# ZeBu® Gate-Level Support Application Note

Version V-2024.03-1, July 2024

SYNOPSYS®

# 1

# ZeBu® Gate-Level Support Application Note



Compiling a Gate-Level design on ZeBu emulator always consumes a lot of resources and time. Gate-Level design is typically characterized by a large number of modules having a lot of library cells and deep modules hierarchy.

The main challenges are high compile time and high memory usage.

This application note captures some recommendations to address the challenges. See the following topics:

- Design Compilation Recommendations
- UTF Settings Recommendations
- Synthesizer Settings Recommendations

## Design Compilation Recommendations

This section describes the following:

### Check for DFT Logic

Usually Gate-Level on emulation is done without DFT. Otherwise, higher resource utilization and very poor timing performance are expected.

In such a scenario, tie-offs are required to optimize the DFT logic. This translates into a list of force statements that must be added to the UTF file.

### Check for Memory Blocks

It is recommended to use emulation-friendly memory RTL source instead of Gate-Level representation of memory because memories are bit-blasted. Therefore, there is no option of back-door write or read.

Some embedded memories have a dual-edge. Earlier, ZeBu silently ignored the usage of dual-edged logic and internally created a multi-driven logic. The multi-driven logic was optimized and led to functional failure.

With the recent release, ZeBu displays an error for dual-edge memories. So, to allow dual-edge clock synthesis, add the following commands `synthesis -advanced_command {Compile:HandleDualPhaseClock=true}`.

## Design Compilation

Full analysis and elaboration compilation for every build unnecessarily replicates computation effort and wall clock time.

It is recommended to review `+define` present in the `vcs` script. It helps to avoid bad branch selection, create libraries independently, and then compile your designs for the hardware using these precompiled libraries.

To review `+define`, perform the following steps:

1. Map the logical libraries to appropriate files in the `synopsys_sim.setup` file as follows:

   ```
   lib1 : <path_to_lib>/lib1

   lib2 : <path_to_lib>/lib2
   ```

2. In your `vcs` script, add the following commands:

   a. Create the work directories for each library using `mkdir -p lib1 lib2`.

   b. Add the analysis commands (`vhdlan/vlogan`) for the source files as follows:

   ```
   vhdlan -full64 -w lib1 <path_to_source_file>.src1.vhd

   vlogan - full64 -w <path_to_source_file>.src2.v
   ```

3. At **zCui** compilation, provide a compilation script (or a command line) to the UTF command, `vcs_exec_command`, which contains only elaboration command (`vcs`) for building the design hierarchy from the library files generated during the analysis stage.

   ```
   vcs -full64 [elab_opts] [libname.]top_unit
   ```

When there are no dependencies across commands, multiple analyze commands can be invoked to reduce the compile time.

## UTF Settings Recommendations

To improve compile time and memory footprint, the following UTF settings are recommended:

- Ensure that **zFmCheck** is not enabled. To disable **zFmCheck**, remove `synthesis -generate_db_for_fmcheck true` or set `synthesis -generate_db_for_fmcheck false`.
- Target a machine with a large amount of free memory.
- Set an optimized number of the synthesizer threads using following optimization UTF command: `-number_of_threads 2/4/8/16 (worst case N=1)`
- Enable the inlining feature if you are using a ZeBu release earlier than S-2021.09-1. In ZeBu S-2021.09-1, the inlining feature is enabled by default with a limit set to 50. This limit can be increased if a higher value is needed using the following UTF command: `optimization -auto_inline_limit <limit>`
- Enable the Virtual File System to avoid large number of files using the following UTF command: `synthesis -use_vfs true`

## Synthesizer Settings Recommendations

It is recommended to enable the following ZeBu synthesizer settings:

- `synthesis -wls_option {-netlist}`: Lets the synthesizer to skip some of the steps that are not relevant for gate-level modules.
- `synthesis -wls_option {-enhancedBlackboxSupport}`: Improves the synthesizer circuit linking strategy for the entire design. It is recommended to add this optimization if a ZeBu release earlier than S-2021.09 is used (enabled by default starting 2021.09).
- `synthesis -wls_option {-clearVcsData=0}`: Frees up the VIR memory when it is not required.
- `setenv VCS_SIMON_PERF 0`: Disables synthesizer profiling.

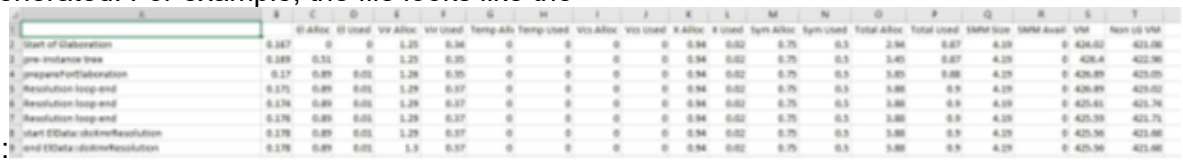
When the initial compilation is available, you can understand the design by reviewing the `SM_PME` messages present in the zcui.work/zCui/log/vcs_splitter log file. It indicates the total number of elaborated modules processed.

You can also check the main steps, which consumes more time by reviewing the zcui.work/design/synth_Default_RTL_Group/vcs_report.log file.

For more details on subsequent stages, enable advanced profiling options:

- **VCS profiler:** Add `-Xbymod=0x400` in the `vcs` command line for details about subsequent stages.

  A file named radvcm/vcs.<pid>/report.dat with detailed data is generated. For example, the file looks like the following:



- **Synthesizer profiler:**
  - Add `synthesis -wls_option{-fullStats}`.
  - Get `<nb_thread>`, which corresponds to the number of child directories in <zcui.work>/vcs_splitter/simon.out.
- Run `$VCS_HOME/bin/simonstatscollector.py <zcui.work>/vcs_splitter/simon.out <nb_thread>`

  It generates simon_stats.txt, which illustrates the longest SIMON processing times from top to bottom.

You can share all the profiling data with experts to do more design-specific analysis and some specific optimizations that can be suggested, if applicable.

# Copyright and Proprietary Information Notice



## Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

## Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

## Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

## Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

## Third-Party Links

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

www.synopsys.com