ZeBu® Server 4 Smart Z-ICE Interface User 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

Free and Open-Source Licensing Notices

Third-Party Links

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

www.synopsys.com

Contents

Preface

This chapter has the following sections:

- About This Book
- Contents of This Book
- Related Documentation
- Typographical Conventions
- Synopsys Statement on Inclusivity and Diversity

About This Book

The *ZeBu® Server 4 Smart Z-ICE Interface User Guide* gives a physical description of the Smart Z-ICE interface and the steps to instantiate and use it on ZeBu Server 4.

Contents of This Book

The *ZeBu®Server 4 Smart Z-ICE Interface User Guide* has the following chapters:

Chapter	Describes...
Physical Description	The physical construct of the Smart Z-ICE Interface of ZeBu Server 4
Instantiating Smart Z-ICE	The steps to instantiate the Smart Z-ICE Interface
Remapping Smart Z-ICE For Emulation	The steps to use the Smart Z-ICE Interface

Related Documentation

Document Name	Description
ZeBu User Guide	Provides detailed information on using ZeBu.
ZeBu Debug Guide	Provides information on tools you can use for debugging.

Document Name	Description
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 Synthesis Verification User Guide	Provides a description of zFmCheck.
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.
ZeBu Hybrid Emulation Application Note	Provides an overview of the hybrid emulation solution and its components.

Typographical Conventions

This document uses the following typographical conventions:

To indicate	Convention Used
Program code	`OUT <= IN;`

To indicate	Convention Used
Object names	`OUT`
Variables representing objects names	`<sig-name>`
Message	Active low signal name '<sig-name>' must end with _X.
Message location	`OUT` <= IN;
Reworked example with message removed	`OUT_X` <= IN;
Important Information	`NOTE:` This rule...

The following table describes the syntax used in this document:

Syntax	Description
[ ] (Square brackets)	An optional entry
{ } (Curly braces)	An entry that can be specified once or multiple times
| (Vertical bar)	A list of choices out of which you can choose one
...`(Horizontal ellipsis)`	Other options that you can specify

Synopsys Statement on Inclusivity and Diversity

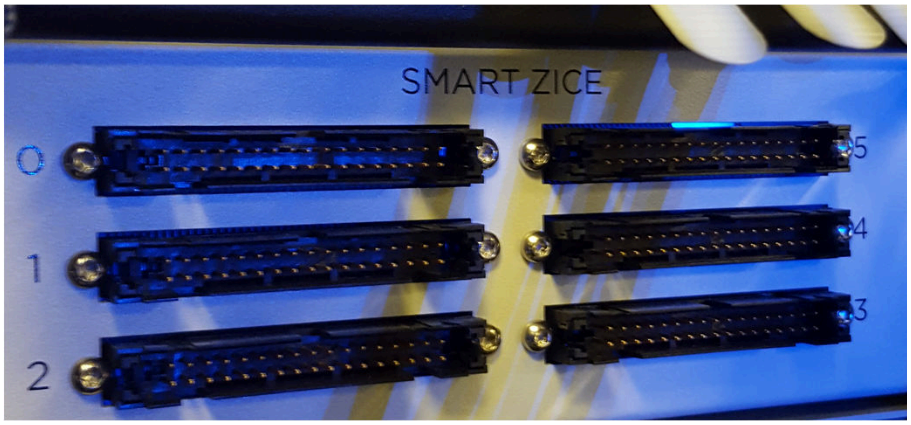
Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language 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

Physical Description

The Smart Z-ICE interface supported by ZeBu Server 4 is based on standard HE10 connectors. The interface consists of 6x34 pin connectors available on the front panel of the ZeBu Server 4 control interface. The following figure displays the port connectors on the ZeBu Server 4 control interface.

Figure 1 Smart Z-ICE Port Connectors on the ZeBu Server 4 Control Interface



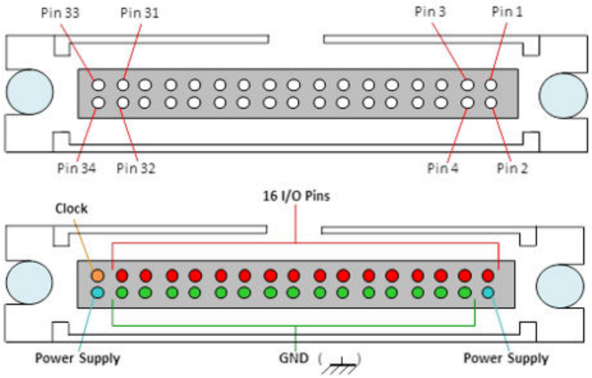
In ZeBu Server 4, Smart Z-ICE voltage level cannot be configured and is fixed at 1.8V.

The 34 pins of each Smart Z-ICE connector are allocated as follows:

- 16 pins are I/O pins for data
- 15 pins are connected to GND
- 2 pins are connected to power supply
- 1 pin for clock

The following figure displays the HE10 connector and the pin allocation.

Figure 2 Pin Allocation in the ZeBu Server 4 Smart Z-ICE HE10 Connector



The following table lists the pin allocation on *ZeBu Server 4 Smart Z-ICE* HE10 connector.

Table 1 ZeBu Server 4 Smart Z-ICE HE10 Connector Pins for Data

HE10 Connector Pin#	Signal Name
1	Data
3	Data
5	Data
7	Data
9	Data
11	Data
13	Data
15	Data
17	Data
19	Data
21	Data
23	Data
25	Data
27	Data
29	Data

Table 1 ZeBu Server 4 Smart Z-ICE HE10 Connector Pins for Data (Continued)

HE10 Connector Pin#	Signal Name
31	Data
33	Clock

2

Instantiating Smart Z-ICE

In ZeBu Server 4, the Smart Z-ICE connection is established by instantiating a dedicated module named `SMART_ZICE_ZSE`. This module can be instantiated anywhere in the design hierarchy (DUT or transactor); it is supported only in **SystemVerilog** and **Verilog** languages.

The syntax of the module instantiated to use Smart Z-ICE is given below.

```
SMART_ZICE_ZSE (
   .sampling_clock (<sampling clock name>),
   .<port direction>_port<port number>_<pin number> (<wire name>),
   .<port direction>_port<port number>_<pin number> (<wire name>),
   .<port direction2>_port<port number2>_<pin number2> (<wire name2>),
  .clock_out_port<port number>_33 (clock wire name)
);
```

The following table describes each port of the `SMART_ZICE_ZSE` module.

Table 2 Description of SMART_ZICE_ZSE Module Ports

Port	Description
`.<port direction>_port<port number>_<pin number> (<wire name>)`	Specifies the connection between a specific Smart Z-ICE port, which is defined with a direction (`<port direction>`), a port number (`<port number>`), a pin number (`<pin number>`), and a wire (`<wire name>`). Note that the code must be written in Verilog or SystemVerilog. • `<port direction>` accepts the following values: ◦ `-input`: Data is transferred from Smart Z-ICE. ◦ `-output`: Data is transferred to Smart Z-ICE. • `wire < wire name>` drives value on Smart Z-ICE. ◦ `-inout`: It means that the wire `<wire name>` is a bi-directional wire. • `<port number>` is an integer between 0 and 5. This is linked to the Smart Z-ICE port number expected to be used (see Figure 1) • `<pin number>`: Represents the specific pin to be used on the Smart Z-ICE port. Table 1 in the Physical Description of Smart Z-ICE chapter summarizes the allocation for each pin available on each Smart Z-ICE connector.
`.clock_out_port<port number>_33 (clock wire name)`	Specifies the Smart Z-ICE port dedicated to the clock. In contrast to ZeBu Server 3, only clock going from design to Smart Z-ICE is supported.

Table 2 Description of SMART_ZICE_ZSE Module Ports (Continued)

Port	Description
`.sampling_clock (<sampling clock name>)`	Specifies `<sampling clock name>` as the sampling clock for data coming from Smart Z-ICE.The main purpose is to synchronize the input data of Smart Z-ICE. `<sampling clock name>` can be one of the following: • a primary (derived) clock signal (output of a `zceiClockPort` instance) • a design signal

3

Remapping Smart Z-ICE For Emulation

During emulation, you can remap an entire Smart Z-ICE port to another one. There are two possible ways to remap Smart Z-ICE given below. For details, see the following sections:

- By Using the Environment Variable
- By Using the designFeatures Command

In addition, this chapter covers how to disable ports at runtime. For detailed steps, see Disabling Smart Z-ICE ports at Runtime

By Using the Environment Variable

To remap a Smart Z-ICE port to another one for emulation runtime, you can use the following environment variable: `ZEBU_SMARTZICE_<PORT>_PHYSICAL_LOCATION`

where, `<PORT>` is the identifier of the Smart Z-ICE port instantiated in the design (`P0` to `P5`).

This variable must be set to the value of the port targeted for emulation (`P0` to `P5`).

Example

```
setenv ZEBU_SMARTZICE_P0_PHYSICAL_LOCATION P1
```

In this example, the Smart Z-ICE port `P0` is allocated to physical port `P1` for emulation runtime.

By Using the *designFeatures* Command

To remap a Smart Z-ICE port to another one for emulation runtime, you can use the following `designFeatures` command: `$smartZICE.connectorRemap_<PORT NB>`

where, `<PORT NB>` is the number of the instantiated Smart Z-ICE port (`0` to `5`).

This command sets the number of the port targeted for emulation (`0` to `5`).

Example

```
$smartZICE.connectorRemap_0 = 2;
```

In this example, the Smart Z-ICE port `P0` is allocated to physical port `P2` for emulation runtime.

Disabling Smart Z-ICE ports at Runtime

To disable all the Smart Z-ICE ports at runtime (that have been enabled during compilation), set the environment variable `ZEBU_SMARTZICE_DISABLE` to `TRUE`.

A warning message appears as follows:

Warning:

> You've compiled for using SmartZice but you've decided not to use it by setting the variable, `ZEBU_SMARTZICE_DISABLE` as `OK|YES|TRUE|ON`.