.

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 following procedure only applies

if the Host PC runs only releases newer than ZeBu 2017.03 (included). If this is not the case, contact Synopsys support.

The latest version of the driver must be installed on the host PC, as described in Initializing PCIe Boards on Host PC.

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.

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 SolvNetPlus:

  - Instructions to Download the ZeBu Device Drivers' Package for ZeBu Server 4

- You must install the Linux package with the source files of your Linux kernel version (the typical name of this package is: `kernel-level-<version>`
- If you have several host PCs in your configuration, the initialization step described in Initializing PCIe Boards on Host PC must be performed on each host PC.
- The ZeBu Device Drivers package is available as a `.spf` package, which must be uncompressed using the Synopsys Installer as described in Installing ZeBu Server Software.

Initializing PCIe Boards on Host PC

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 the `drivers/zebu_dd` for ZeBu Server 4.

Feedback

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

Installing the ZeBu Device Driver on the Host PC

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

- For ZeBu Server 4, execute the `zDriverInstall install` script available at the following location:

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

  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_dd`, 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, perform the following steps:

- For ZeBu Server 4, execute the code `zDriverInstall remove` script available at the following location:

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

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.

   For ZeBu Server 4, navigate to `<driver_package_path>/drivers/zebu_dd/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
   ```

   For ZeBu Server 4, execute the following command:

   ```
   <driver_package_path>/drivers/zebu_dd/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.

Viewing ZeBu Server Installation Log Files

Logs of Connection or Disconnection

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 4 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
```

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.

Feedback

This section has the following topics:

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.

4

Initializing the ZeBu Server System

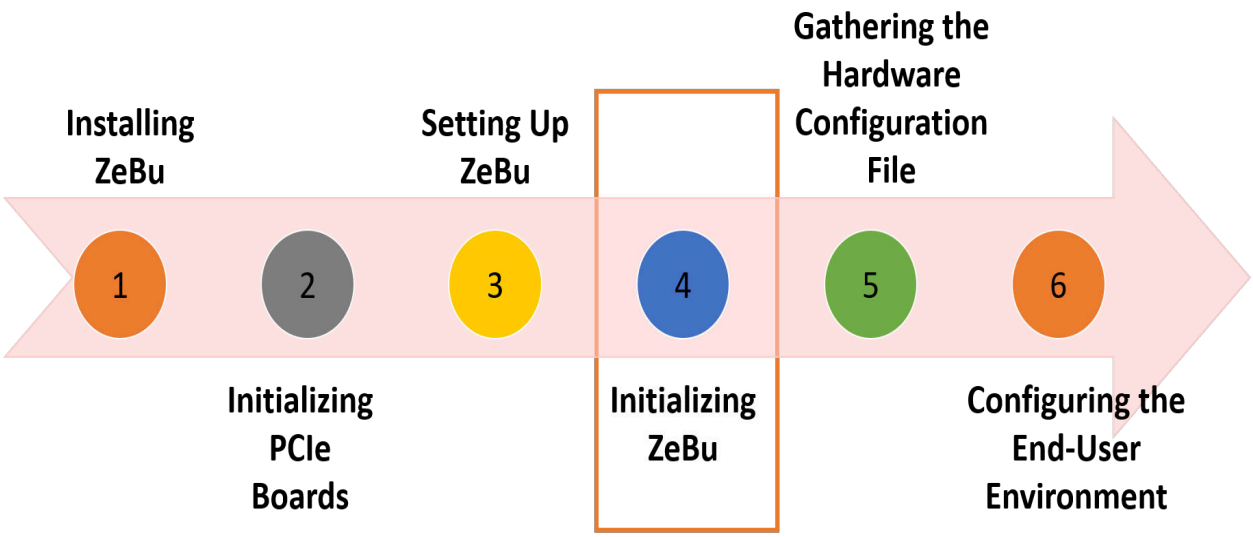
You must use the `zUtils` tools to initialize the ZeBu Server system every time any unit in the system is switched OFF or ON (without any modification of the interconnection between units). To initialize the ZeBu Server system, execute the following command from any PC connected to unit `U0`:

```
$ 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 2 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.

5

Setting Up ZeBu

For the first installation of the ZeBu system and after 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.

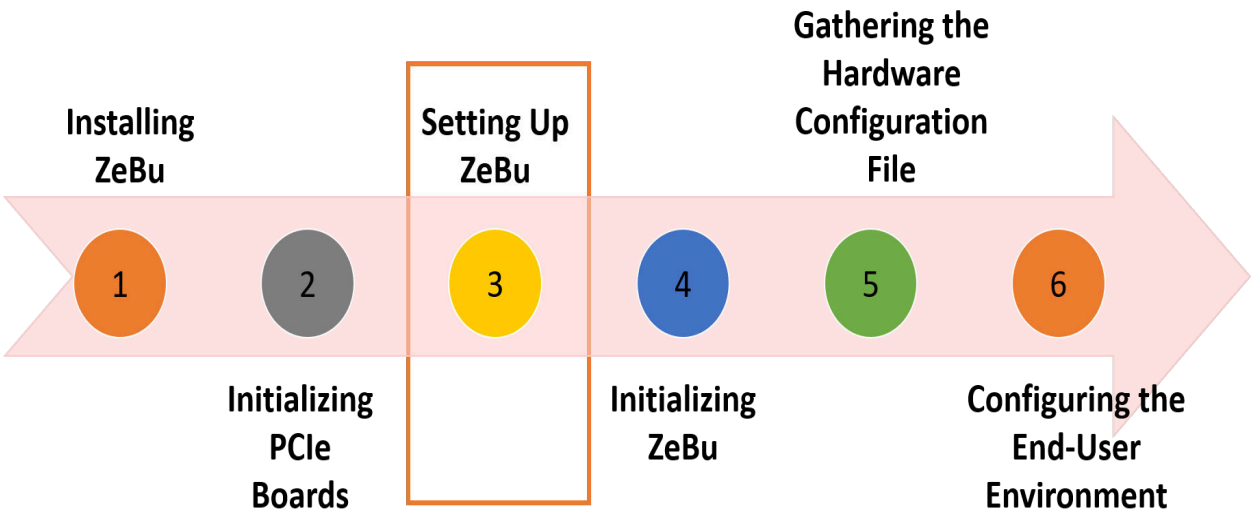
The data resulting from diagnostics and calibration process is stored by `zSetupSystem` in a dedicated directory for the entire ZeBu system, known as the ZeBu System Directory. To support multiunit and multiuser features, all PCs connected to the ZeBu system must have Read/Write accesses to this directory.

After the setup process, the system directory includes the results of diagnostics 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 installation process.

Figure 3 Step 3: ZeBu Installation Process



Feedback

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

zSetupSystem Tool

The `zSetupSystem` is a dedicated tool for:

- Creating and launching system diagnostics
- Generating the target hardware configuration file for compilation

The `zSetupSystem` integrates the following steps:

- Initialization, detection, modification, and verification of the ZeBu system with `zUtils`:
  - Initializes the ZeBu system
  - Detects the full configuration and cabling of all modules in the system
  - Generates a `config.dff` configuration file with module IDs
- Generation of the bitstreams required for the diagnostics of the ZeBu system
- Running diagnostics
- Generation of 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, for the current ZeBu system, as described in Generating the Target Hardware Configuration File.
- Generation of the host topology file for use by the Resource Management API:
  - This file describes the connection between the Host PCs and the ZeBu system
  - 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.

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.

One Directory for All PCs

When several PCs are connected to the same ZeBu system, Synopsys recommends creating a symbolic link to the system directory of the ZeBu 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
```

Naming Rules

There is no naming rule for the system directory, any path name can be used. However, when several non-identical systems are available, it is recommended to choose a significant name.

Feedback

Preparing the Setup File (zini File)

The template for the setup file, `setup_template_zs4.zini`, is available after the software installation in the `$ZEBU_ROOT/etc/configurations/` directory. See Appendix A: Examples of setup_template_zs4.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_zs4.zini` file contains helpful comments for tag modifications.

For more information, see the following subsections:

- Declaring the System Directory in the zini file
- Testing the Memories
- Skipping Memory Tests

Declaring the System Directory in the zini file

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

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

This is the only mandatory declaration.

Testing the Memories

By default, **zSetupSystem** tests all memories at the end of setup, after the calibration of the ZeBu 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_memories* section

Skipping Memory Tests

To skip the memory tests performed at the end of setup, the `setup_template_zs4.zini` file must be modified before launching **zSetupSystem**. The following line should be uncommented:

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

Feedback

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. A template for this setup file, *setup_template_zs4.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 zs4 [-topology]
```

For more information, see Creating the Host Topology

Creating the Host Topology

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

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

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

- *PCIe slot*: Specifies the number of the PCIe slot inside the Host PC.
- *PCI connector*: Specifies the number of the connector on the ZeBu PCIe board.

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

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
- ZeBu Runtime Log Files

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 system
`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
`system_config.dff`	

Feedback

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
`bitstreams_dt`	Contains bitstreams for diagnostics
`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/connection`	Contains temporary connection files
`logs/srdCpu`	Contains access reservation logs to hardware resources Format: `srdCpu_<yyyy_mm_dd>.log`
`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`

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

Feedback

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>)
```

Each line corresponds to a step below:

- (1) The process starts on the Host PC.

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.

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 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-88858-1"?>
<connectionInfo date="20160418">
  <open id="58221839" date="20160418-15:23:51">
    <pc>host</pc>
    <pid>22559</pid>
    <user>username</user>
    <group>usergroup</group>
    <release>/path/to/release/Q-2020.03-SP1/</release>
```

Feedback

```
    <systemDir>/path/to/zebu_system_dir</systemDir>
    <currentDir>/path/to/rundir/</currentDir>
    </open>
  <reserve id="7EF15174" date="20160418-15:23:51">
    <utcreserveDate>20160418-13:23:51</utcreserveDate>
  </reserve>
  <close id="58221839" date="20160418-15:24:07">
    <utccloseDate>20160418-13:24:07</utccloseDate>
    <pc>host</pc>
    <pid>22559</pid>
    <user>username</user>
 <group>usergroup</group>
    <release>/path/to/release/Q-2020.03-SP1/</release>
    <systemDir>/path/to/zebu_system_dir</systemDir>
    <currentDir>/path/to/rundir</currentDir>
    <dutDir>/path/to/zcui.work/backend_defa
    <nbUnits>1</nbUnits>
    <nbDutFpga>1</nbDutFpga>
    <unit index="0">
      <modulesList nbModules="1">M0.S0</modulesList>
    </unit>
    <openPendingDuration>00:00:00</openPendingDuration>
    <reservePendingDuration>00:00:02</reservePendingDuration>
    <runningDuration>00:00:14</runningDuration>
    <clocks>
      <driverClock>
        <zTime>
          <defaultFreqKHZ>10000</defaultFreqKHZ>
        </zTime>
      </driverClock>
    </clocks>
  </close>
</connectionInfo>
```

Temperature Log File

A temperature log file, `temperature_<yyyy_mm_dd>`.log, is generated in `$ZEBU_SYSTEM_DIR/logs/temperaturedirectory`. 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)

Feedback

- **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_lx330_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 of the unit (format: `Ux`)
  - `<fan_speeds>`: Specifies the fan speed coefficients for `<unit_Id>` (format: `{SP SP}`)

Feedback

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.

```
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.

ZeBu Runtime Log Files

The log files generated by the **zUtils** and **zInstall** tools are saved in */zebu* local subdirectories.

The log files generated by testbenches are stored in the `$ZEBU_SYSTEM_DIR/logs/run` directory by default.

Each emulation run session creates the following log files:

- `zServer.<hostname>.<username>.<timestamp>.log`
- `profiler.<hostname>.<username>.<timestamp>.log`
- `zLicenseManager.<hostname>.<username>.<timestamp>.log`

Feedback

In addition, one log file for each testbench process is created and named using the following pattern:

```
<processName>.<hostname>.<username>.<timestamp>.log
```

Symbolic links to the logs of five most recent run sessions are also available in the directory where the emulation is started. The names of these symbolic links use the following patterns:

- `zServer.<hostname>.<index>.log`
- `profiler.<hostname>.<index>.log`
- `zLicenseManager.<hostname>.<index>.log`
- `<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

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`

Feedback

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`

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.

Links to Logs in the Run Directory

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

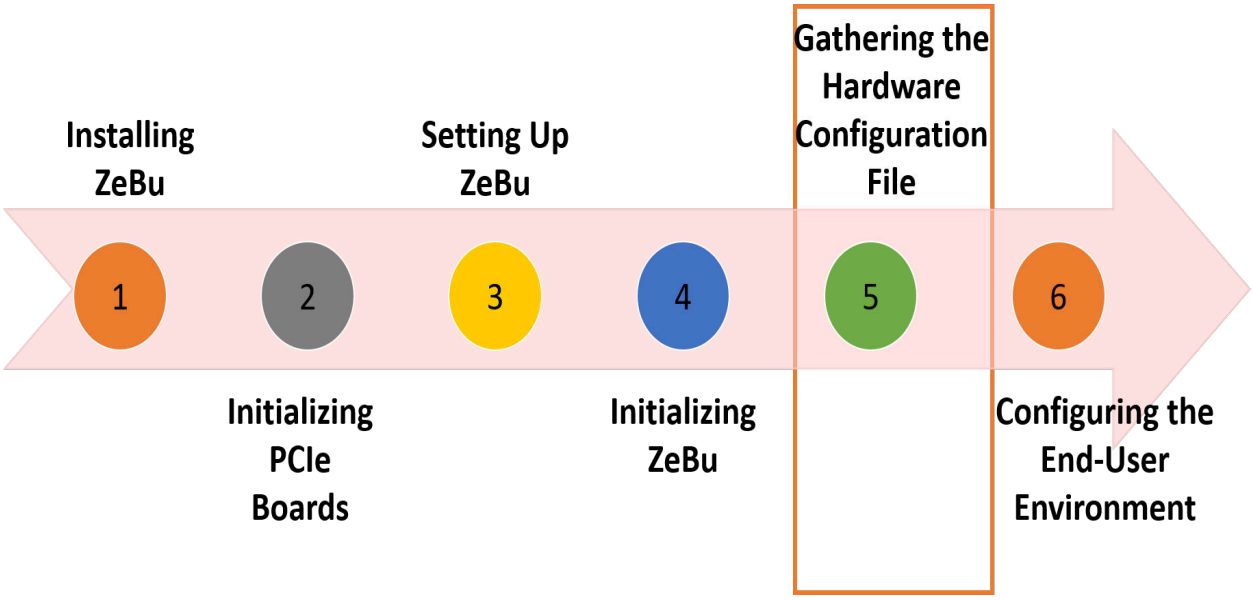
6

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 4 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, 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 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 Generating a Configuration File for Several 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).

Feedback

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. 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.

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 because the `status_se.tcl` files are required.

For more information, see the following subsections:

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.

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.

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

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 4 Syntax Rules

Character in Syntax	Value	Type of Module
a through y	0	No of module

Feedback

Table 5 ZeBu Server 4 Syntax Rules (Continued)

Character in Syntax	Value	Type of Module
	M	12F
	H	Hub

See the following examples:

- Example Common to All ZeBu Server Hardware
- ZeBu Server 4 Examples

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
```

ZeBu Server 4 Examples

To create a default configuration file with 4 FPGA modules:

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

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
  ```

Feedback

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

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

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 -nbfpga <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

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

```
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*.

Feedback

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>
```

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.

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 4 Example 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
```

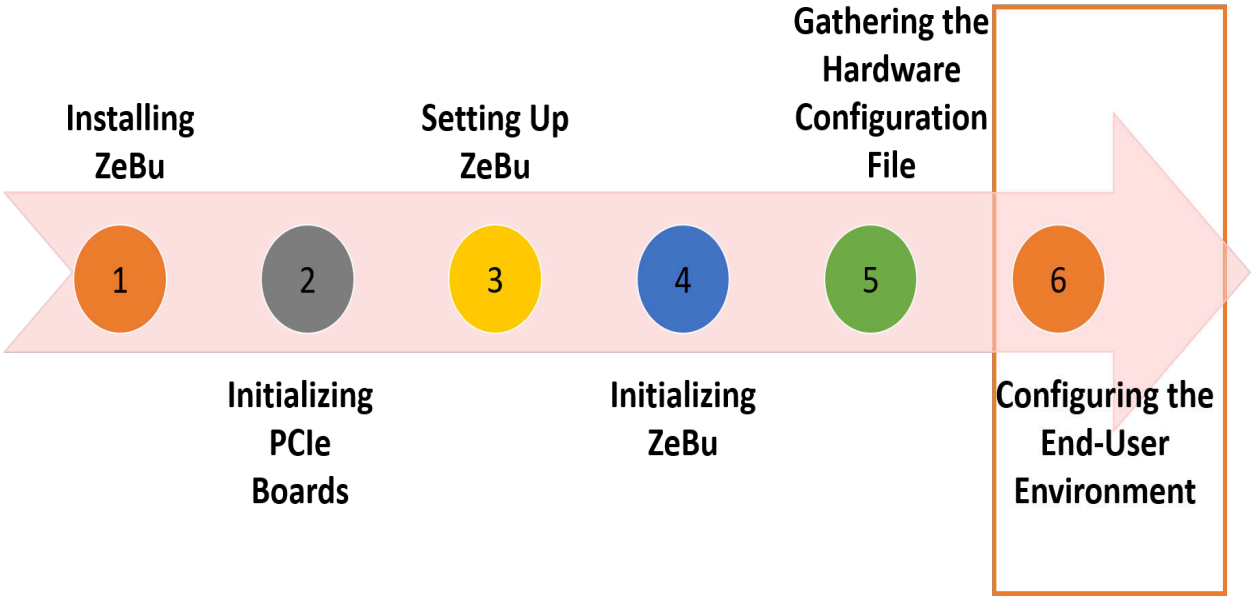
Following are some ZeBu Server 4 configuration files:

7

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 5 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
- Setting the Stack Size for Compilation
- Activating a ZeBu and Xilinx License
- Declaring the System Directory for Emulation Runtime
- Runtime Settings for Relocation

Feedback

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.

Setting the Stack Size for Compilation

You must set the `stacksize` to *unlimited* when compiling using **zCui**. This recommendation applies to all hosts of the grid that is used to compile.

To set the `stacksize`, execute the following commands:

- **In Bash Shell**:

  ```
  ulimit -s unlimited
  ```

  To check the value of `stacksize` in bash, use the following commands:

  ```
   $ ulimit -s -H
  unlimited
   $ ulimit -s -S
  8192
  ```

  For more details, see bash manpage.

- **In C Shell**:

  ```
  $ limit stacksize unlimited
  ```

  or

  ```
  $ unlimit stacksize
  ```

  To check the value of `stacksize` in `csh`, use the following commands:

  ```
   $ limit stacksize
  stacksize    unlimited
  ```

Feedback

```
 $ limit -h stacksize
stacksize    unlimited
```

For more details, see `csh` manpage.

The following error message is reported if the `stacksize` is not set to `unlimited`:

```
### Internal error (ref '11')
### warning in zHandler [ZH0001W] : Stack size is limited. Segmentation
 Fault could be because of insufficient stack space. Check the stack
 printed below. Try to run with unlimited stack size to rule out this
 possibility for the crash.
```

The following warnings are reported if the `stacksize` is not set to `unlimited`:

- **zTopBuild**

```
### warning in zTopBuild [ZTB0249W] : can't set stack to unlimited,
 zTopBuild may randomly crash.
```

- **zCoreBuild**

```
### warning in zCoreBuild [ZBD0218W] : can't set stack to unlimited,
 zCoreBuild may randomly crash.
```

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.

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 in the zini file.

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 environment 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.

Feedback

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 4
- Runtime Settings for Speed Adapters

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. 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
```

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
```

Checking the Utilization of a ZeBu System

To know the current utilization of the modules 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
```

8

Monitoring the Usage of the ZeBu System

This chapter explains how to monitor the usage of the ZeBu system in the following subtopics:

ZeBu HTML Monitor

ZeBu HTML monitor is a tool that provides real time, and daily overview of the usage of a ZeBu emulator. The ZeBu HTML monitor is a process that generates all the HTML pages with all the reports, which are updated continuously, in real time. The amount of resources required to run the ZeBu HTML monitor are very light.

The ZeBu HTML monitor package is a set of scripts that generate HTML pages based on data extracted from the `ZEBU_SYSTEM_DIR` directory.

All ZeBu platforms (ZeBu Server 4, ZeBu Server 5, ZeBu EP1, and ZeBu EP2) can be monitored.

The following prerequisites are required to use the ZeBu HTML monitor:

- The ZeBu installation is required (`ZEBU_ROOT`).
- The `ZEBU_SYSTEM_DIR` directory of the emulator to be monitored.
- The Perl scripting language needs to be installed.

Feedback

Installing ZeBu HTML Monitor

To install the ZeBu HTML monitor, perform the following steps:

1. Copy `$ZEBU_ROOT/etc/scripts/hardware_resource_monitor/hardware_resource_monitor.tar.gz` to your local directory.
2. Extract the package in a new user directory.
3. Execute `zebu_monitor.sh`.

   The output is displayed as follows:

```
 $ ./zebu_monitor.sh
Please, leave this script running...
HTML pages will keep on being generated in
 file:///path/to/directory/html/zebu_status.html
```

4. Use one of the browsers to view the HTML page that is generated.

   Example:

```
firefox /path/to/directory/html/zebu_status.html
```

Use Model

All the required files are installed in a user directory, including a configuration file (`config.txt`) to tweak the HTML monitor's behavior. The result of the background process (script generating / updating the web pages) and all the associated data is stored in this directory.

The `config.txt` file is used to provide required parameters to the HTML monitor and can be changed. The current variables and their possible parameters are captured in the following table:

Table 6 config.txt Variables and Parameters

Variable name	Possible values (default)	Description
`ZEBU_SYSTEM_DIR`	Path (`/path/zebu/system/files`)	Path to the ZeBu system directory where the data from **zRscManager** is stored
`HTML_DIR`	Path (`./html`)	Path to the generated HTML pages
`ROOT_DIR`	Path (.)	Deprecated, do not change

Feedback

Table 6 config.txt Variables and Parameters (Continued)

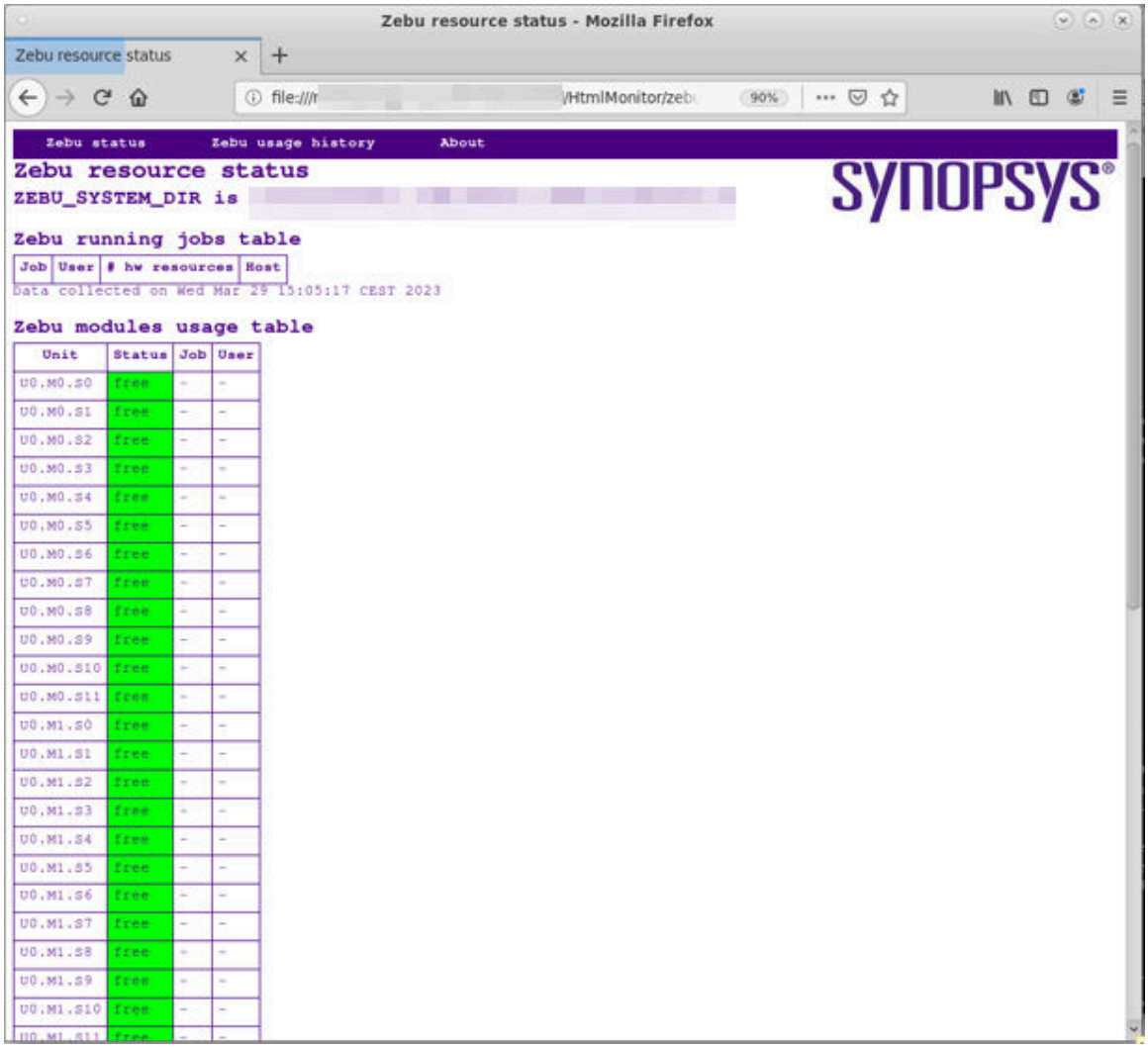
Variable name	Possible values (default)	Description
DAYS_TO_REPORT	Positive integer (10)	Number of days before the current time to take into account when gathering job history data
ZSD_DESCRIPTION_FILE	Path to file (./zsd_description.txt)	The entire emulator's structure are detailed.
MONITOR_DATA_DIR	Path (./monitor_data)	Information about all the jobs are detailed in files organized in the report.
USERS_DATA_FILE	Path to file (./users_data.txt)	Information about the emulator's users are displayed.

Feedback

Example Reports

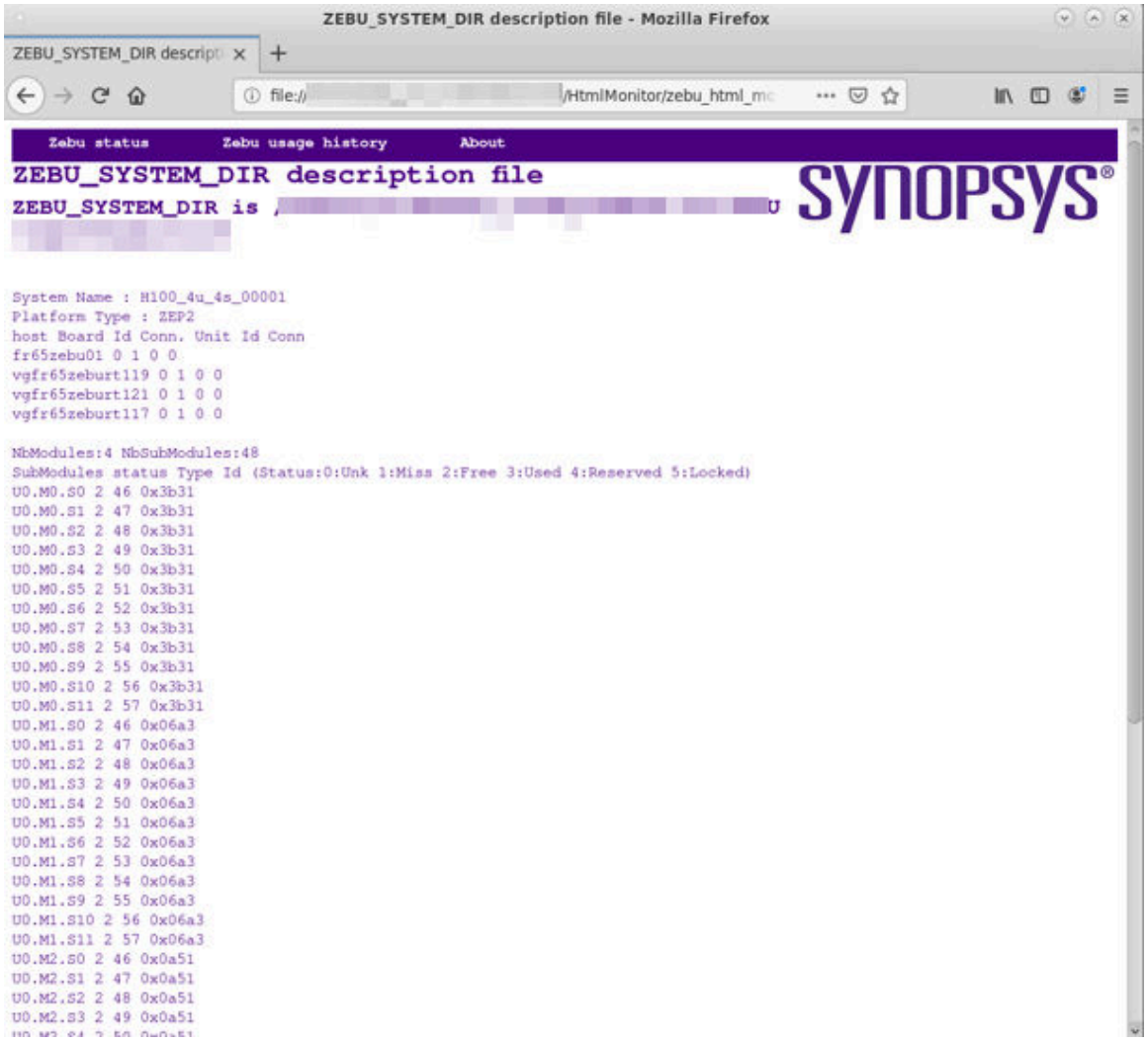
The ZeBu HTML monitor displays the following information in a graphical mode:

- Emulator Status: Displays the real time usage of the emulator.



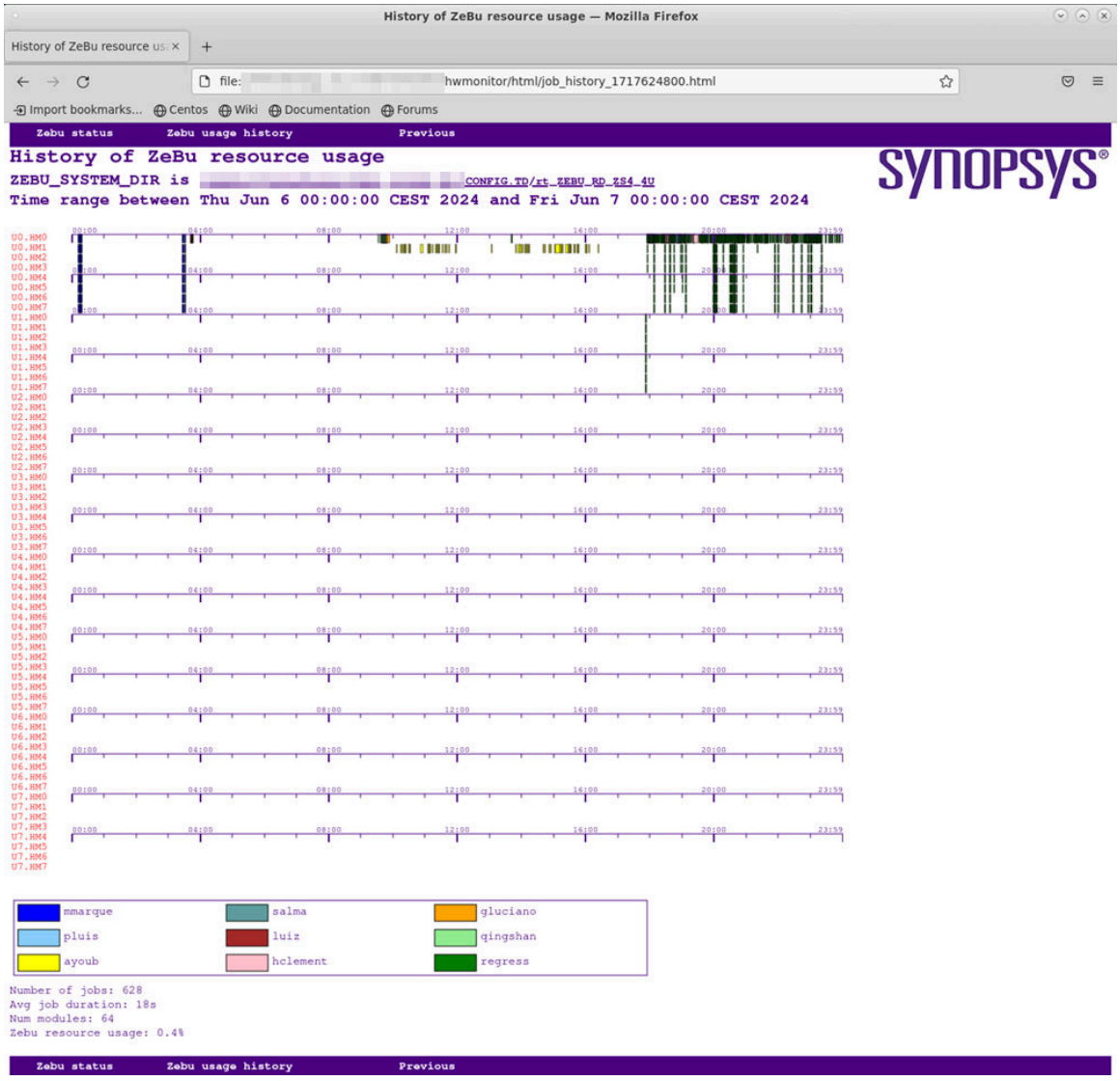
- Emulation Description: Displays the description of each module in the emulator when you click the `ZEBU_SYSTEM_DIR` link on the home page.

Feedback


ZEBU_SYSTEM_DIR description file - Mozilla Firefox
Zebu status
Zebu usage history
About
ZEBU_SYSTEM_DIR description file
ZEBU_SYSTEM_DIR is
SYNOPSYS®
System Name : H100_4u_4s_00001
Platform Type : ZEP2
host Board Id Conn. Unit Id Conn
fr65zebu01 0 1 0 0
vgfr65zeburt119 0 1 0 0
vgfr65zeburt121 0 1 0 0
vgfr65zeburt117 0 1 0 0
NbModules:4 NbSubModules:48
SubModules status Type Id (Status:0:Unk 1:Miss 2:Free 3:Used 4:Reserved 5:Locked)
U0.M0.S0 2 46 0x3b31
U0.M0.S1 2 47 0x3b31
U0.M0.S2 2 48 0x3b31
U0.M0.S3 2 49 0x3b31
U0.M0.S4 2 50 0x3b31
U0.M0.S5 2 51 0x3b31
U0.M0.S6 2 52 0x3b31
U0.M0.S7 2 53 0x3b31
U0.M0.S8 2 54 0x3b31
U0.M0.S9 2 55 0x3b31
U0.M0.S10 2 56 0x3b31
U0.M0.S11 2 57 0x3b31
U0.M1.S0 2 46 0x06a3
U0.M1.S1 2 47 0x06a3
U0.M1.S2 2 48 0x06a3
U0.M1.S3 2 49 0x06a3
U0.M1.S4 2 50 0x06a3
U0.M1.S5 2 51 0x06a3
U0.M1.S6 2 52 0x06a3
U0.M1.S7 2 53 0x06a3
U0.M1.S8 2 54 0x06a3
U0.M1.S9 2 55 0x06a3
U0.M1.S10 2 56 0x06a3
U0.M1.S11 2 57 0x06a3
U0.M2.S0 2 46 0x0a51
U0.M2.S1 2 47 0x0a51
U0.M2.S2 2 48 0x0a51
U0.M2.S3 2 49 0x0a51

Feedback

- Usage History: Displays the usage history of ZeBu resource.



Limitations of ZeBu HTML Monitor

ZeBu HTML monitor has the following limitations:

- You can use only one `ZEBU_SYSTEM_DIR` and one ZeBu environment (`ZEBU_ROOT`) in a working directory. If a new session needs to be created with a different couple (`ZEBU_SYSTEM_DIR` / ZeBu Env), it is mandatory to set `config.txt` and run `zebu_monitor.sh` in the new directory.

- Only one instance of the script is allowed to run in a directory. However, multiple viewers can access the generated HTML pages.
- The script that updates the real-time information and the daily data is running in background, and records data in user directory.

  You must remove the used directory after terminating the script.

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

When launching the **zDriverInstall** 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:

```
$ ./zDriverInstall compile
zDriverInstall INFO : Compiling the module using /bin/gcc version 4.8.5
make -C /lib/modules/3.10.0-1127.10.1.el7.x86_64/build M=/A/B/zebu_dd/src
 modules
make: *** /lib/modules/3.10.0-1127.10.1.el7.x86_64/build: No such file or
 directory. Stop.
make: *** [modules] Error 2
mv: cannot stat '/A/B/zebu_dd/src/zebu_dd.ko': No such file or directory
rm -rf *.o *~ core .depend .*.cmd *.ko* *.mod.c .tmp_versions
 Module.symvers modules.order
```

Solution

Feedback

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 **zDriverInstall** to compile the **zKernel** module (see Initializing the PCIe Boards).

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 (http://zebu_support@synopsys.com):

```
$ zUtils -getPCInfo
```

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 the PCIe Boards.

Note:

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

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.

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.

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
```

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_2
s_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_2
s_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 Setting Up ZeBu) or `zUtils -initSystem` (described in Initializing the ZeBu Server System).

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.

Feedback

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

This chapter has the following sections:

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:

```
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
```

Feedback

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
  ```

Hardware Modules Status

If there is any issue in the maintenance status of hardware modules, the following messages are displayed.

zServer

```
- ZeBu : zServer : ERROR : LUI3636E : Fatal error in board connection.
 Current software version cannot use the following modules which are out
 of maintenance (IDs are displayed in hexadecimal):
- ZeBu : zServer : ERROR : LUI3636E : 7fb2
- ZeBu : zServer : ERROR : LUI3636E : dd32
- ZeBu : zServer : ERROR : LUI3636E : 5122
```

Feedback

zUtils

```
- ZeBu : zUtils : WARNING : Following modules are out of maintenance and
 shouldn't be used with current software version (IDs are displayed in
 hexadecimal):
- ZeBu : zUtils : WARNING : 7fb2
- ZeBu : zUtils : WARNING : dd32
- ZeBu : zUtils : WARNING : 5122
```

10

Appendix A: Examples of setup_template_zs4.zini

This chapter provides an example of the **setup_template_zs4.zini** file for ZeBu Server 4.

Example of `setup_template_zs4.zini` File for ZeBu Server 4

```
# This file is a template to help creating your own <my_setup>.zini file
# <my_setup>.zini file is an input for the following tools:
# - zSetupSystem <my_setup>.zini
# - zUtils -initSystem <my_setup>.zini

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

# All these system files are used by the Zebu Server-4 system at runtime.
# The following directories are generated (or updated if they already
 exist):
#   - dt          : contains all input files for 'zUtils -calibration'
 and 'zUtils -dt' user tests
#   - config      : output directory generated by zConfig
# The following files are generated (or updated if they already exist):
#   - config.dff    : complete architecture of the detected ZeBu Server-4
 system
#   - memories.dff  : status of all the memories in the ZeBu system
#   - status_se.tcl : system status for the design compilation process
 (input file for zConfig)
#   - <my_setup>.zini    : a copy of the setup file with the same name as
 the file used for zSetupSystem

#$ZEBU_SYSTEM_DIR = "my_ZeBu_System_Dir";

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

##### Input configuration diagnostics file #####
#$diagFileVersion = 3.0;

## NOTE:
#  - if $diagFileVersion is not specified:
#         zUtils looks for the default <file.diag> into the
 $ZEBU_ROOT/etc/firmwares/ORION/dt/V3.0/ directory.
```

```
#  - if $diagFileVersion is specified:
#         zUtils looks for the <setup.file> into the
 $ZEBU_ROOT/etc/firmwares/ORION/dt/V<$diagFileVersion>/ directory.

# Use $copyBitstreams to request a copy instead of a link
#$copyBitstreams = 1;

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

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

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 4:

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