ZeBu ® Hybrid Emulation Application Note

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

The *ZeBu® Hybrid Emulation Application Note*provides a brief overview of the hybrid emulation solution and its components. It also lists the UTF commands required for hybrid emulation.

Contents of This Book

The *ZeBu® Hybrid Emulation Application Note*has the following chapters:

Chapter	Describes...
Introduction to Hybrid Emulation	Provides a brief overview to hybrid emulation.
Hybrid Platform Bring-Up	Describes components of a hybrid platform bring-up.
Hybrid Platform Runtime	Briefly describes hybrid platform at runtime.

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.

Feedback

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

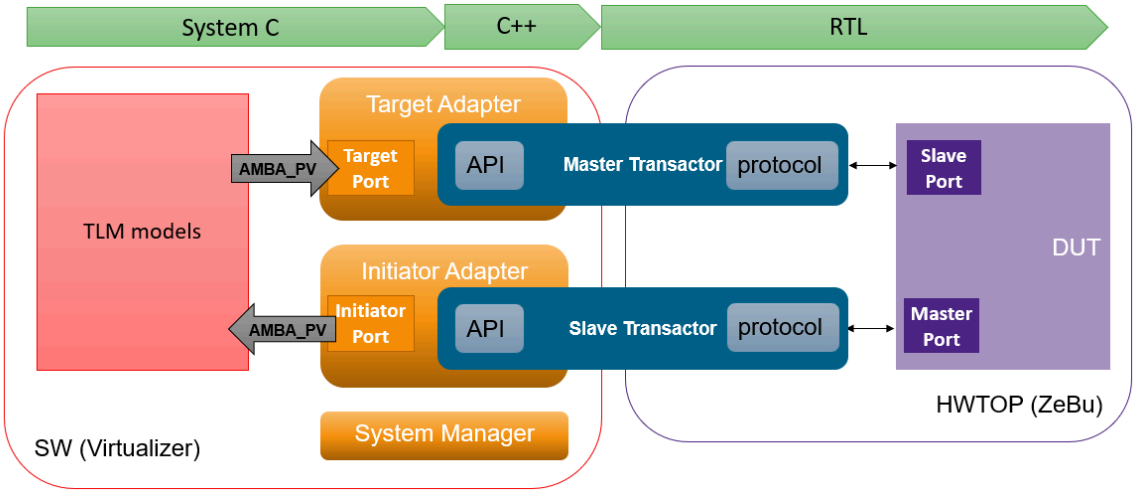
Introduction to Hybrid Emulation

Hybrid emulation entails running part of a design on a software simulator using Transaction-Level Models (TLM) and another part of the design on a hardware emulator. The ZeBu hybrid emulation solution consists of the following three components:

- A virtual platform that hosts fast models and runs in Virtualizer.
- The ZeBu Hybrid Adapter Library (ZHAL) that acts as a bridge between the virtual platform and a ZeBu transactor (typically AMBA).
- The Design Under Test (DUT) platform mapped onto ZeBu.

The following figure displays the components of the hybrid emulation platform with ZeBu.

Figure 1 ZeBu Hybrid Emulation Platform



The subsequent chapters describe only the mapping of the DUT platform on ZeBu.

For information about the virtual platform, refer to the *Virtualizer Documentation*.

For information about the TLM2 adapter, refer to the *ZeBu Hybrid Adaptor Library User Manual*.

Both these documents are available on SolvNetPlus.

2

Hybrid Platform Bring-Up

Hybrid platform bring-up consists of the following:

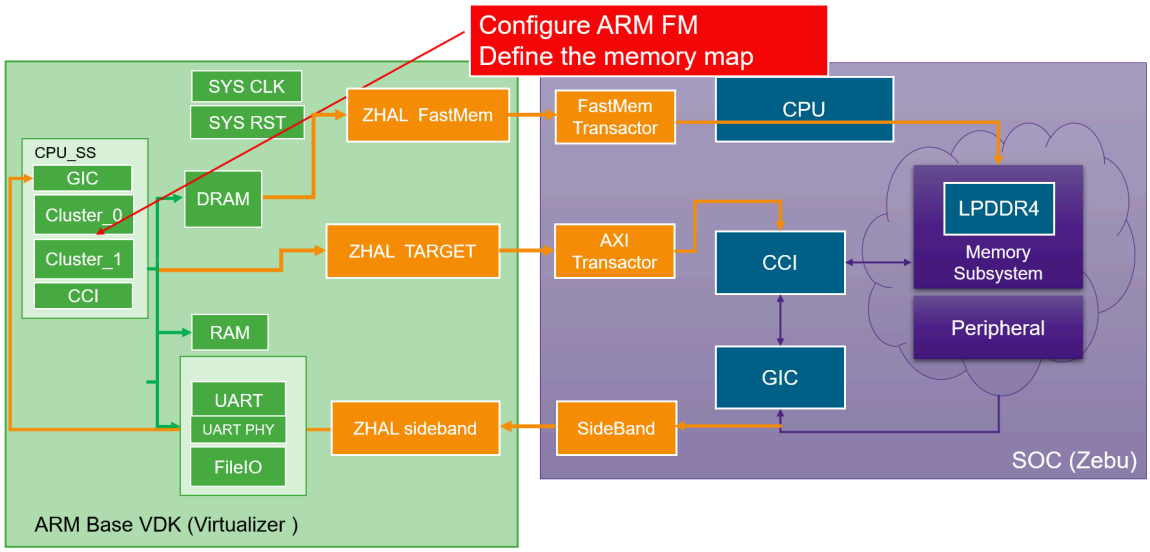
1. Using the ARM Base Virtualizer Development Kit (VDK) in Virtualizer to leverage the features of the ARM Fast Model ecosystem in just two steps.
2. Establishing connections between Virtualizer and ZeBu by:
   - Connecting an AMBA Transactors via the ARM Fast Model with the DUT/ interconnect
   - Connecting a ZHAL SideBand Transactors with:
     - Reset from Virtualizer to the DUT
     - Interrupts from the DUT to the GIC fast model
   - Connecting a shared memory, called FastMem Transactors, between the VDK and ZeBu
3. Setting up the ARM Base VDK in Virtualizer. This involves:
   - Configuring the ARM Fast Model
   - Setting up the memory map

For each transactor instantiated in the HWTOR, there is a counterpart ZeBu Hybrid Adaptor Library (ZHAL) adaptor in the VDK aimed to bridge the transactor API and the TLM2. ZHAL offers the ZHAL_CBB where CBB stands for Configurable Building Block.

The hybrid platform bring-up architecture is shown in the figure below.

Feedback

Figure 2 Hybrid Platform Bring-Up



In addition, this chapter consists of the following sections:

- AMBA Transactors
- SideBand Transactors
- FastMem Transactors

AMBA Transactors

ZHAL supports the APB, AHB, AXI3/4, ACElite, ACE and CHI ARM Advanced Microcontroller Bus Architecture (AMBA) ZeBu transactors.

SideBand Transactors

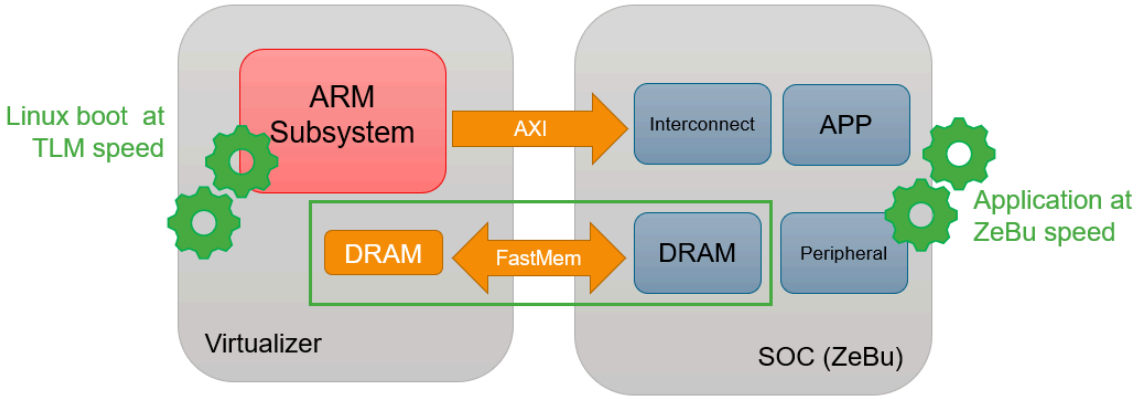
ZHAL offers SidebandIn and SidebandOut pin-level transactors aimed to wire signals between Virtualizer and ZeBu.

FastMem Transactors

ZHAL FastMem is a key memory sharing technology between ZeBu and VIrtualizer that helps achieve high performance for hybrid prototypes.

Feedback

Figure 3 FastMem Architecture



3

Hybrid Platform Runtime

Virtualizer is a fully integrated GUI able to create the TLM/systemC platform and then a hybrid platform. It is also in charge of running the hybrid emulation.

It offers comprehensive debugging features that includes:

- Trace capability on TLM2 transactions and sockets
- Capability to attach a software debugger such as T32 for software development

ZHAL brings a ZeBu plugin to Virtualizer that enables:

- Full logging of the transactors and adaptors
- ZeBu waveform formats
- A RTL browser

ZHAL also supports a zRci testbench and handles most of the UCLI commands.