



The Fastest Verification

---

# **ZeBu-Server Release Note**

**Version 6\_2\_0 B\_00**

February 2010

Copyright © 2010 by EVE — All rights reserved.

This publication is confidential and may not be reproduced, in whole or in part,  
in any manner or in any form, without prior written permission of EVE S.A.



## Table of Contents

<b>ABOUT THIS DOCUMENT .....</b>	<b>5</b>
<b>1 SPECIFIC INFORMATION FOR PATCH B_00 .....</b>	<b>6</b>
1.1 INSTALLATION RECOMMENDATIONS .....	6
1.2 MAJOR NEW FEATURES IN B_00 PATCH .....	7
1.2.1 <i>Offline Debugging Feature</i> .....	7
1.2.2 <i>Direct ICE</i> .....	7
1.3 MOST RECENT CORRECTIONS AND ENHANCEMENTS .....	8
1.3.1 <i>Installation</i> .....	8
1.3.2 <i>Corrections and Enhancements in zCui</i> .....	9
1.3.3 <i>Corrections and Enhancements for zFAST Synthesis</i> .....	11
1.3.4 <i>Corrections and Enhancements for zRtlFrontEnd</i> .....	12
1.3.5 <i>Corrections and Enhancements for ZeBu backend compilation</i> .....	12
1.3.6 <i>Corrections and Enhancements for Runtime</i> .....	13
1.3.7 <i>Corrections and Enhancements in zRun</i> .....	14
1.3.8 <i>Other Corrections and Enhancements</i> .....	15
1.4 SPECIFIC LIMITATIONS FOR PATCH B_00 .....	15
<b>2 NEW FEATURES IN V6_2_0.....</b>	<b>16</b>
2.1 MULTI-USER ENVIRONMENT.....	16
2.1.1 <i>Recommendations for Installation</i> .....	16
2.1.2 <i>New Calibration Process</i> .....	17
2.1.3 <i>Generation of the Configuration File</i> .....	17
2.1.4 <i>Compiling a design for relocation</i> .....	17
2.1.5 <i>Runtime settings for identification of the user</i> .....	18
2.1.6 <i>Runtime settings for relocation</i> .....	18
2.2 ZDPI FEATURE .....	19
2.3 NEW VERSIONS OF THIRD-PARTY TOOLS.....	19
2.3.1 <i>New version of ISE software</i> .....	19
2.4 zCUI NEW FEATURES.....	20
2.4.1 <i>FPGA settings</i> .....	20
2.4.2 <i>HTML Browser</i> .....	20
2.5 zRTLFRONTEND NEW FEATURES.....	21
2.5.1 <i>Hierarchical References</i> .....	21
2.5.2 <i>Support for Combinational UDPs</i> .....	21
2.5.3 <i>Better handling of the SystemVerilog interface</i> .....	21
2.6 ZEBU BACK-END COMPILATION NEW FEATURES.....	22
2.6.1 <i>Performance and Density</i> .....	22
2.6.2 <i>Enhanced usability</i> .....	22
2.6.3 <i>Dynamic Forces</i> .....	23
2.6.4 <i>Static timing analysis with glitch filtering</i> .....	23
2.6.5 <i>Reducing the critical paths with zPar_timing</i> .....	23



2.6.6	Automatic clustering .....	24
2.6.7	Improvement for the declaration of blackboxes .....	24
2.6.8	Modification for the declaration of probes.....	24
2.7	COMPATIBILITY OF A DESIGN WITH A ZEBU-SERVER SYSTEM.....	25
2.8	SUPPORT FOR ZEBU MEMORY IPS .....	25
2.9	ADD-ONS FOR THE C / C++ API .....	26
2.9.1	Support of C-Calls on Flexible Probes .....	26
2.9.2	Sampling Clock Selection for Flexible Probes .....	30
2.10	SMART Z-ICE INTERFACE WITH 4 OR 5 PORTS .....	30
<b>3</b>	<b>DOCUMENTATION PACKAGE .....</b>	<b>31</b>
3.1	MASTER DOCUMENT IN THE DOCUMENTATION PACKAGE .....	31
3.2	ZEBU-SERVER DOCUMENTATION PACKAGE .....	31
3.3	ADDITIONAL DOCUMENTS .....	32
3.4	UPDATED LICENSE AGREEMENT .....	32
<b>4</b>	<b>KNOWN LIMITATIONS IN V6_2_0 .....</b>	<b>33</b>
4.1	MODIFIED DEFAULT SETTINGS VS. V6_1_1.....	33
4.1.1	Default Number of Clock Buffers.....	33
4.1.2	Name of the compilation directory for the default back-end .....	33
4.1.3	Additional Error message in zFAST when inferring memories .....	33
4.1.4	Modified Settings for Automatic Clustering in zCui .....	33
4.1.5	New naming conventions for the units and modules .....	34
4.1.6	Deprecated option for probe declaration.....	34
4.1.7	Modified option to generate the list of faulty LVDS pairs.....	34
4.1.8	Change of behavior in transaction-based verification .....	34
4.2	NEW LICENSE VERSION FOR ZEBU-SERVER.....	34
4.3	LIMITATIONS FOR OPERATING SYSTEM .....	35
4.3.1	Specific SUSE Linux Enterprise Server 10 Requirement.....	35
4.3.2	Red Hat Enterprise Linux 5.1 and 5.2.....	35
4.3.3	GTKWave Limitation.....	35
4.4	zCUI LIMITATIONS.....	35
4.4.1	Project File Compatibility .....	35
4.4.2	Archive feature is not available in zCui .....	36
4.4.3	zCui uses /bin/sh and ignores .cshrc aliases/functions .....	36
4.4.4	Limitation for probe declaration with multiple RTL Groups.....	36
4.5	zFAST LIMITATIONS.....	36
4.5.1	Functional Limitations.....	36
4.5.2	Interface Limitations with zCui .....	37
4.5.3	Memory Synthesis Limitations with zFAST .....	37
4.6	BACK-END COMPILATION LIMITATIONS.....	38
4.6.1	Limitation for tristate signals .....	38
4.6.2	Clock connection to the SRAM_TRACE driver .....	38
4.6.3	Deprecated command in zNetgen.....	38



4.6.4	Limitations for Memory Modeling .....	38
4.7	RUNTIME LIMITATIONS.....	39
4.7.1	Trace feature.....	39
4.7.2	Clock declaration for HDL co-simulation .....	39
4.7.3	Limitation for threadsafe environment .....	39
4.7.4	Fast hardware state limitation on V6_2_0 .....	39
4.7.5	Limitation for Selection/Deselection of Dynamic Probes.....	39
4.7.6	Limitation for Selection of Nets or Ports of the Design at Runtime.....	40
4.7.7	Limitation on clock specification for restore.....	40
4.7.8	zInstall multi-release limitations .....	40
4.8	RUNTIME RESTRICTIONS TO WRITE OPERATIONS.....	40
4.8.1	Restrictions to Write to Registers .....	40
4.8.2	Restrictions to Write to BRAM Memories .....	40
4.8.3	Restrictions to Save & Restore.....	41
4.9	ADVANCED TRIGGERS .....	41
4.10	ZEMI-3 LIMITATION FOR SYNTHESIS.....	41
<b>5</b>	<b>FIXED ISSUES .....</b>	<b>41</b>
<b>6</b>	<b>VERSION COMPATIBILITY .....</b>	<b>42</b>
6.1	SUPPORTED PC CONFIGURATIONS.....	42
6.2	OPERATING SYSTEM COMPATIBILITY .....	42
6.3	HARDWARE COMPATIBILITY .....	42
6.4	THIRD-PARTY TOOLS COMPATIBILITY .....	43
6.5	EVE VERTICAL SOLUTIONS COMPATIBILITY.....	44
6.5.1	ZeBu Transactors.....	44
6.5.2	ZeBu Memory IPs.....	44
6.6	ZEBU LICENSES .....	44
<b>7</b>	<b>EVE CONTACTS.....</b>	<b>45</b>



# About this document

This Release Note describes the major features available for Version 6\_2\_0 release of the ZeBu-Server software with cumulative patch B\_00 (dated 22 February 2010).

Note that the present software version is intended for ZeBu-Server only and is not intended for use with ZeBu-XL, ZeBu-XXL, ZeBu-UF or ZeBu-Personal.

This document is intended for users who are familiar with the ZeBu product range.

You can find press releases, useful white papers and technology documents on our website: <http://www.eve-team.com>.



# 1 Specific Information for Patch B\_00

## 1.1 Installation Recommendations

This qualified cumulative patch obsoletes any previous V6\_2\_0 patch. It includes all the files for installation. Software installation of the ZeBu V6\_2\_0 base release is NOT a prerequisite to install the B\_00 cumulative patch.

- After installation, a text-formatted patch note is also available in the \$ZEBU\_ROOT/version/patch.txt file. It lists the most recent corrections and enhancements (not already delivered in previous patches) and gathers relevant information of intermediate patches since V 6\_2\_0.
- If V6\_2\_0 software was installed on the ZeBu-Server system, it is not necessary to proceed with all the steps of the installation process (setup of the ZeBu-Server system and generation of the configuration file for compilation are not mandatory unless you use the Smart Z-ICE or Direct ICE interfaces). Only the software installation, configuration of end-user environment and **zUtils** - **initSystem** steps have to be done. See the *[ZeBu-Server Installation Manual](#)* (Rev c) for details.
- It is not necessary to recompile the design with B\_00 cumulative patch when it was previously compiled for V6\_2\_0. However, recompilation is mandatory when migrating from older software versions and is recommended in any case to take advantage of the improvements of this release.
- This patch includes version 11.3i\_patched of the Xilinx ISE software ZeBu-specific subset for FPGA Place & Route. Note that Xilinx ISE 11 requires specific Xilinx licenses which should be requested from EVE (license@eve-team.com).

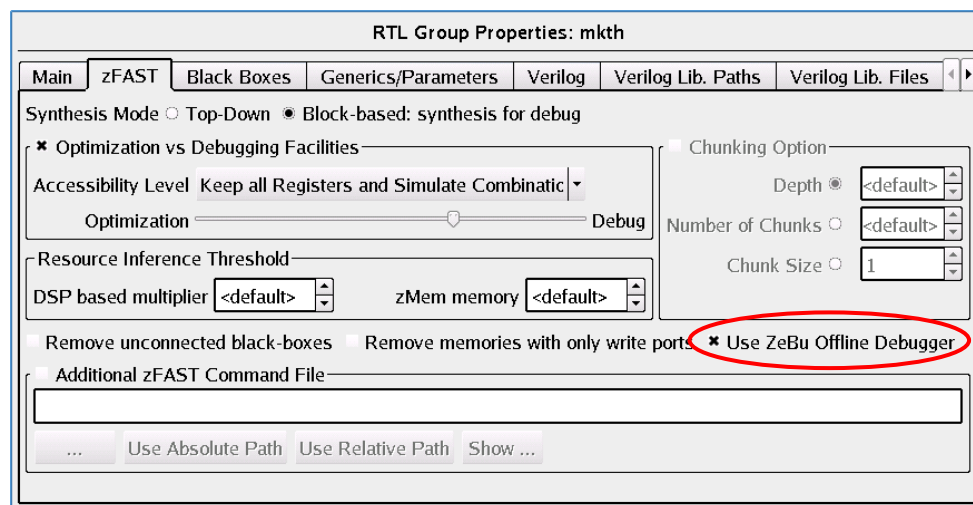


## 1.2 Major New Features in B\_00 Patch

### 1.2.1 Offline Debugging Feature

When synthesizing with **zFAST**, ZeBu-Server offers a new debugging use model, called Offline Debugging, where all the debugging features are enabled by the ZeBu compiler for runtime access without speed restriction.

The Offline Debugging feature is activated in the **RTL Group Properties** → **zFAST** panel:



Offline Debugging offers the capability to sniff the stimulation of the design at cycle level during emulation and to have this stimulation re-injected later on in emulation. For that purpose, the DVE should be modified manually with the instantiation of the sniffer transactor.

This feature will be described with examples in a dedicated Application Note. Please contact EVE support for details.

This feature requires some separate license features for compilation and runtime.

### 1.2.2 Direct ICE

This patch introduces the support of the Direct ICE interface. However, some of the corresponding features are not yet available and will be released later on.

You should contact EVE support if you intend to use the Direct ICE features.

An application note will be available in a near future to describe how to migrate from a ZeBu-XXL Direct ICE environment to a ZeBu-Server Direct ICE environment.



### 1.3 Most Recent Corrections and Enhancements

#### 1.3.1 Installation

##### 1.3.1.1 New name for the setup tool

In order to avoid any confusion with the **zUtils** `-initsystem` option, the tool dedicated to the setup of the ZeBu-Server system has been renamed into **zSetupSystem** (it was **zInitSystem** in V6\_2\_0).

The ***ZeBu-Server Installation Manual*** (Rev c) and Section 2.1.1 of the present Release Note have been updated to match this modification.

This modification was requested in RT#21629.

##### 1.3.1.2 Modification of the template file for setup

The `setup_template.zini` file has been updated with **zSetupSystem** and Smart Z-ICE and Direct ICE voltage settings.

##### 1.3.1.3 Settings for ISE software

Note that the path to ISE software is no longer added in `$PATH` by the `zebu_env.bash` script. A specific environment variable is set so that the ZeBu compilation tools use the ZeBu-specific version of ISE and launch it from the appropriate directory.

##### 1.3.1.4 New version of GateVision® PRO

A new version of GateVision PRO netlist analyzer (V4.6.5), developed by Concept Engineering, is provided in the ZeBu software package. Installation of Concept Engineering tools is proposed during the configuration script for ZeBu software installation.

GateVision PRO can be used directly from **zCui**, using the corresponding toolbar buttons after a global netlist is available. After installation, user documentation for Concept Engineering tools is available in both html and PDF formats in the following directory: `$ZEBU_ROOT/cevision-4.6.5/doc`.

##### 1.3.1.5 Updated list of sample configuration files

The list of sample configuration files delivered in the package has been updated, in particular to support the 8C/Direct ICE module and to provide sample files for the most popular configurations for 2-slot single-unit systems, 5-slot single-unit systems and 5-slot multi-unit systems. These files are available after installation in the `$ZEBU_ROOT/etc/configurations` directory.

The complete list of sample configuration files has been updated in the ZeBu-Server Installation Manual (Rev c).





### 1.3.1.6 Improvement for multi-release support

To improve the capability to run **zKernel** and **zInstall** for different software releases on the same ZeBu-Server system, the memory allocation for the communication between the PC and the ZeBu system has been modified.

If you intend to run several software versions on your ZeBu system, it is recommended to follow this procedure in order to allocate 64Mbits of the system memory for the ZeBu-PC communication:

1. Modify the boot command line to set the memmap option (with size\$address parameter), for example memmap=64M\$64M.
2. Restart the PC.
3. Launch **zInstall** with the -bootAllocAddr option and force the base address. When memmap is set as in the above example, the address to map the syntax should be (64M = 0x4000000):

```
$ zInstall -bootAllocAddr 0x04000000
```

### 1.3.2 Corrections and Enhancements in zCui

#### 1.3.2.1 Interactive ISE summary

It is now possible to get information about the FPGA compilation process while the process is running. This table can be accessed during the compilation from the menu bar: **Tools** → **Interactive FPGA summary**.

	S	Physical ID	Logical ID	Z Reg	X Reg	Z LUT	X LUT	Z Bram	X Bram	Z DSP48	X DSP48	Z Bufg	X Bufg	Z Set	X Set	Directory
Route U0_M0_F14		Unit0:Module0:Fpga14		961	929	962	772	2	2	0	--	10	4	79	63	backend_default/U0/M0/f
U0_M0_IF		Unit0:Module0:Fpga17		--	--	--	--	--	--	--	--	--	--	--	--	backend_default/U0/M0/f
U0_M0_F16		Unit0:Module0:Fpga16	Z_DEFAULT_CORE	35304	--	69881	--	12	--	29	--	13	--	792	--	backend_default/U0/M0/f
Route U0_M0_F15		Unit0:Module0:Fpga15		961	929	962	772	2	2	0	--	10	4	79	63	backend_default/U0/M0/f

The FPGAs are listed with two different identifiers: the physical identifier gives the unit, module and FPGA location; the logical identifier includes the zCore which is mapped on the FPGA. The last columns of the summary shows in which directory the compilation files are stored.

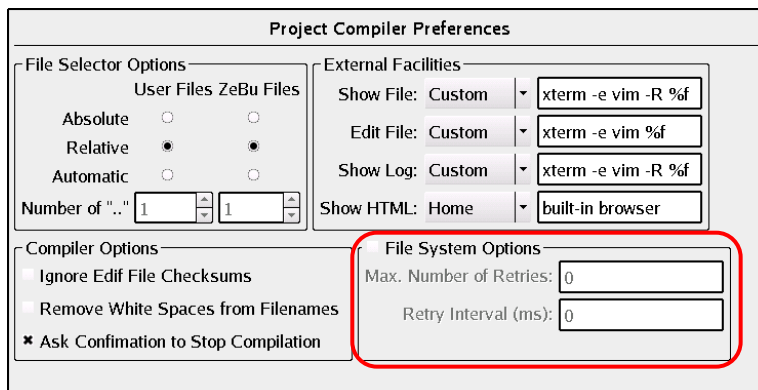
This summary includes information about the resources estimated by the ZeBu compiler (columns starting with a Z) and the resources estimated by Xilinx ISE (columns starting with an X). The listed resources are: registers, LUTs, BRAMs, DSPs, clock buffers (Bufg) and Registers with same SET/RESET connections (Set).

It also reports non-converging routing processes (in the S column) when 'intermediate status' message is returned by ISE.

## 1.3.2.2 Improvement of the retry mechanism for NFS failures

Some settings were added in **zCui** to improve the retry mechanism which sometimes caused the "Missing Inputs" failures because the files were not written in time on the network for the next compilation step.

The number of attempts to reach a file before returning an error and the delay (in ms) between two attempts can be set in the **Preferences** → **Compiler** → **File System Options** frame:



The screenshot shows the 'Project Compiler Preferences' dialog box. The 'File System Options' section is highlighted with a red rectangle. It contains two input fields: 'Max. Number of Retries' and 'Retry Interval (ms)', both set to 0. Other sections visible include 'File Selector Options' (with 'Relative' selected), 'External Facilities' (with 'Show File', 'Edit File', 'Show Log', and 'Show HTML' all set to 'Custom'), and 'Compiler Options' (with 'Ignore Edif File Checksums', 'Remove White Spaces from Filenames', and 'Ask Confirmation to Stop Compilation' all checked).

The values for these fields must be chosen with much care since they will impact the delay of error messages for files which are actually not available (delay = max nb retries \* interval).

The **zCui** full log file includes information which may help to determine appropriate values:

```
# Network File System Stats: Number of Retry = 0 ; Number Of Failure = 0 ; Average Number of Looping = 0 ; Maximum Number Of Looping = 0
```

- The Maximum number of Looping value is a correct basis for delay, but a security margin is recommended (5ms seems a reasonable margin).
- Average number of Looping: if this number is high (it should stay below 500), it means that interval should be increased in order to reduce the number of system calls).

The **zCui** full log is available in the **View** menu or by clicking the following icon: .

### Note:

This enhancement does not fix the case where **zCui** cannot detect that a file has changed because the file stamp is not updated between two consecutive compilation tasks. In such a case, you should check the **Tell Why?** contextual menu of the failing compilation task: if it shows "same date" but it is linked to an incrementality issue.



### 1.3.2.3 Other Corrections and Enhancements

- When synthesizing with **zFAST**, a panel has been added in the **RTL Group Properties** for declaration of blackboxes.
- The error message was improved when the front-end (transactor or STB) with `-f` (or `--frontend`) option in the command line does not exist in the `zpf` file (requested in RT#21568 and RT#21905):

```
Cannot start compilation
foo is not a known front end
```

### 1.3.3 Corrections and Enhancements for zFAST Synthesis

- The SystemVerilog synthesis was not correct for a packed array synthesized as latches (for example `bit [7:0][63:0]latch_mem;`): the msb addresses were stuck to ground, resulting in a removal of the corresponding address latches during the back-end compilation and the runtime results were not correct (reported in RT#21646).
- Verilog defparams were not taken into account when they were not defined as complete absolute paths (reported in RT#21881).
- The following issues were reported in RT#21612, RT#21683 and RT#21580:
  - Using the `Compile:MakeSyncWrites` attribute on a continuously written memory generated a stack trace.
  - The following fatal error was thrown while synthesizing Verilog or SystemVerilog code:

```
### fatal error in ZFAST [110.6552] : '/ztools-Z-1-
5/Code/include/Thinterra/TeNomNode.h:933':NULL value passed for parameter of type
nomNet *.
```

- When a memory was inside a SystemVerilog struct, **zFAST** failed with the following error message (reported in RT#21801):

```
Internal Compiler Error (11), Compiler Terminates [2962]
```

- When doing a port implicit connection in the instantiation of a module with SystemVerilog structs at the interface that are declared in a package, **zFAST** failed with the following error message (reported in RT#21670):

```
### fatal error in ZFAST [97.687] : 'file.sv:10': Illegal output/inout port ( out )
specification for instance ( il ).
```

- **zFAST** was throwing an error on modules that were provided but were not instantiated (reported in RT#21944).
- The generation of the memory init file with the `compile:rirm=true` attribute was not supported and caused the following error message (reported in RT#21925):

```
### fatal error in ZFAST [97.361] : 'src/dut.sv:14': Procedural assignment statement
cannot drive a net : err_signal.
```



- When synthesizing a module instance array in SystemVerilog, **zFAST** failed with the following error message (reported in RT#21991):

```
### fatal error in ZFAST [97.481] : '../.../top.sv:5': Syntax error near ( [ ] ).
```

### 1.3.4 Corrections and Enhancements for **zRtlFrontEnd**

- The capability to use **zMem** memory templates with **zRtlFrontEnd** is now integrated in **zCui** (Requested in RT#21267).
- BRAM inference sometimes failed due to an inappropriate strategy in Synplify synthesizer. **zRtlFrontEnd** now detects multiple-port memories and sets the synthesis option for Synplify synthesizers (reported in RT#18017).
- Support of SystemVerilog typedef and structs sometimes caused the following fatal error (reported in RT#21632):

```
### fatal error in RTL [RFE0058F]
```

- When generic VHDL parameter were used in the range of a port, **zRtlFrontEnd** elaborated code was not correct and Synplify failed with the following error message (reported in RT#21673):

```
@E:'zcui.work/design/synth_Default_RTL_Group/src/dut.vhd':22:53:22:53|  
No identifier datasize in scope
```

- zRtlFrontEnd** now generates the correct entity with a port and a range type.

### 1.3.5 Corrections and Enhancements for ZeBu backend compilation

- In **zTime**, HTML reports are now limited to 10 files and they are saved in a directory called `ztime_html` (only the first HTML report remains in the backend compilation directory).
- Encrypted EDIF transactors netlists are now supported by the ZeBu-Server compilation flow.
- edf2x** has been improved so that wire or instance names with escaped characters ('\') are handled correctly (requested in RT#21577).
- Some clocks at the interface between zCores were not correctly dumped in the waveforms (reported in RT#22012): the clock signals were inverted in the waveform but the design actually worked correctly.
- In **zBrowser**, signals with escaped identifiers ('\') or with spaces in their name can now be selected for trace (requested in RT#21756).
- When a zCore was largely overmapping the FPGA resources, the zCore-level clustering failed with the following error message (reported in RT#21864):

```
### internal error [ACC0113E] : Cannot resize constraints parameters
```

The internal error has been replaced by a detailed log explaining the overmapping.



- The clock processing algorithms have been improved in terms of duration and memory resources for both system-level and zCore-level compilation.
- In incremental compilation mode, when using a previous auto-clustering result, the zCore-level compilation crashed with the following message (reported in RT#21633):

```
### internal error [LST0044E] : Module U0_M1_F3 already exists in library  
zsplit_lib_Z_DEFAULT_CORE
```

This was due to multiple names for a single FPGA:

```
defmapping U0.M1.F3 == UNIT0.MOD1.F3
```

- The mapping constraints generated by automatic clustering for some of the design memories were not correct and the next incremental compilation failed with the following error message (RT#21928 and RT#21886):

```
### internal error [PRO0297E] : ModuleTab[1]->Netlist pointer is NULL.
```

- When **Keep all Registers and Simulate Combinational Signals** was selected in **zCui**, the **Build Accessibility Graph** step of the compilation did not finish after hours and the last displayed message was (reported in RT#21941):

```
Constant propagation took less than 1s
```

### 1.3.6 Corrections and Enhancements for Runtime

- When the memory content file (Verilog file) declared for memory initialization was empty, the memory parser failed (reported in RT#21954).

The following warning message is now displayed in such a case:

```
WARNING : The file '../run_zebu/memory_content.txt' used to load memory 'dut.my_bram'  
is empty.
```

- When a comment line (starting with '#') was empty in the file which lists the memory to be initialized (the file declared in the designFeatures file), the emulation failed (RT#22017). Such an empty comment line is now parsed with no error or warning.
- When the logic state file could not be read, **zState** caused a segmentation fault (reported in RT#20611). The ZSTAT076E error now stops **zState** correctly and the following message is displayed:

```
ZSTAT0076E : Cannot read logic state file .
```

- A segmentation fault occurred when declaring a file with an incorrect extension to dump dynamic probes. This issue was detected in **zRun** (reported RT#21376) but it was applicable to other runtime environments as well.

The runtime software now displays the following error message:

```
ZHW0810E: Cannot dump file "<filename>", unexpected file extension
```

The correct file extensions for dynamic probes are .vcd, .fsdb, .vpd and .bin.

- Local tracer disabling is now forced at hardware state save (reported in RT#21486).



- The runtime error messages for the following cases have been modified:
  - When the issue may be fixed by selecting the **Save and Restore - Write Back** option of **zCui**.
  - The 'exchanged' typo has been fixed.
- In threadsafe environment, a deadlock sometimes occurred when one thread tried to read or write a memory while another thread was executing the `ZEBU_Driver_run` or `ZEBU_Driver_wait` function.
- In HDL co-simulation, the simulator is now stopped by the PLI when the connection to ZeBu fails. An error message was previously returned by the simulator.
- The following improvements are available in the C/C++ APIs:
  - It is now possible to force internal nets while emulation is running.
  - It is now possible to control the value change trigger.
  - It is now possible to access parts of signals and parts of memories.
- When some simulated combinational signals were selected, runtime sometimes failed with the following error message because of multi-driven net issues (reported in RT#21916):

```
### internal error [SEG0079E] : Cannot find node CkGridM1N00 in the graph lcpubox  
### internal error [SEG0079E] : SupportAndConeFinder.cc, line 171 : _findArcInGraph
```

### 1.3.7 Corrections and Enhancements in zRun

- In the **zRun** GUI, all clock counters now start at 0. Previously it could not be so when the master clock was not the fastest clock.
- In the **zRun** GUI, the flexible probes clock can no longer be selected when the design uses SVAs (requested in RT#21262).
- The error message thrown when two processes attempt to validate SVAs (e.g. **zRun** + testbench) has been improved (requested in RT#21265): process names are now displayed along with their PIDs, as shown in the following example:

```
ERROR : LUI0788E : 'zRun' (pid 7235) cannot enable Flexible Local Probes or SVA,  
because 'tb_PP' (pid 7234) has already enabled this feature.
```

- It is now possible to deselect all the signals previously selected via **zSelectProbes** or **zDbPostProc** from a **zRun** Tcl script with the `ZEBU_Signal_deselectAllSignals` command.
- It is now possible to start a clock while dumping dynamic probes into a waveform file.
- When **zRun** GUI was launched with `-testbench` option, saving a hardware state caused a segmentation fault (reported in RT#21771).
- It is now possible to force internal nets with **zRun** Tcl commands while emulation is running.





- The **zRun** Tcl command to dump a waveform file (ZEBU\_Dump\_file) has been upgraded. It previously accepted two arguments only.  
It now accepts an optional third argument which makes it possible to add all selected signals in the waveform file. This argument is the name of a file which lists the signals to dump. In this file, each line contains the hierarchical name of a signal to be dumped and ends with an EOL character, i.e. a line does not contain any space character.

### 1.3.8 Other Corrections and Enhancements

- A Smart Z-ICE input clock can now be connected to several modules in the same unit.
- Note that the setting of the Smart Z-ICE voltage has to be declared for the setup phase of the ZeBu-Server System and is stored in a file in the ZeBu system directory (\$ZEBU\_SYSTEM\_DIR).  
The default value for the Smart Z-ICE voltage is 1.5 V.  
If you need another voltage level for the Smart Z-ICE interface, the `setup_template.zini` file must be modified and you have to relaunch the setup phase with **zSetupSystem**.

## 1.4 Specific Limitations for Patch B\_00

- On 5-slot units, connector P4 of the Smart Z-ICE interface cannot be used to connect data. The runtime fails with the following error message:

`Unexpected parameter : if_con 0 connector 4 (max = 3-4)`
- In a multi-unit system, it is not possible to connect a Smart Z-ICE input clock on any unit other than U0. No message is thrown during compilation.
- BRAM-based memories sometimes use twice the number of BRAMs they would normally require because of a Virtex-5 problem.



## 2 New Features in V6\_2\_0

This section describes new features in V6\_2\_0 software, with respect to ZeBu release V6.1\_1.

### 2.1 Multi-User Environment

Adding multi-unit and multi-user features to ZeBu-Server has implied several modifications regarding installation, calibration, compilation and runtime processes.

#### 2.1.1 Recommendations for Installation

The following procedure summarizes the various steps from installing the ZeBu software to initializing the ZeBu-Server system. This procedure includes references to detailed information in the [ZeBu-Installation Manual](#).

1. Install the Zebu software (see Chapter 4 in the [ZeBu-Server Installation Manual](#)).
2. Configure the end-user's environment (see Chapter 5 in the [ZeBu-Server Installation Manual](#)).
3. Run **zInstall** from each host PC connected to the ZeBu-Server system to initialize the PCIe board and load **zKernel** in the Linux Operating System (see Chapter 6 in the [ZeBu-Server Installation Manual](#)).
4. Initialize the ZeBu-Server system:
  - a. After each configuration change, perform a system setup for the entire ZeBu-Server system (system calibration and diagnostics to generate a configuration file). For that purpose, launch the following command from any PC connected to unit U0:  

```
$ zSetupSystem <setup.zini>
```
  - b. After each power off/power on, load the backplane(s) and the modules' system FPGAs. For that purpose, launch the following command from any PC connected to unit U0:  

```
$ zUtils -initSystem
```

Note that in a multi-unit environment, U0 is the unit which is connected to the HUB through C4 connector, as described in Section 1.7 in the [ZeBu-Server Installation Manual](#).





### 2.1.2 New Calibration Process

In previous releases, the /zebu directory of the PC connected to a ZeBu unit was used to store the diagnostics and calibration data for runtime.

As of release V 6\_2\_0, the directory which includes diagnostics and calibration data is attached to the whole ZeBu-Server system and it should be in an appropriate location to be accessed by all the PCs connected to the ZeBu-Server system. This directory is created during the installation process and has to be declared with an environment variable (\$ZEBU\_SYSTEM\_DIR) before proceeding with emulation runtime.

### 2.1.3 Generation of the Configuration File

#### 2.1.3.1 New options for zConfig

To support multi-unit systems and multi-user environment, the generation of the configuration file with **zConfig** has been improved as described in Chapter 9 of the *ZeBu-Server Installation Manual*.

The following table shows the new options which have been added in **zConfig** and gives a brief description for them:

-mi	can be used (with the -cfg option) to generate a configuration file for non-identical ZeBu-Server systems
-merge	can be used (with the -cfg option) to generate a configuration file which supports relocation for a multi-user environment.
-default	can be used to generate an accurate default configuration file
-nbfpga	can be used to generate an average default configuration file

**Note:** The -status option is now the only option available in V6\_2\_0 to generate the file which lists the faulty LVDS pairs (the -lstatus option which was in V6.1\_1 has been deprecated).

#### 2.1.3.2 Available sample configuration files

The list of sample configuration files has been modified for cumulative patch B\_00. These files can be used for compilation, but it is **recommended** to compile with a specific configuration file for your actual system before proceeding with runtime.

### 2.1.4 Compiling a design for relocation

The ZeBu compiler supports the declaration of the module(s) on which the design should be map (use\_module command). Declaring a limited number of modules is mandatory for an efficient multi-user environment since the design is compiled by default to be mapped on all the modules.

By default, a design compiled for relocation (with use\_module commands) will automatically support relocation for multi-user environment.

This feature is fully described in Section 4.1.4 in the *ZeBu-Server Compilation Manual* (Rev b).



### 2.1.5 Runtime settings for identification of the user

In order to differentiate the sessions in a multi-user environment, each user of a ZeBu-Server system has a unique identifier. This identifier is intended to differentiate the emulations running on a system and to avoid that the same user connects to the same system from several PCs.

This user identifier is an integer (maximum is 255) declared with the ZEBU\_SRD\_SET\_RUNID environment variable which has to be set before launching emulation runtime.

If the user identifier is not set to an appropriate value before launching the emulation, **zServer** displays the following error message:

The SRD CPU runId should be 255 max (xxx).

Where xxx is the identifier declared by user.

### 2.1.6 Runtime settings for relocation

In a multi-user environment, a design which was compiled for relocation can be run on modules different from the reserved ones, assuming that the actual configuration of the ZeBu system makes it possible.

By default, the design will be run on the modules which were declared for the ZeBu compiler (use\_module command). If the expected modules are currently used for another design, the emulation will be queued by **zServer**.

In order to run the design on other modules of the system, user must set the ZEBU\_M0\_PHYSICAL\_LOCATION environment variable before launching emulation runtime. This environment variable changes the "physical" location of the logical "M0" module resulting from the compilation of the design.

If information is needed to know which modules can be used for M0, the user should check in the loc.xref file (in the zcui.work/zebu.work/tools/zPar/ directory): the modules which are authorized for declaration are the one listed in the \$U0.M0.physicalLoc line of the file.

When initializing the emulation, the module declared in the ZEBU\_M0\_PHYSICAL\_LOCATION environment variable is checked for coherence with the information resulting from the compilation of the design and with the actual ZeBu system. **zServer** provides the following message to know on which module the design is actually mapped:

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



### Example:

For a design which uses 1 module compiled on a 2-unit system with 3 modules, the following line appears in the `loc.xref` file:

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

This line means that the logical M0 of the design can be mapped on U0.M0 or U0.M1 or U1.M1.

When the emulation is launched from the PC connected to U0, user can declare one of the following values:

```
$ export ZEBU_M0_PHYSICAL_LOCATION=M0
```

or

```
$ export ZEBU_M0_PHYSICAL_LOCATION=M1
```

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

```
$ export ZEBU_M0_PHYSICAL_LOCATION=M0
```

## 2.2 zDPI Feature

When synthesizing with **zFAST**, ZeBu now supports DPI import function calls with inputs only in the SystemVerilog source files of the design. Note that context import functions are not supported.

DPI calls are synthesized by **zFAST** and a user-written dynamic library is integrated for runtime usage. Control from the ZeBu runtime environment is possible through **zRun** (Tcl commands) or using a C/C++ API.

The compilation options for zDPI are available in a new **zDPI** tab in **zCui** (in the **RTL Group Properties** panel when synthesizing with **zFAST**).

This feature is fully described in a ZeBu Application Note (AN029).

## 2.3 New Versions of Third-party Tools

### 2.3.1 New version of ISE software

The ISE software 11.3i\_patched is now provided (64-bit version only).

As for previous versions, the ISE software delivered in the ZeBu package contains a ZeBu-specific version. XST synthesizer is not included in this package.

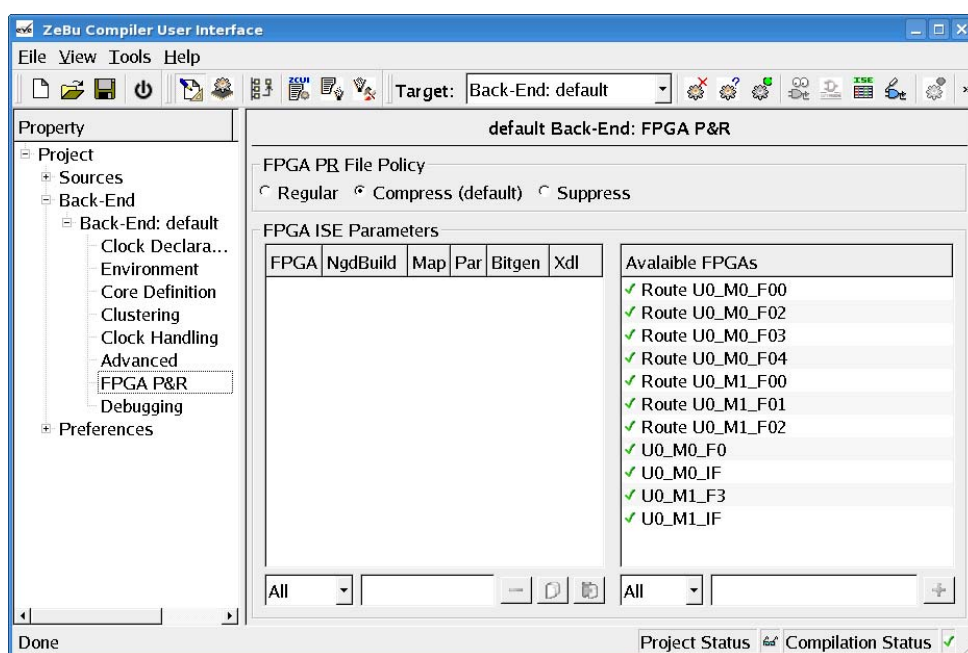
Note that this ISE 11 software version requires a specific license which should be requested from EVE, as described in the [ZeBu-Server Installation Manual](#).

## 2.4 zCui New Features

**zCui** has been updated to support the most recent features of the ZeBu compiler, in particular the support of zDPI feature with **zFAST**. The following sections list the new features for **zCui** itself, either for the graphical environment or the command-line options.

### 2.4.1 FPGA settings

A new panel has been added in the back-end environment to modify the FPGA Place & Route options for the FPGAs of the ZeBu system:



Note that the list fields in this panel are empty when no compilation has been previously run; their content is updated when the design is compiled.

Tooltips are available for this panel in **zCui** and this feature is described with details in Section 5.3.1 of the *ZeBu-Server Compilation Manual* (Rev b).

### 2.4.2 HTML Browser

It is now possible to choose among different HTML browsers for the compilation report files. In the **Preferences** → **Compiler** panel, the HTML browser spin box offers the following values:

- **Home:** **zCui** built-in browser
- Konqueror
- Firefox
- **Custom:** any HTML browser can be declared by user. Note that %f is mandatory in the command line to have the appropriate file opened.



## 2.5 zRtlFrontEnd New Features

### 2.5.1 Hierarchical References

Hierarchical references are now supported in both designs and synthesizable testbenches source files.

For that purpose, **zRtlFrontEnd** modifies the interface of all the modules which are traversed by the hierarchical reference before generating the elaborated code for the RTL synthesis tools. Synthesis tools keep the same interface during synthesis and the hierarchical reference becomes part of the resulting merged gate-level netlist.

#### Notes:

- **zRtlFrontEnd** supports hierarchical references which refer to or cross a VHDL entity when referring to an underlying Verilog module.
- Hierarchical references strongly impact the incrementality of the design compilation because all the components which are crossed by hierarchical references must be recompiled for each modification (addition, modification, removal of a hierarchical reference).

#### Limitation:

- **zRtlFrontEnd** does not support hierarchical references on the left side of a blocking or a non-blocking assignment.

### 2.5.2 Support for Combinational UDPs

Combinational UDPs are now supported (sequential UDPs are not supported). Any Verilog UDP is now replaced by LUT number.

### 2.5.3 Better handling of the SystemVerilog interface

- SystemVerilog interface `modPort` for synthesis script generation is now supported.
- The real names of the renamed module and of the renamed SystemVerilog interface names are now retrieved in the synthesized netlists.



## 2.6 ZeBu Back-End Compilation New Features

### 2.6.1 Performance and Density

In order to improve the compilation process as well as the runtime performance, the ZeBu compiler has been improved with respect to V6\_1\_1, in particular in terms of FPGA allocation, clustering, and clock handling.

Note that the execution time has been reduced for both system-level and zCore-level compilations.

### 2.6.2 Enhanced usability

#### 2.6.2.1 Improvement for the system-level compilation command files

In case some commands which should be in the **Netlist Edition File** are included by mistake in the **Additional Command File**, they will be executed correctly and a warning message will be issued. Ideally, user should move these commands to the appropriate file for future compilations.

The error messages have been improved when a command cannot be executed correctly because it is in the wrong command file.

#### 2.6.2.2 Improvement of the zTopBuild\_report files (System-level compilation)

Some additional information is available in the the zTopBuild\_report.log and zTopBuild\_report.html files.

In the Pre-clustering section (Section 3), the following subsections have been added:

- 3.1.Summary pre clustering
- 3.2.Multiple instantiation
- 3.3.Blackbox modules
- 3.4.Blackbox instances
- 3.5.ZMem resource utilization
- 3.6.Detection of sub-system candidates
- 3.7.Clock report

In the User and netlist tristate directives section (Section 5), the following subsection has been added:

- 5.1.Tristate global parameters.

Section 12 (Post-split statistics) has been added with the following subsection:

- 12.1.Core interconnections



### 2.6.3 Dynamic Forces

It is now possible to declare signals (`-net` option) or ports (`-pin` option) for the `force_dyn` command in the **Build System** → **Additional Command File**.

This feature is described with details in Section 3.7.5 of the *ZeBu-Server Compilation Manual* (Rev b).

### 2.6.4 Static timing analysis with glitch filtering

When the **Filter Glitches** feature is activated in **zCui** (which is the default setting in the **Clock Handling** panel), the static timing analysis process now estimates the `zClockFilterTime` value (runtime programming for the filters) and takes the filters into account for a better estimation of `zClockSkewTime`.

A new report is also available which lists the longest paths on which filters are inserted and provides statistical information for these paths. This HTML report (`ztime_filter_out_paths.html`) is very similar to the HTML report on clocks paths.

Two constraint files are also available which can be used to optimize the routing for the most critical nets during routing, thus enhancing the achievable driverClock performance:

- `ztime_critical_wires.tcl`: only the wires at the end of the longest paths
- `ztime_critical_path_wires.tcl`: all the wires of all most critical paths

These files can be sourced in an **Additional Command File** for the System Place and Route (**Advanced** panel, **System Place and Route parameters (zPar)** frame)

### 2.6.5 Reducing the critical paths with **zPar\_timing**

In order to optimize the system routing with the timing analysis results, a dedicated tool is now available for manual launching from the compilation directory (`zebu.work`):

```
$ zPar_timing
```

---

This tool launches **zPar** and **zTime** alternatively to reduce the most critical paths of the design and iterates as many times as necessary until no progress can be done.

This iterative process results into a file which contains the appropriate commands to route the most critical paths for the compiler. This `best_critical_cores.tcl` file is generated in the directory where **zPar\_timing** was launched.

It is recommended to copy this file in the same directory as the ZeBu-dedicated files (`<my_project>/src/zebu` for example) so that it is not removed in a **Clean** operation in **zCui**.

This file can be declared in **zCui** as an **Additional Command File** for the System Place and Route (**Advanced** panel, **System Place and Route parameters (zPar)** frame).





## 2.6.6 Automatic clustering

### 2.6.6.1 Modeling of zDelay insertion

The delays inserted for clock modeling (zDelay insertion) are now taken into account for automatic clustering.

The final table in `zCoreBuild.log` looks as follows:

#	FPGA	REG	LUT	BRAM	DSP	AC_REG	AC_REG	ZDELAY	AC_REG	OVERALL	STATUS
#	UNIT0.MOD0.F0	54808 / 207360	77368 / 207360	55 / 286	127 / 192	29718		25144		54862	OK
#	UNIT0.MOD0.F1	19636 / 207360	103477 / 207360	5 / 286	0 / 192	14309		9292		23601	OK
#	UNIT0.MOD0.F2	42444 / 207360	94340 / 207360	42 / 286	19 / 192	28415		19257		47672	OK

where :

- REG: total number of registers in the FPGA vs. FPGA register capacity
- AC\_REG ZDELAY: number of zDelay registers estimated by auto-clustering
- AC\_REG OVERALL: total number of registers estimated by auto-clustering

When the design only has primary clocks, there is no delay insertion and this new feature should be disabled to reduce the duration of the automatic clustering. For that purpose, the following command has to be added in the **Additional Command File** for the System Compilation (**Advanced** panel, **Build System** parameters (**zTopBuild**) frame):

```
cluster disable -ac_opt_zdelay
```

### 2.6.6.2 Reserved BRAMs taken into account

The BRAMs which are reserved to enhance the zrm memory reading/writing feature are now correctly taken into account in the automatic clustering and for the resource estimation.

## 2.6.7 Improvement for the declaration of blackboxes

When a `blackbox` command is declared in the **Netlist Edition File**, it is now possible to leave a wire unconnected in the resulting netlist by setting the `value` option to `NONE` as in the following example:

```
blackbox remove my_blackbox -value NONE
```

## 2.6.8 Modification for the declaration of probes

The `probe_signals` command has new options useful for DPI and C-Calls probe types, in order to support the zDPI feature. These additional options are:

`scope` (string): scope of a DPI import for C-Calls or DPI functions. If omitted, it is automatically inferred from design hierarchy.

`wrapper_name` (string): name of the C wrapper name for DPI function calls.

`import_name` (string): scope of a DPI or C call function. If omitted, the path in the hierarchy is assumed to be the scope.





Note that the `enable` option has deprecated for flexible probes, DPI, and C-Calls declarations, as described in Section 4.1.6.

## 2.7 Compatibility of a Design with a ZeBu-Server System

**zPinCheck** is a new tool which allows user to find out whether or not a compiled design can be run on a given Zebu-Server system, and on which module it can be mapped.

**zPinCheck** takes the following arguments:

```
$ zPinCheck -zebu.work <path_to_compilation_dir> -systemdir  
<path_to_sys_dir>
```

where:

- `-zebu.work <path_to_compilation_dir>`: path to the back-end compilation directory
- `-systemdir <path_to_sys_dir>`: path to the system directory (the same directory must be declared with the `ZEBU_SYSTEM_DIR` environment variable at runtime)

**zPinCheck** reads the following files:

- in the backend compilation directory (e.g. `zcui.work/zebu.work`):
  - `tools/zPar/loc.xref`: contains possible mappings of the design on the ZeBu-Server system.
  - `tools/zPar/design.pin`: contains all the pins used by the design.
- in the ZeBu-Server system directory (created via the `$ZEBU_SYSTEM_DIR` environment variable):
  - `config/status_0.tcl`: contains a description of the ZeBu-Server system configuration.
  - `config/status_*.pin`: lists the faulty LVDS pairs of the ZeBu-Server system (one file per ZeBu-Server unit)

For each possible mapping found in `tools/zPar/loc.xref`, **zPinCheck** informs whether the design can be mapped according to the lists of faulty LVDS pairs.

**Note:** The directory where the `status_*.pin` files are looked for (default is `systemdir/config`) is overwritten by the directory defined when the `ZEBU_STATUS_DIR` environment variable is exported (as in previous releases).

## 2.8 Support for ZeBu Memory IPs

The ZeBu memory IPs from EVE Vertical Solutions are now supported by ZeBu-Server.



## 2.9 Add-ons for the C / C++ API

### 2.9.1 Support of C-Calls on Flexible Probes

The C++ and C APIs have been updated in order to provide an easy access to flexible probes from a user testbench:

- `CCall.hh` header files provides the `ZEBU::CCall` class for the C++ API
- `ZEBU_CCall.h` header file provides the `ZEBU_CCall` struct for the C API.

The methods and functions for this feature are available for the following ZeBu libraries:

- `libZebu.so`
- `libZebuDebug.so`
- `libZebuThreadSafe.so`
- `libZebuThreadSafeDebug.so`.

Note that C-calls methods and functions should not be used with the `libZebuRestore.so`, `libZebuRestoreDebug.so` and `libZebuState.so` dynamic libraries although the corresponding symbols are defined in the header files.

#### 2.9.1.1 Select a sampling clock expression or a sampling clock group:

1. To select the clock group on which the function calls are sampled on emulation side, use the `SelectSamplingClockGroup` method:

```
static void SelectSamplingClockGroup
(Board *board, const char *clockGroupName = NULL)
throw(std::exception);
```

Where `board` is a C++ handler on Board; `clockGroupName` is the name of a controlled clock group (if NULL, the first arbitrary group is selected).

C-calls functions are sampled on all positive/negative edges of all the clocks in the selected group.

This method must be called before any C-call start.

2. To select a clock expression (set of clock signals and active edges) on which the C-call functions are sampled on emulation side, use the `SelectSamplingClocks` method:

```
static void SelectSamplingClocks
(Board *board, const char *clockExpression = NULL)
throw(std::exception);
```

Where `board` is a C++ handler on Board; `clockExpression` is a string with the following format (if NULL all clocks of the first arbitrary group and all edges are selected):

"[posedge|negedge] <clock name> [or [posedge|negedge] <clock name>] and so on"

This method must be called before any C-call start.

**Examples for `clockExpression`:**



"posedge clock1" → sampling on clock1's posedges

"posedge clock1 or negedge clock2" → sampling on clock1's posedges and clock2's negedges

"clock3" → sampling on clock3's posedges and clock3's negedges

### 2.9.1.2 Enable the synchronization of the C-calls:

In order to have a deterministic order when executing the C-calls during emulation, use the `EnableSynchronization` method:

```
static void EnableSynchronization  
(Board *board) throw(std::exception);
```

When the C-calls are synchronized, both the execution time (emulation time at which the C-call function is executed) and the call number (call order for a set of functions in the same scope) are both deterministic.

Without this synchronization, each C-call function is executed in an increasing time order corresponding to its executions on the emulation side; but the software side does not order function calls between each other and may cause time races between several C-calls functions.

#### Notes:

- Synchronization is disabled by default.
- Synchronization decreases runtime performance.
- This method must be called before any C-call start.

### 2.9.1.3 Select the C-call functions for event detection:

In order to activate the C-call function only when the sampled values for the corresponding probes change, use the `SetOnEvent` method:

```
static void SetOnEvent(Board *board) throw(std::exception);
```

#### Notes:

- Functions are called for each execution on the emulation side by default.
- This method must be called before any C-call function.

### 2.9.1.4 Load the dynamic libraries

The symbols of the C functions to import can be loaded with the `ZEBU::CCall::LoadDynamicLibrary` method.

```
static void LoadDynamicLibrary(Board *board,  
const char *fullname) throw(std::exception);
```

Where `board` is a C++ handler on `Board`; `fullname` is the name of the dynamic library, with the following pattern: [`<path>/`]`<library name>.so`.

If `<path>` is not specified, the `LD_LIBRARY_PATH` environment variable must contain the path where the `<library name>.so` file is located.

#### Notes:

- This method must be called before any C-call function.
- This can be called several times to load several libraries



### 2.9.1.5 Start and stop the C-call functions:

Each function call can be started or stopped with the `ZEBU::CCall::Start` and `ZEBU::CCall::Stop` methods. This enables or disables a set of function calls, knowing that all function calls are disabled by default.

The C-call functions can be specified by their scope or by a regular scope expression and/or their import name and/or their call number.

```
static void Start(Board *board, const char *scope = NULL, const char
*importName = NULL, const int callNumber = -1)
throw(std::exception);
```

---

Where:

board is a C++ handler on Board

scope is the scope of the C-calls function to enable (if null, all C-calls functions are enabled)

importName is the name of the C-calls function to enable (if null, all C-calls functions of the scope are enabled)

callNumber is the call number in the scope of the C-calls function to enable (if -1, all C-call functions of the scope are enabled)

This method can be executed several times to start a set of C-calls function calls.

```
static void Stop(Board *board, const char *scope = NULL, const char
*importName = NULL, const int callNumber = -1)
throw(std::exception);
```

---

Where:

board is a C++ handler on Board

scope is the scope of the C-calls function to disable (If null all enabled function calls are disabled).

importName is the name of the C-calls function to disable (if null, all C-calls functions of the scope are disabled)

callNumber is the call number in the scope of the function call to disable (if -1, all function calls of the scope are disabled)

This method can be executed several times to stop a set of C-calls function calls.

```
static void Start(
    Board *board,
    const char *scopeExpression,
    const bool invert = false,
    const bool ignoreCase = false,
    const char hierarchicalSeparator = '.',
    const char *importName = NULL,
    const int callNumber = -1
) throw(std::exception);
```

---

Where:

board is a C++ handler on Board

scopeExpression is a regular expression specifying the scopes of the C-calls function to enable.



`invert` inverts the criteria of the regular expression

`ignoreCase` disables case sensitivity in the regular expression

`hierarchicalSeparator` is the hierarchical separator character

`importName` is the name of the C-calls function to enable (if null, all C-calls functions of the scope are enabled)

`callNumber` is the call number in the scope of the C-calls function to enable (if -1, all C-call functions of the scope are enabled)

This method can be executed several times to start a set of C-calls function calls.

```
static void Stop(
    Board *board,
    const char *scopeExpression,
    const bool invert = false,
    const bool ignoreCase = false,
    const char hierarchicalSeparator = '.',
    const char *importName = NULL,
    const int callNumber = -1
) throw(std::exception);
```

Where:

`board` is a C++ handler on Board

`scopeExpression` is a regular expression specifying the scopes of the C-calls function to disable.

`invert` inverts the criteria of the regular expression

`ignoreCase` disables case sensitivity in the regular expression

`hierarchicalSeparator` is the hierarchical separator character

`importName` is the name of the C-calls function to disable (if null, all C-calls functions of the scope are disabled)

`callNumber` is the call number in the scope of the function call to disable (If -1, all function calls of the scope are disabled)

This method can be executed several times to stop a set of function calls.



### 2.9.2 Sampling Clock Selection for Flexible Probes

The `ZEBU_FlexibleLocalProbeFile_selectSamplingClocks` function and `ZEBU::FlexibleLocalProbeFile::selectSamplingClocks` method have been introduced for the declaration of the clock signal which is used to sample flexible probes, together with the active edge.

```
static void SelectSamplingClocks(Board *board, const char  
*clockExpression);
```

---

Where:

`board` is a C++ handler on `Board`;

`clockExpression` is clock sensitivity expression ("`[posedge|negedge]` `<clock name>` [`or [posedge|negedge]` `<clock name>`] and so on")

**Examples for `clockExpression`:**

"posedge clock1" → sampling on clock1's posedges

"posedge clock1 or negedge clock2" → sampling on clock1's posedges and clock2's negedges

"clock3" → sampling on clock3's posedges and clock3's negedges

### 2.10 Smart Z-ICE interface with 4 or 5 ports

The ZeBu-Server Smart Z-ICE interface provides a single-direction mode which uses the 4 or 5 ports of the interface:

- The Smart Z-ICE interface on 2-slot units has 4 connectors (P0 through P3) which provide a total of 64 data signals and 4 clock signals (16 data and 1 clock for each connector).
- The Smart Z-ICE interface on 5-slot units has 5 connectors (P0 through P4) which provide a total of 80 data signals and 4 clock signals (16 data for each connector and 1 clock for connectors P0 to P3).

The *[ZeBu-Server Smart Z-ICE Manual](#)* is now part of the documentation package and describes this interface in details for both hardware and software implementations.



## 3 Documentation Package

The documentation package has been updated for B\_00 cumulative patch.

### 3.1 Master Document in the Documentation Package

The ZeBu documentation package includes a master document which is a PDF file with bookmarks for direct access to all documents of the package (in the bookmark panel of Acrobat). This master document also includes one bookmark which opens the Acrobat Full Search window.

In the meantime, each document in the package includes one bookmark which returns to the master document (available at the top of the bookmark tab of Acrobat for this Release Note).

### 3.2 ZeBu-Server Documentation Package

The following table lists the complete documentation package, showing the manuals which are available for ZeBu-Server and mentioning which manuals can be used when ZeBu-Server dedicated manuals or up-to-date ZeBu common manuals are not available. Note that new or updated documents are in *italics*:

Ind.	Manual Name	Rev	Comments
01	<i>Installation Manual</i>	c	<ul style="list-style-type: none"><li>• Corrected an error in the max number of users for a 2-slot unit.</li><li>• Added a reference to the Direct ICE interface.</li><li>• Updated example of delivered file name for a 5-slot single-unit ZeBu-Server system to include one ICE module (ID=9).</li><li>• Changed name of system setup tool from <b>zInitSystem</b> to <b>zSetupSystem</b>.</li><li>• The runtime daily history file is now available for the last 15 days.</li><li>• Corrected section title 7.2.1.2, "single-unit" to "multi-unit".</li><li>• Fixed a typo in the path to <code>setup_template.zini</code>.</li><li>• Additional information for runtime settings in a multi-user environment.</li><li>• Updated list of sample configuration files for support of Direct ICE and 8C.</li><li>• Updated <code>setup_template.zini</code> (Appendix B).</li></ul>
02	<b>Compilation Manual</b>	b	Major update for V6_2_0. See detailed History table in the manual.





Ind.	Manual Name	Rev	Comments
03	HDL Co-Simulation Manual	b ZeBu	Last update for V 3.1_0.
04	C++ Co-Simulation Manual	b ZeBu	Last update for V4.3_0.
08	zRun Manual	c ZeBu	Last update for V4.3_3.
10	Smart Z-ICE Manual	a ZSE	New edition.
11	<i>Direct ICE Manual</i>	<i>a</i>	<i>First Edition</i>
13	C API Reference Manual C++ API Reference Manual	V6_1_1	
14	ZEMI-3 Manual	b ZeBu	Last update for V4.3_3.

### 3.3 Additional Documents

The following Application Notes can be used with the present software release:

Ind.	Application Note	Rev	Comments
AN029	DPI for ZeBu	a	

### 3.4 Updated License Agreement

For your reference, the updated license agreement for the ZeBu software and for usage of the third-party software packages delivered with the ZeBu software is available as a PDF file in the documentation package.

- The updated Xilinx license agreement is included.
- The Concept Engineering license agreement is also available for information about RTLvision PRO and GateVision PRO usage.





## 4 Known Limitations in V6\_2\_0

This chapter lists the known limitations in V6\_2\_0 software. If some of these limitations were no longer applicable for the B\_00 cumulative patch, they are not listed here. A dedicated section lists the specific limitations for the B\_00 cumulative patch.

### 4.1 Modified Default Settings vs. V6\_1\_1

#### 4.1.1 Default Number of Clock Buffers

The default number of clock buffers, `BUFGMAX`, has been reduced from 6 to 3.

#### 4.1.2 Name of the compilation directory for the default back-end

In an effort for coherence in **zCui**, the name of the default back-end became `default` and the name of the associated compilation directory (previously `zebu.work`) has been renamed to `backend_default`.

A symbolic link named `zebu.work` has been created in which the user can launch runtime tools.

**Warning:** In **zCui**, **Clean** operations must be launched on the entire project.

#### 4.1.3 Additional Error message in zFAST when inferring memories

When a memory has more than 128 ports of the same type, an error is now issued during synthesis. You have to investigate if this high number of ports is an expected value. In most cases it should not be.

The error message that user will find in **zFAST** log is:

ERROR: Memory [MEMNAME] has 0 read port(s), 0 write port(s), and 192 RW port(s) ERROR+ which is more than the port number limit (Compile:PortLimit = 128)
--

To modify the number of ports to trig this error message, the following attribute should be added in the **Additional zFAST Attribute File (Project → Sources → RTL Group Properties panel)**:

<code>Compile:PortLimit = &lt;maxval&gt;</code>
---

Where `<maxval>` is the maximum number of ports which is accepted for a memory.

#### 4.1.4 Modified Settings for Automatic Clustering in zCui

When **Automatic with Parameters** mode is selected, the default filling rates have been modified (compared to V6\_1\_1 default settings):

- Registers: 55% (5% overflow activated by default)
- LUTs: 70% (10% overflow activated by default)
- DSP: 100%
- BRAM: 80%



### 4.1.5 New naming conventions for the units and modules

In order to support multi-unit environments, the naming conventions have been modified: `U0.M0` should be used for Module 0 of Unit 0 (it used to be `B0`).

This change is applicable in particular in the compilation filetree and when declaring commands in the `designFeatures` file.

### 4.1.6 Deprecated option for probe declaration

The `enable` option has been deprecated for flexible probes, DPI, and C-Calls declarations.

The appropriate syntax is `enable_wire` or `enable_port` options for probes declared on wires or ports.

If a probe file for a previous software version includes the `enable` option, it is automatically interpreted as `enable_port` (and an error message is displayed if the instance is not a port).

### 4.1.7 Modified option to generate the list of faulty LVDS pairs

In `V6_2_0`, user can only generate the list(s) of faulty LVDS pairs locally (using the `-status` option); `-lstatus` option has been deprecated.

### 4.1.8 Change of behavior in transaction-based verification

Transaction-based testbenches designed on ZeBu-XXL may not work on ZeBu-Server because of some modifications for message handling in the communication infrastructure.

With ZeBu-Server, the messages from the hardware part of the transactor do no longer include any timestamp when they are transmitted to the software part. As a consequence, the sequential processing which was automatic with the ZeBu Service Loop in ZeBu-XXL must now be implemented in the transactor and the testbench if such a feature is necessary.

## 4.2 New license version for ZeBu-Server

The licenses for the ZeBu tools have been upgraded to version 4.0 for ZeBu-Server, thus licenses which were used on ZeBu-XXL/UF/Personal (license version 3.0) cannot be used on ZeBu-Server.

Section 6.6 of the present Release Note lists the license features that have been created for this software release.



### 4.3 Limitations for Operating System

#### 4.3.1 Specific SUSE Linux Enterprise Server 10 Requirement

The present version of the ZeBu software requires the availability of the `compat-readline-4.3` package which is not installed by default on SUSE Linux Enterprise Server 10 operating system.

#### 4.3.2 Red Hat Enterprise Linux 5.1 and 5.2

The present version of the ZeBu software can be used with Red Hat Enterprise Linux 5.1 and 5.2 with the following limitations:

- Compiling with **zCui** is not possible because of a compatibility issue for the graphical libraries. However a design compiled with Red Hat Enterprise Linux 4 runs correctly on a ZeBu-Server system connected to a PC with Red Hat Enterprise Linux 5.1 or 5.2 operating system.
- The `autofs` Linux functionality, which allows automatic mounting of a remote disk volume, does not work correctly (note that this Red Hat Enterprise Linux 5.1/5.2 limitation is unrelated to ZeBu software).

It is necessary to install the following Red Hat Enterprise Linux 5.1 RPMs:

- o `kernel-2.6.18-53.1.6.el5.x86_64.rpm`
- o `kernel-devel-2.6.18-53.1.6.el5.x86_64.rpm`
- o `kernel-headers-2.6.18-53.1.6.el5.x86_64.rpm`

Some additional recommendations must be taken into account:

- You have to compile any software which is linked to a ZeBu API (binary library) such as a C/C++ testbench or the software part of a transactor using `gcc/g++` version 4.1. This is the default version for this operating system.
- The C++ standard library version 6 (`libstdc++.so.6`) is the default version for this operating system.
- The Linux environment variable `LD_ASSUME_KERNEL` must be unset.

#### 4.3.3 GTKWave Limitation

The package for GTKWave waveform viewer can be installed only with Red Hat Enterprise Linux 4.0 operating systems, but not with Red Hat Enterprise Linux 5.1 or SUSE Linux Enterprise Server 10.

Once installed from a RHEL4 workstation, the program runs correctly on all supported platforms.

### 4.4 zCui Limitations

#### 4.4.1 Project File Compatibility

Once you have saved an existing **zCui** project file (`.zpf`) with Version 6\_2\_0, you can no longer use this file with previous releases. If you intend to use your previous



version project file for non-regression testing, you should save it with a different name before migrating to Version 6\_2\_0.

When opening an existing **zCui** project file, the **Report** window comes to front and shows the problems with the project file, in particular the **zCui** items that have been modified and that will not be kept in the V6\_2\_0 project file.

### 4.4.2 Archive feature is not available in **zCui**

The archive feature is not available in **zCui**. This is due to a recent change in the architecture of the working directory which led to incorrect archive contents.

### 4.4.3 **zCui** uses `/bin/sh` and ignores `.cshrc` aliases/functions

**zCui** uses `/bin/sh` and ignores aliases and functions defined in `.cshrc` and the like. Recommendation: use scripts in your PATH instead of aliases.

### 4.4.4 Limitation for probe declaration with multiple RTL Groups

When the design includes several RTL groups, the hierarchical path to the probes in the Tcl probe command file must not be declared relative to the top of the design. The highest hierarchical level that should appear in a `probe_signals` command is the top of the RTL group to which this probe belongs.

In VHDL, the outputs of modules are not probed because the instrumentation would be too intrusive. Also this version of **zRtlFrontEnd** can only instrument wires and ports of type `std_logic` and `std_logic_vector`; user-defined types (record, arrays, etc.) are not processed.

Note that sometimes the synthesis tool may remove signal aliases so that you might see a signal only under one name.

## 4.5 **zFAST** Limitations

### 4.5.1 Functional Limitations

- Top-Down Synthesis:
  - The generated netlist is flattened or partially flattened and impacts the hierarchical paths. In particular, a mapping file created for an existing block-based synthesized netlist cannot be used for manual clustering with **zCui/zBuild** since the hierarchical name of instances have changed.
  - The runtime access to signals is reduced: only registers can be accessed in the runtime database.
- Memories described in SystemVerilog source files can only have the Read-After-Write priority when processed by **zMem**.

As a workaround, it is possible to force the R/W priority of ALL the ports of



the memory in an Additional **zFAST** Command File. The following command should be used:

```
MemoryStyle:ReadBeforeWrite = (<list_of_memories>)
```

Where <list of memories> lists the memory instances which will have Read-Before-Write priority.

**Example:**

```
MemoryStyle:ReadBeforeWrite = (top.mem, mod.mem1)
```

- When there are several RTL groups in the design, the declaration of a top name for each group is mandatory. If there is only one RTL group, the name of the top of the design will be used for the group.
- The top of the design cannot have a Verilog escaped identifier.
- **zFAST** infers DSPs only for multiplications.

### 4.5.2 Interface Limitations with zCui

When selecting **Verilog95** in the **Main → Verilog Language** list, **zFAST** actually synthesizes the design with Verilog2001 constraints, in particular the Verilog2001 keyword usage in Verilog95 code will cause synthesis errors.

### 4.5.3 Memory Synthesis Limitations with zFAST

Memories inferred as **zMem** memory models by **zFAST** are sometimes not optimized, creating backend compilation issues or decreasing the runtime performance. The following cases have been identified.

#### 4.5.3.1 Bit-enabled and Byte-enabled Memories

In this case **zFAST** may create a **zMem** memory model with a large number of ports that cannot be compiled in the back-end or decrease the runtime performance.

Following is a list of possible workarounds:

- Write manually the **zMem** Tcl script of the memory
- Modify the memory inference threshold in **zCui** (but it impacts all the memories in the design).
- Force this specific memory as registers in the **Advanced zFAST Command File** in **zCui**:

```
Compile:ImplementAsFlops= (module1.mem1 module2.mem2 ...)
```

- An optimization algorithm for bit-enabled memories can be activated by adding the following line in the **Advanced zFAST Command File**:

```
Compile:OptimizeBitEnable = true
```

Note that the command which forces usage of a **zMem** memory model is:

```
Compile:ImplementAsMemory= (module1.mem1 module2.mem2 ...)
```



### 4.5.3.2 Arrays of wires

Even if an array of wires is not an actual memory, it may be synthesized as a fully asynchronous **zMem** memory models. The work around is to force is as registers:

```
Compile:ImplementAsFlops= (module1.mem1 module2.mem2 ...)
```

---

### 4.5.3.3 Latch-based memories

Latch-based memories may be synthesized as **zMem** memory models with asynchronous ports even if their ports are synchronous.

The possible workarounds are the same as for bit-enabled/byte-enabled memories, as described in Section 4.5.3.1.

## 4.6 Back-end Compilation Limitations

### 4.6.1 Limitation for tristate signals

Top-level tristate ports cannot be defined with keeper resolution (other resolutions are correctly supported).

The recommendations for tristate signal handlings are described with details in Section 3.7.3 of the *ZeBu-Server Compilation Manual* (Rev b).

### 4.6.2 Clock connection to the SRAM\_TRACE driver

Connecting a primary clock (zceiClockPort output) as a traced signal in the instantiation of the SRAM\_TRACE driver may cause a routing error during FPGA Place and Route.

### 4.6.3 Deprecated command in zNetgen

Since **zNetgen** is no longer used for probe declaration, the probe command no longer exists in **zNetgen**.

Starting with V6\_1\_1, the declaration of probes has to be done in the **Sources** → **Probe** files item as described in the *ZeBu-Server Compilation Manual*.

### 4.6.4 Limitations for Memory Modeling

#### 4.6.4.1 Limitation on Clocks for DDR2 zrm Memories

Because of an inappropriate frequency range, it is not possible to drive a DDR2 zrm memory instance with a synthesized uncontrolled primary clock generated by the frequency synthesizers (declared by instantiating a zClockPort in the DVE file). The minimum frequency of the synthesizers is much too high to connect such a clock to DDR2 memories.



### 4.6.4.2 Limitation on Read/Write Priority for multi-port, zrm Memories

The read-before-write priority is set individually on one port. The read/write priority between two ports of the same memory instance is not deterministic for zrm memories, even if both ports use the same clock signal.

## 4.7 Runtime Limitations

### 4.7.1 Trace feature

It is not possible to activate the trace feature (SRAM trace) when the xClk frequency is forced at a frequency higher than 300 MHz. The following message is displayed

"Cannot treat "initTrace" request : Trace feature is not functional with your selected xClk frequency"

### 4.7.2 Clock declaration for HDL co-simulation

For HDL co-simulation, the clocks controlled by the HDL testbench must be declared without being attached to a group in the designFeatures file.

Additional design clocks which are not controlled from the HDL testbench can be grouped.

### 4.7.3 Limitation for threadsafe environment

The Z\_REPEAT\_CALLBACK is not available when emulating in a threadsafe environment (when the testbench is linked with libZebuThreadsafe.so library).

### 4.7.4 Fast hardware state limitation on V6\_2\_0

A fast hardware state object must be initialized after board initialization.

### 4.7.5 Limitation for Selection/Deselection of Dynamic Probes

Selection and deselection of a dynamic probe are not supported when a .bin waveform file is opened. Any change in the list of selected dynamic probes causes the following fatal error:

"ZWAVE0050E: "Cannot continue to dump bin file while recomputing dynamic probes".

You can avoid these issues by closing the .bin waveform file before selecting or deselecting any new signal.

Note the following points:

- A selection is implicit if you read the value of a signal for the first time.
- A deselection is implicit if you destroy a signal handler.
- RACC checking or randomization features can also change implicitly the selected dynamic probes.





### 4.7.6 Limitation for Selection of Nets or Ports of the Design at Runtime

When selecting nets or ports in the design with `zDbPostProc` or from the testbench, care must be taken if some ports and nets have the same names but are not interconnected: it is not possible to predict if the net or the port will be selected.

### 4.7.7 Limitation on clock specification for restore

Before V4.3\_3, when the user declared the clock parameters from the C/C++ testbench before or after initialization, they were not taken into account and no message was issued.

Starting with V4.3\_3, an error message is correctly issued when the clock parameters are declared in the C/C++ testbench **after initialization**.

However the same error message is erroneously issued when the clock parameters are declared in the C/C++ testbench **before initialization**, which makes it impossible to declare the parameters when restoring a saved state. As a workaround for restore, EVE recommends that the user declares clock parameters from the `designFeatures` file only (not from the C/C++ testbench).

### 4.7.8 `zInstall` multi-release limitations

The `zInstall` multi-release feature will not work if a job is running on the ZeBu hardware. To run the multi-release feature of `zInstall`, make sure that no job is running on the ZeBu hardware when you run `zInstall`.

## 4.8 Runtime Restrictions to Write Operations

### 4.8.1 Restrictions to Write to Registers

Writing to registers at runtime is supported, except that it is not possible to write to registers in FPGAs connected directly on RLDRAM in 4C and 8C/ICE modules, when RLDRAM is used by the design.

In `zCui`, the **Enable BRAM Read&Write / Write Register / Save&Restore** item in the **Debugging** tab must have been enabled. It is disabled by default.

### 4.8.2 Restrictions to Write to BRAM Memories

Writing to BRAM memories at runtime is supported with the following restrictions.

If neither the **Enable BRAM Read&Write / Write Register / Save&Restore** nor the **BRAM Instrumentation for Read&Write** items in the **Debugging** tab have been set, then at runtime, all output registers of all BRAMs in a FPGA will be reset each time one BRAM is read or written on that FPGA.

Setting the **Enable BRAM Read&Write / Write Register / Save&Restore** item can avoid this behavior, except for FPGAs connected directly on RLDRAM in 4C or 8C/ICE modules, when RLDRAM is used by the design. To avoid the behavior





described above for these FPGAs, set the **BRAM Instrumentation for Read&Write** item.

### 4.8.3 Restrictions to Save & Restore

Save & Restore has the following restrictions:

If neither the **Enable BRAM Read&Write / Write Register / Save&Restore** nor the **BRAM Instrumentation for Read&Write** items in the **Debugging** tab have been set, then all output registers of all BRAMs will be reset each time the design is saved or restored.

Setting the **Enable BRAM Read&Write / Write Register / Save&Restore** item can avoid this behavior, except for FPGAs connected directly on RLDRAM in 4C/8C modules, if the RLDRAM is used by the design. To avoid the behavior described above for these FPGAs, set the **BRAM Instrumentation for Read&Write** item.

## 4.9 Advanced Triggers

Advanced triggers are not supported in the current version.

## 4.10 ZEMI-3 Limitation for Synthesis

The XST synthesizer is not supported for ZEMI-3 transactors.

# 5 Fixed Issues

The list of fixed issues in V6\_2\_0 software (compared to V6\_1\_1 software) is not available in the present document since it was no longer relevant after delivery of cumulative patch B\_00.

Chapter 1 includes detailed information about the issues fixed in B\_00 cumulative patch.



## **6 Version Compatibility**

This chapter includes information which is applicable for Version 6\_2\_0 with cumulative patch B\_00.

### **6.1 Supported PC Configurations**

The ZeBu compilation and runtime tools are 64-bit software which can be used only on a 64-bit PC configuration. The present version of ZeBu has been successfully tested on the following PC configurations:

For compilation:

- Dell Optiplex 760
- Dell PE2950
- Dell PER410

For emulation runtime (PC connected to the ZeBu-Server system):

- Dell Optiplex 760
- HP DL360G6
- HP z400
- Dell Precision T7500

The present version of the ZeBu software only runs on 64-bit PCs; it does not support 32-bit PCs.

### **6.2 Operating System Compatibility**

EVE recommends using Red Hat Enterprise Linux 4.0 operating system, kernel 2.6.

However, this version of the ZeBu software has been tested with the following operating systems for runtime emulation:

- SUSE Linux Enterprise Server 10, kernel 2.6
- Red Hat Enterprise Linux 5.1 and 5.2, kernel 2.6

Refer to Section 4.3, "Limitations for Operating System", for detailed information on how to use these operating systems.

### **6.3 Hardware Compatibility**

This software release is ONLY compatible with ZeBu-Server.

It is NOT compatible with ZeBu-XL, ZeBu-XXL, ZeBu-UF and ZeBu-Personal.



### 6.4 Third-Party Tools Compatibility

The following table gives information about the third-party tools that have been successfully tested by EVE with the present version of the ZeBu software. This list is given for information purposes and does not necessarily imply that other versions will cause system malfunction.

Tool	Tested Versions	Tested OS compatibility *	Remarks
Synplify Pro (Synopsys)	2009-06.SP1	RHEL 4.0	FPGA synthesizer. <a href="http://www.synopsys.com">http://www.synopsys.com</a>
XST (Xilinx)	11.3i	RHEL 4.0	FPGA Synthesizer. <a href="http://www.xilinx.com">http://www.xilinx.com</a>
Precision (Mentor Graphics)	Not Tested		FPGA Synthesizer. <a href="http://www.mentor.com">http://www.mentor.com</a>
VCS (Synopsys)	mx-2008.09	RHEL 4.0	HDL Simulator. <a href="http://www.synopsys.com">http://www.synopsys.com</a>
ModelSim (Mentor Graphics)	6.2f	RHEL 4.0	HDL Simulator. <a href="http://www.mentor.com">http://www.mentor.com</a>
NCSIM (Cadence)	Not Tested		HDL Simulator. <a href="http://www.cadence.com">http://www.cadence.com</a>
SystemC	2.2.0	RHEL 4.0	Required for SystemC co-simulation. <a href="http://www.systemc.org">http://www.systemc.org</a>
GTKWave	3.1.11	RHEL 4.0	Waveform viewer. <a href="http://gtkwave.sourceforge.net/">http://gtkwave.sourceforge.net/</a>
Debussy (Novas)	200707	RHEL 4.0	Interface for design debugging and waveform viewer. <a href="http://www.springsoft.com/">http://www.springsoft.com/</a>
RTLVision GateVision PRO (Concept Engineering)	4.6.5	RHEL 4.0	Graphical netlist analyzers. <a href="http://www.concept.de/">http://www.concept.de/</a>
gcc	3.4	RHEL 4.0	C Compiler. Part of the Linux operating system distribution. Required for C/C++ and SystemC co-simulation. May be required during installation process in case of kernel compilation. <a href="http://gcc.gnu.org/">http://gcc.gnu.org/</a>
	4.1	RHEL 5	For runtime on a RHEL 5 operated PC, the testbench and the software part of the transactor should be compiled with this version of gcc C/C++ compiler.
ISE Place&Route (Xilinx)	11.3i	RHEL 4.0	Xilinx tools for FPGA Place&Route during ZeBu compilation. <a href="http://www.xilinx.com">http://www.xilinx.com</a> Included in ZeBu software package.

\* The Operating System Compatibility of these tools can change and you are recommended to visit the third-party websites for information.



### 6.5 EVE Vertical Solutions Compatibility

#### 6.5.1 ZeBu Transactors

The 6\_2\_0 release is compliant only with 64-bit EVE transactors and utilities.

These packages/utilities can be requested from your usual EVE representative. The following table shows the latest releases of packages/utilities in 64-bit versions.

Transactor Package	First Version for 64 bits	Doc Id
Virtual Display (LCD)	1.3	VSXTOR001
UART	2.0	VSXTOR002
JTAG transactor with TAP controller API	1.2	VSXTOR003
SRAM SW	1.1	VSXTOR004
PCI Express	4.1.2	VSXTOR005
Ethernet (GMII)	1.3.1	VSXTOR006
<b>etherPlug</b> utility	1.1	VSXTOR006
I2S	1.3	VSXTOR007
Digital Video	3.2	VSXTOR008
IEEE 1394 FireWire	1.2.1	VSXTOR009
MMC Device	1.2	VSXTOR010
AHB Master	1.5	VSXTOR011
Streaming Toolkit	1.6	VSXTOR012
USB	1.1.1	VSXTOR013
AXI Master	2.2	VSXTOR014
AXI Slave	2.2	VSXTOR014
I2C	1.0	VSXTOR015
MMC Host	1.1	VSXTOR016
HDMI Sink	1.0	VSXTOR017

#### 6.5.2 ZeBu Memory IPs

ZeBu Memory IPs with specific licenses are all supported by V6\_2\_0 software.

### 6.6 ZeBu Licenses

License features are the same as for ZeBu version 6\_1\_1.

The following new features require some specific license:

- zDPI feature
- Offline Debugger

Some specific licenses are required for Concept Engineering netlist analyzers and for Xilinx ISE 11 software.

You should contact your usual EVE representative for detailed information.



## 7 EVE Contacts

For product support, contact: [support@eve-team.com](mailto: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 5F-2, No. 229, Fuxing 2nd Rd. Zhubei City, Hsinchu County 30271, TAIWAN (R.O.C.) Tel: +886-(3)-657-7626