



Accessing Combinational Signals with ZeBu

AN027
Revision d

ZeBu Application Note

October 2011

Purpose:

This document presents the ZeBu capability to provide RTL waveforms which include the combinational signals of the design without the addition of probes in the design.

This Combinational Signals Accessibility (CSA) feature includes the software simulation after emulation runtime without any third-party simulator.

Two different use-models are available according to the debugging strategy: off-line CSA when targeting simulation of all or a large part of the design and iCSA for interactive selection of the signals before simulation.

This document describes the compilation settings for CSA feature and provides information for emulation runtime and generation of RTL waveform files for both off-line CSA and iCSA.

Applicability:

This document is applicable for V6_3_1 software version for ZeBu-Server.

History:

This table gives information about the content of each revision of this manual, with indication of specific applicable version:

Doc Revision	Date	Evolution
d	Oct 11	Update for V6_3_1 features, focusing on off-line CSA and iCSA. Online CSA is out of the scope.
c	Feb 10	Update for V4_3_3D_00 and V6_2_0.
a		First Edition.



Copyright Notice Proprietary Information

Copyright © 2009-2011 EVE. All rights reserved.

This software and documentation contain confidential and proprietary information that is the property of EVE. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of EVE, or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with EVE permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

"This document is duplicated with the permission of EVE, for the exclusive use of _____ and its employees. This is copy number ____."

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

EVE 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.



Table of Contents

1	INTRODUCTION.....	4
2	COMPILATION OPTIONS.....	5
2.1	ACTIVATING CSA	5
2.2	SPECIFIC OPTIONS FOR ICSA	6
2.3	DURING COMPILATION	6
3	OFF-LINE CSA	8
3.1	INTRODUCTION.....	8
3.2	SELECTING THE SUPPORT SIGNALS WITH zDbPostProc	8
3.3	WRITING THE C++/C TESTBENCH.....	9
3.4	GENERATING THE CSA WAVEFORMS.....	9
3.5	LOAD SHARING FOR GENERATION OF THE RTL WAVEFORMS.....	12
3.5.1	Launching zSplitResimu	12
3.5.2	Merging the resulting files with zFSDBmerge	13
4	INTERACTIVE WAVEFORM GENERATION WITH ICSA.....	14
4.1	LAUNCHING VERDI AND nWAVE VIEWERS FOR ICSA.....	14
4.1.1	To launch Verdi with iCSA	14
4.1.2	To launch nWave with iCSA.....	14
4.1.3	iCSA interface.....	14
4.2	CONFIGURING ICSA.....	15
4.3	CONTROLLING ICSA	16
5	EVE CONTACTS	19

Figures

Figure 1: Activating CSA in zCui	5
Figure 2: nWave menus with iCSA	14
Figure 4: iCSA control window before selection.....	16
Figure 5: Selection of signals for iCSA	17
Figure 6: iCSA control window after selection.....	17
Figure 7: iCSA control window after simulation	17

Tables

Table 1: zResimu Mandatory Parameters	9
Table 2: zResimu Options.....	10
Table 3: zSplitResimu Mandatory Parameters	12
Table 4: zSplitResimu Options.....	13
Table 5: Fields in the <i>Initialize</i> window for iCSA.....	16



1 Introduction

The Combinational Signals Accessibility (CSA) feature provides the possibility to obtain full RTL waveforms by calculating combinational signal values through software simulation.

This capability drastically reduces debugging time in the design being verified. With CSA, it is no longer necessary to insert probes to access combinational signals. Any RTL signal can be monitored without recompiling the design.

All the appropriate tools for CSA are embedded in the ZeBu software and there is no need for any additional simulator.

The CSA feature is activated as an option for **zFAST** synthesis in **zCui** so that specific information is stored in the runtime database. Before proceeding with emulation runtime, all the necessary registers to simulate the combinational signals, known as support registers, are selected so that they can be dumped during emulation. The software simulation itself is an off-line process for which no connection to the ZeBu-Server system is required.

The support registers which have been dumped during emulation are the inputs for the software simulation which generates waveforms for the combinational signals.

Although gate-level waveforms can be generated by CSA, in most cases the FSDB format is selected to get RTL waveforms for debugging. For gate-level waveforms, both VCD and FSDB formats are supported.

There are 2 use models for waveform generation:

- Full waveform dump via Off-line CSA:
Waveforms are generated for the entire chip or for a large portion of the chip, based on ZeBu-proprietary software simulation accelerator, as described in Chapter 3.
- Interactive waveform generation via iCSA:
With FSDB waveform files, the user can interactively select the signals for debugging in Novas™ waveform viewers (Verdi™ and nWave™) and the ZeBu-proprietary software simulation accelerator is automatically invoked to generate the corresponding waveforms, as described in Chapter 4.

Offline CSA and iCSA both require some specific licenses for ZeBu. For iCSA, Verdi or nWave tools are not delivered by EVE.

2 Compilation Options

The CSA feature is supported only for RTL source files synthesized with **zFAST**. The corresponding settings in **zCui** are located in the **zFAST** tab of a given RTL group. When several RTL groups have been declared, the CSA feature can be activated only for the group which includes the top level of the design-under-test.

2.1 Activating CSA

In the **zFAST** tab, select the **Optimization vs Debugging Facilities** check box to activate the frame and choose **Keep all Registers and Simulate Combinational Signals (CSA)** option in the **Accessibility Level** list:

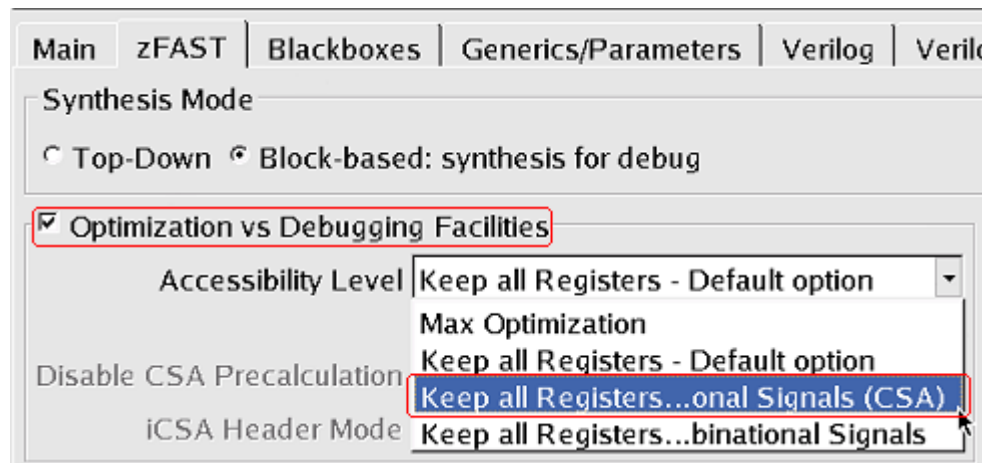


Figure 1: Activating CSA in zCui

When CSA is selected, two checkboxes in other panels of **zCui** are automatically set:

- In the **Memory Sources** panel, the **Memory I/O Accessibility on all zMem models**.
- In the **Backend → Debugging** panel, the **Top-level I/O Accessibility** (same as the default setting when the **zCui** project is created).

These checkboxes should not be cleared while CSA is selected.

When CSA is selected, the **Optimization vs Debugging Facilities** frame offers additional options:





- When the **Disable CSA Precalculation** checkbox is selected, some of the data processing for CSA (**Support Calculator** step in the **Compilation** view) is not done when compiling and is delayed to the simulation step (with a specific option as described in Section 3.5.1).
- The **iCSA Header Mode** options are described in Section 2.2.

Note: If **Keep all Registers and Insert Probes for Combinational Signals** is selected in the **Accessibility Level** list, the CSA feature is not activated and Combinational Dynamic Probes are created for all the combinational signals of the RTL design, with a large impact on the design size.

2.2 Specific options for iCSA

In order to use the integration of CSA with Novas' Verdi or nWave viewers, additional options have to be set in the **Optimization vs Debugging Facilities** frame:

- Selecting **iCSA Header Mode** generates the mandatory additional data. The format of the waveform file (VCD, FSDB or FSDB RTL) has to be selected in the list so that the generated data match the waveform file format.
- With the **Check Library** button, a user targeting FSDB or FSDB RTL format can check that the necessary Novas library is actually available.

2.3 During Compilation

In the **Compilation** view, **zCui** creates a specific group of tasks named **CSA** in the **System Data Base** part of the task tree:

Tasks	S	Comment
Initial Check		1p,0f/1tasks
Design		7p,0f/7tasks
Backend: default backend		46p,0f/46t...
System Compilation		18p,0f/18t...
FPGA Place & Route		18p,0f/18t...
System Data Base		10p,0f/10t...
CSA		6p,0f/6tasks
zCui Accessibility Files Copy	✓	
ZeBu Build Accessibility Graph	✓	
zCui Analyze Accessibility Graph Results	✓	
Support Calculator		2p,0f/2tasks
ZeBu zSupportCalculator (dut.adder)	✓	
ZeBu zSupportCalculator (dut)	✓	
ZeBu Merge CSA Support Calculation	✓	
zCui Create Script of Global DB	✓	
ZeBu Create Global DB	✓	
zCui Create Script (DB PP)	✓	
ZeBu Post-Process Global DB	✓	
Final Check		1p,0f/1tasks

The compilation tasks for CSA are processed in parallel with other tasks, thus they do not adversely impact compilation time.



When the **Disable CSA Precalculation** checkbox is selected, fewer compilation tasks are processed for CSA:

Tasks	S	Comment
Initial Check		1p,0f/1tasks
Design		7p,0f/7tasks
Backend: default backend		42p,0f/42t...
System Compilation		18p,0f/18t...
FPGA Place & Route		18p,0f/18t...
System Data Base		6p,0f/6tasks
CSA		2p,0f/2tasks
zCui Accessibility Files Copy	✓	
ZeBu Build Accessibility Graph	✓	
zCui Create Script of Global DB	✓	
ZeBu Create Global DB	✓	
zCui Create Script (DB PP)	✓	
ZeBu Post-Process Global DB	✓	
Final Check		1p,0f/1tasks

When archiving with **zCui** (**File** → **Archive** menu), the CSA information is stored when selecting **Runtime Files**.



3 Off-line CSA

3.1 Introduction

For off-line CSA, the simulation of the combinational signals is done as a separate process after emulation runtime. Since the simulation is done without being connected to the ZeBu-Server system, off-line CSA allows better productivity: the Zebu-Server system can be used for other tasks in the meantime.

During emulation runtime, only the support registers are dumped (support waveform files). The software simulation results into waveform files for the complete design or for a large portion of it, with RTL hierarchical names (FSDB format only) or with gate-level hierarchical names (FSDB or VCD formats).

The software simulation can be parallelized for very large designs, as described in Section 3.5.

3.2 Selecting the support signals with `zDbPostProc`

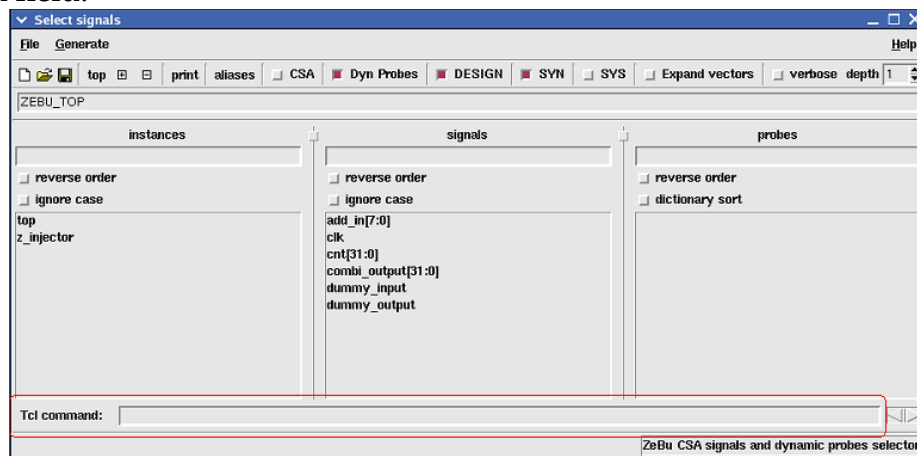
In order to simulate the combinational signals afterward, all the support registers for CSA can be selected to be dumped during emulation runtime. They can also be selected implicitly in the testbench as described in Section 3.3.

In `zDbPostProc`, a dedicated command selects all these registers:

```
select_all_csa_support_regs
```

Additional registers can be selected by user at the same time as the support registers for later debugging. These additional registers will be dumped during emulation runtime and they will be available in the waveform files resulting from software simulation.

Note that no button is available in `zSelectProbes` interface to select all the support registers at once. However, the above command can be entered in the **Tcl command** field:





3.3 Writing the C++/C testbench

For off-line CSA, the runtime environment is almost the same as if the CSA feature was not activated except that the support registers must be dumped all along the time necessary for debug.

If the support registers were previously selected with **zDbPostProc**, the dump is activated with the two following methods (dumpfile creates the register file and dumpvars activates the dump):

```
dumpfile("<register_file>.<format>")  
dumpvars("<path_to_instance>", depth);
```

Where <register_file> is the name of the file in which the support registers are dumped; <format> is the mandatory extension which can be either fsdb or vcd.

It is possible to implicitly indicate that the support registers must be dumped by declaring the name of the file with a dummy .resimu extension in arguments for the dumpfile method. The dumpfile creates the register file and dumpvars activates the dump of the support registers necessary to simulate the combinational signals declared in <path_to_instance>:

```
dumpfile("<register_file>.<format>.resimu")  
dumpvars("<path_to_instance>", depth);
```

Where <register_file> is the name of the file in which the support registers are dumped; <format> is the mandatory extension which can be either fsdb or vcd. The .resimu additional extension is not present in the actual name of the file.

3.4 Generating the CSA waveforms

zResimu is the command-line tool which simulates the combinational signals to generate the CSA waveforms from the support waveforms dumped during emulation runtime. It has only few mandatory parameters:

```
$ zResimu -p <zebu.work> -i <reg_file> -r <root_scope>  
-o <output_file> [options]
```

Table 1: zResimu Mandatory Parameters

Parameter	Description
-p <zebu.work>	Path to the compilation directory where the runtime database is available.
-i <reg_file>	Path to the file in which the support registers were dumped during emulation runtime. Note that when generating the FSDB header file for iCSA, this parameter can be omitted (it is ignored if present).
-r <root_scope>	Hierarchical path to the top of the design in the waveform file.
-o <output_file>	Path to the resulting waveform file (the .fsdb or .vcd extension of the file defines the format of the waveform).

Note that the runtime database which is in the directory given with -p parameter must result from the same compilation as the design which was run to dump the support signals given with -i parameter. In case of mismatch, a warning is



displayed at the very beginning of **zResimu** but the resulting waveforms are not relevant.

Various options are available, in particular to reduce the simulation time by reducing the number of simulated signals or the number of cycles.

Table 2: zResimu Options

Option	Description
-u	The resulting waveforms show RTL names. For FSDB format only.
-e <end_time>	Time at which the simulation will stop (number of cycles of the reference clock of the register file)
-a	Simulates all design signals (recommended). If not present, only signals previously selected with zSelectProbes will be available in the resulting waveforms.
-s <inst_to_simulate>	Hierarchical path of the instance of the design which is selected for simulation. The signals of this instance and of its sub-levels will be available in the resulting waveform file (taking into account -a option or selection).
-f <select_file>	Path to a file which lists the hierarchical paths to signals which are expected in the resulting waveforms, whatever the presence of -a option.
-x <exclude_file>	Path to a file which lists the hierarchical paths to instances in the design which are not expected in the resulting waveforms, taking into account -a option or selection.
-d <depth>	Number of hierarchical levels to simulate. When not present, the simulation is done for all the hierarchical levels in the selected instances.
-n	Proceeds with precalculation if it was disabled in zCui .
-g	For FSDB waveforms only. Only the FSDB header file for iCSA is generated (-i parameter is not mandatory with -g option).
-t <hier_sep>	Hierarchical separator (default is .)
-l <log_file>	If user want to change the path and name of the log file. Default is zResimu.log.

The <select_file> and <exclude_file> have similar syntax: on each line, the hierarchical path to a signal in the design (<select_file>) or to an instance in the design (<exclude_file>). In both cases, the hierarchical path is from the top of the design.

In <select_file>, RTL hierarchical paths can be used when dumping FSDB files with RTL paths (-u option); EDIF paths are mandatory in all other cases.

In <exclude_file>, only EDIF hierarchical paths are supported.

Comments can be added by starting the line with a # character.

Examples:

To generate an RTL waveform file for the complete design, with CSA precalculation disabled during compilation:

```
$ zResimu -p zcui.work/zebu.work -i support.vcd -o simu.fsdb -r top  
-u -n -a
```



To generate the header file for iCSA (if it was not set during compilation) with RTL signal names:

```
$ zResimu -p zcui.work/zebu.work -g -o header.fsdb -r top -u
```

To run **zResimu** on top_dut and exclude everything in core0 to core8, an exclude file (exclude.zxf) must be created:

```
top_dut.core0  
top_dut.core1  
top_dut.core2  
top_dut.core3  
top_dut.core4  
top_dut.core5  
top_dut.core6  
top_dut.core7
```

And launch the following command line with -x option:

```
$ zResimu -p zcui.work/zebu.work -a -r tb_top_dut.top_dut -u  
-i support_regs.fsdb -o resimu_0.fsdb -x exclude.zxf  
-l resimu_0.log
```

To run **zResimu** on top_dut with a list of selected signals, the file listing these signals (signals.list) must be created:

```
top_dut.core0.dout[0]  
top_dut.core0.dout[1]  
top_dut.core0.dout[2]  
top_dut.core0.dout[3]  
top_dut.core0.dout[4]  
top_dut.core0.dout[5]  
top_dut.core0.dout[6]
```

And launch the following command line with -s option:

```
$ zResimu -p zcui.work/zebu.work -f signals.list  
-r tb_top_dut.top_dut -u -i support_regs.fsdb -o resimu_list.fsdb  
-l resimu_list.log
```



3.5 Load sharing for generation of the RTL waveforms

For very big designs, it is not possible to launch the simulation locally on a single PC because it lasts too long for an efficient debug. Therefore it is possible to automatically split the design and launch a multi-process simulation on a PC farm with **zSplitResimu**. This command-line tool creates several CSA waveform files and the appropriate data to merge them after simulation.

Load sharing with **zSplitResimu** requires an external task scheduler such as Sun™ Grid Engine or Platform LSF® in order to share the load among a PC farm.

Before launching **zSplitResimu**, it is mandatory to set the REMOTECMD environment variable with the appropriate command for your load sharing software, as in the following example:

```
export REMOTECMD="qssh -verbose -l arch=lx24-amd64 -l sy=RHEL4*"
```

The exact syntax and options may vary to match the constraints of the IT configuration, in particular to target only compilation PCs with 64-bit operating systems and to manage priorities.

3.5.1 Launching zSplitResimu

The parameters and options for **zSplitResimu** are very similar to **zResimu**:

```
$ zSplitResimu -p <zebu.work> -r <root_scope> -i <reg_file>  
-n <nb_split> -o <output_dir> [options]
```

Table 3: **zSplitResimu** Mandatory Parameters

Parameter	Description
-p <zebu.work>	Path to the compilation directory where the runtime database is available.
-i <reg_file>	Path to the file in which the support registers were dumped during emulation runtime. Note that when generating the FSDB header file for iCSA, this parameter can be omitted (ignored if it is present).
-r <root_scope>	Hierarchical path to the top of the design in the waveform file.
-o <output_dir>	Path to the directory where the resulting waveform files are stored.
-n <nb_split>	Number of parallel simulation processes.

Note that the runtime database which is in the directory given with **-p** parameter must result from the same compilation as the design which was run to dump the support signals given with **-i** parameter. In case of mismatch, a warning is displayed and the resulting waveforms are not relevant.



Table 4: **zSplitResimu** Options

Option	Description
-u	The resulting waveforms show RTL names. For FSDB format only.
-e <end_time>	Time at which the simulation will stop (number of cycles of the reference clock of the register file)
-l <log_file>	If user want to change the path and name of the log file. Default is <code>zSplitResimu.log</code> .

In addition to the waveform files, **zSplitResimu** generates 2 additional files which are intended to merge the resulting waveform files:

- The `xxx.vf` file (`.vf` stands for virtual file) is intended for use with nWave waveform viewers for iCSA, as described in Section 4.
- The `xxx.zvf` file (`.zvf` stands for ZeBu virtual file) is intended for use with **zFSDBmerge**, an EVE-proprietary tool which merges FSDB waveform files into a single waveform file.

3.5.2 Merging the resulting files with **zFSDBmerge**

When **zSplitResimu** was used for the simulation of combinational signals, the ZeBu virtual file created can be used to merge the resulting waveform files into a single FSDB file:

```
$ zFSDBmerge -i <input.zvf> -o <csa.fsdb> [-l <logfile>]
```

Where `<input.zvf>` is the ZeBu virtual file resulting from **zSplitResimu** and `<csa.fsdb>` is the resulting waveform file which includes the simulated signals; `<logfile>` is optional (default is `zFSDBmerge.log`).

4 Interactive waveform generation with iCSA

The CSA feature has been integrated with the Novas Verdi and nWave waveform viewers to provide a unique interactive capability to select some signals and display their related waveforms, using RTL or gate-level hierarchical names.

iCSA fits the use model where the user wants to quickly debug the design by only generating the waveforms for a limited number of signals (a few thousands). However this integration is less efficient than the off-line CSA if the user needs to generate waveforms for the full design.

The support registers are dumped in FSDB format during emulation runtime and the FSDB header results from the compilation of the design (as described in Section 2.2). The signals expected in the waveforms are selected in Verdi or nWave and the software simulation is launched through a dedicated control window.

iCSA adds a ZeBu-dedicated menu in the Verdi/nWave interface and usual options of these tools are available for the waveform files resulting from CSA.

4.1 Launching Verdi and nWave viewers for iCSA

4.1.1 To launch Verdi with iCSA

The command line to launch Verdi with iCSA includes the path to the FSDB header file:

```
$ Verdi_iCSA -ssf <header.fsdb>
```

4.1.2 To launch nWave with iCSA

With nWave, you can either launch iCSA with the header file or with a script resulting from a previous iCSA session:

- Declare the CSA header file:

```
$ nWave_iCSA -ssf <header.fsdb>
```
- Or declare a script saved during a previous iCSA session:

```
$ nWave_iCSA -iCSAInit <iCSA_script.tcl>
```

4.1.3 iCSA interface

In addition to the standard Verdi/nWave interface, a **ZeBu** menu is available which is used to configure iCSA before selecting the signals and launch CSA simulations:

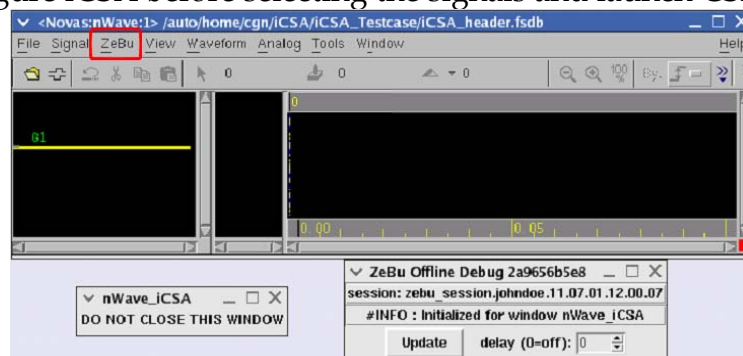
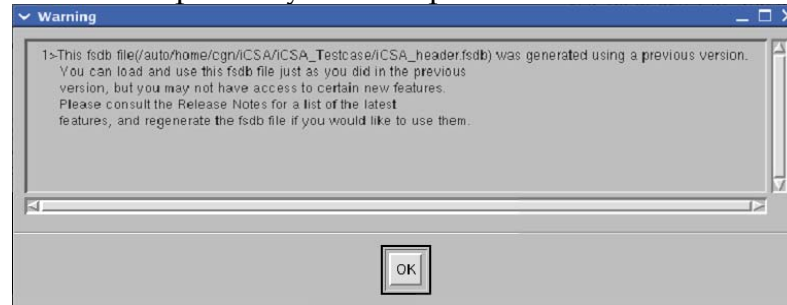


Figure 2: nWave interface with iCSA

4.1.4 Notes about the interface

- Novas Warning Message:

When launching Verdi or nWave for iCSA, the following warning message may appear about compatibility of the input FSDB file:



This message does not refer to ZeBu software but to the Novas libraries which are used when dumping and displaying FSDB waveforms. It does not impact the use of iCSA feature.

- Window for iCSA Communication Control:

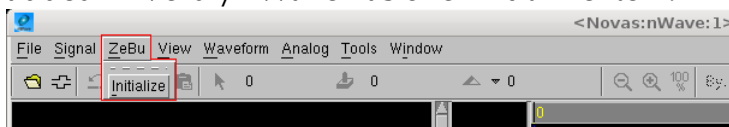
A small window which controls the communication between the ZeBu software and the waveform viewer opens when launching Verdi_iCSA or nWave_iCSA:



It must not be closed before the end of the iCSA session.

4.2 Configuring iCSA

The **ZeBu** menu added in Verdi/nWave has one **Initialize** item:



Selecting **Initialize** shows the following window:

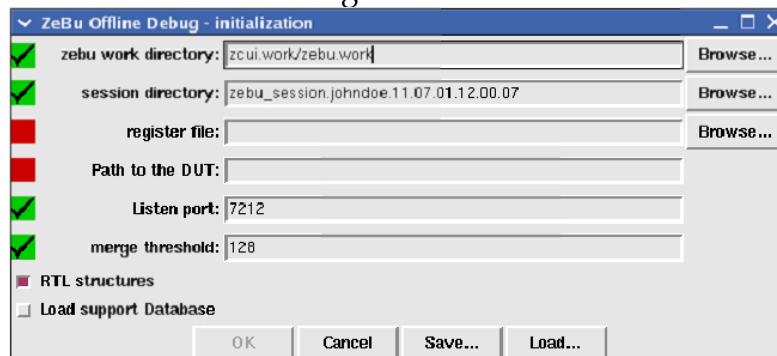





Figure 3: *Initialize* window in Verdi/nWave for iCSA

Table 5: Fields in the *Initialize* window for iCSA

Parameter	Description
zebu work directory	Path to the compilation directory where the runtime database is available.(usually zcui.work/zebu.work)
session directory	Path to the directory in which this session's information is saved. The default value for this field includes the user name and the date and time (e.g. zebu_session.johndoe.11.07.31.12.00.00)
register file	Path to the support waveform file dumped during emulation runtime
path to dut	Hierarchical path to top of the DUT (just the top name in most cases).
listen port	Communication port between simulation software and nWave/Verdi. If several viewer sessions are launched in parallel, this field must have different values for each session.
merge threshold	Maximum number of separate waveform files created in the session. Previous files are automatically merged when this limit is exceeded.

Each item in this window shows a status icon:

-  OK status: the field is filled
-  NOK status: a value is expected.
-  LOCK status: the value cannot be modified (only applicable for the **session directory** filed while a session is already running).

It is not possible to click on the **OK** button if any of the items of the **Initialize** window still has a NOK or LOCK status (the OK button is grayed).

When the **RTL structures** checkbox is selected, the simulation uses RTL hierarchical paths for the signals displayed in the waveform viewer.

Clicking on the **Save** button creates a file with all the initialization parameters. This file can be used in a later session with the **Load** button or as an opening script with nWave, as described in Section 4.1.2.

Clicking on the **OK** button launches the session and new window opens to control iCSA.

4.3 Controlling iCSA

The simulation of combinational signals is controlled from a dedicated window which opens in addition to the Verdi/nWave waveform viewer:

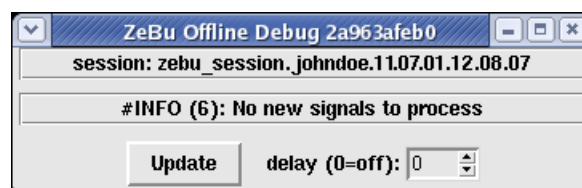


Figure 4: iCSA control window before selection

The signals expected in the waveforms are selected from the following Verdi/nWave window:

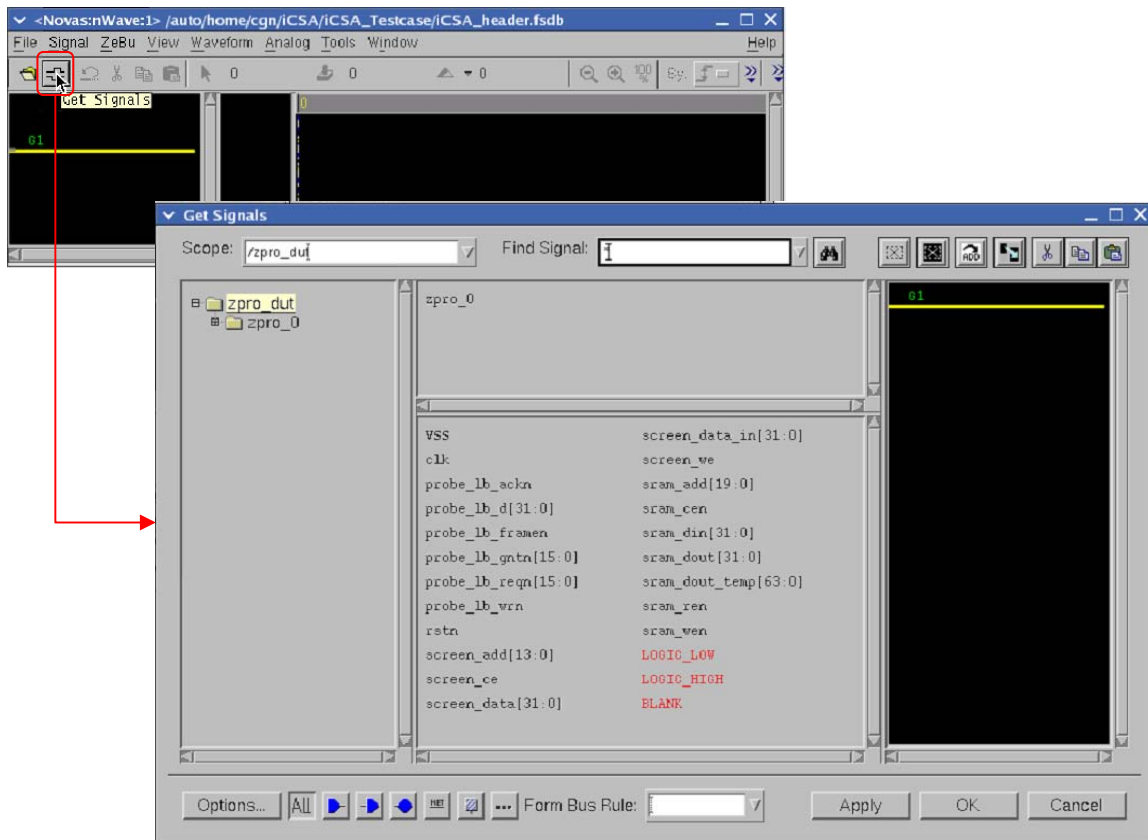


Figure 5: Selection of signals for iCSA

After selection, the iCSA control window shows the following:

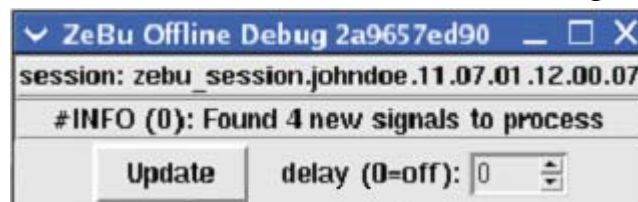


Figure 6: iCSA control window after selection

Clicking on the **Update** button launches the simulation.

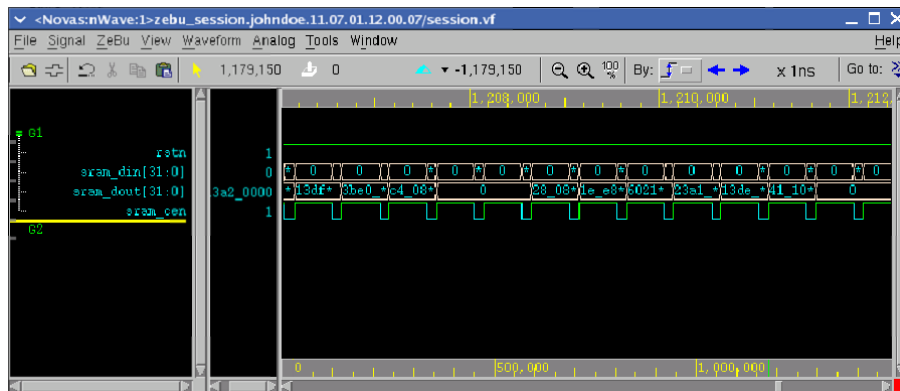
Once the simulation is completed, the control window is modified as follows:



Figure 7: iCSA control window after simulation



The updated waveforms are displayed in nWave/Verdi:





5 EVE Contacts

For product support, contact: support@eve-team.com.

For general information, visit our company web-site: <http://www.eve-team.com>

Europe Headquarters	EVE SA 2-bis, Voie La Cardon Parc Gutenberg, Bâtiment B 91120 Palaiseau FRANCE Tel: +33-1-64 53 27 30
US Headquarters	EVE USA, Inc. 2290 N. First Street, Suite 304 San Jose, CA 95054 USA Tel: 1-888-7EveUSA (+1-888-738-3872)
Japan Headquarters	Nihon EVE KK KAKiYA Building 4F 2-7-17, Shin-Yokohama Kohoku-ku, Yokohama-shi, Kanagawa 222-0033 JAPAN Tel: +81-45-470-7811
Korea Headquarters	EVE Korea, Inc. 804 Kofomo Tower, 16-3, Sunae-Dong, Bundang-Gu, Sungnam City, Kyunggi-Do, 463-825, KOREA Tel: +82-31-719-8115
India Headquarters	EVE Design Automation Pvt. Ltd. #143, First Floor, Raheja Arcade, 80 Ft. Road, 5th Block, Koramangala Bangalore - 560 095 Karnataka INDIA Tel: +91-80-41460680/30202343
Taiwan Headquarters	EVE-Taiwan Branch Room 806 8F, No. 20 GuanChian Road Taipei, Taiwan 100 Tel: +886 2 2375 9275