

NEC**CyberWorkBench®****Pioneering C-based LSI Design**

**Integrated Development
Environment CWB-gui
User Manual
(Version 5.4.1)**

NEC Confidential

Copyright © 1988-2016 NEC Corporation. All rights reserved.

- The information obtained from this manual shall not be disclosed outside the scope of agreement.
- This document shall not be reproduced, altered, reprinted, distributed and disclosed without prior permission from NEC in writing.
- NEC shall not guarantee or grant license for the Intellectual Property or other rights belonging to it or any third party (company) for the usage of products or information mentioned in this manual. NEC shall not own any responsibility in event of any problem arising owing to violation of rights of any third party (company) caused by the above-mentioned usage.
- This manual can be modified without any prior notice. Hence confirmation of latest version should be performed while using the same.
- This product may fall under restricted items according to regulation of Foreign Exchange and Foreign Trade Control Law, therefore caution is required.
- CyberWorkBench is a registered trademark of NEC Corporation. Any other product name, company name are the registered trademarks or brands of their respective company.

SCAD-CY-1300029

Table of Contents

Introduction	5
1. Overview of CWB-gui	6
2. CWB-gui Startup method.....	9
2.1. Windows.....	9
2.2. Linux.....	9
2.3. Post startup settings.....	10
3. Appearance of CWB-gui.....	11
3.1. Drop Down Menu.....	12
3.2. Tool Bar.....	13
3.2.1. General tool bar.....	13
3.2.2. Operation tool bar.....	14
3.3. Project management window.....	15
3.4. Command palette.....	15
3.5. Work screen.....	15
3.6. Console window.....	16
4. Project Management	17
4.1. Details of each component.....	19
4.1.1. Workspace.....	19
4.1.2. Intermediate project.....	20
4.1.3. RTL project.....	22
4.1.4. Source files, include files.....	23
4.1.5. Test bench.....	24
4.1.6. Process.....	24
4.2. Support for actual file system.....	25
4.3. Project editing command.....	27
4.3.1. Commands related to workspace.....	27
4.3.2. Intermediate project, RTL project related command....	29
4.3.3. File related commands.....	32
4.3.4. Import directory.....	34
4.4. Policy and limitations of project management.....	36
4.4.1. Policy related to project structure.....	36
4.4.2. Behavioral Policy.....	38
4.4.3. Constraints.....	39
4.5. Restoration processing at time of abnormal exit.....	43
4.6. Reading of workspace of different versions.....	44
4.7. Example of project structure.....	45
5. Properties setting of the project.....	47
5.1. Clock.....	47
5.2. Device.....	47
5.3. Basic Library.....	47

5.4.	Functional unit library.....	48
5.5.	Changes to the sub module.....	48
6.	Policy of Option settings screen.....	49
7.	Source file analysis.....	52
7.1.	BDL analysis.....	52
7.2.	SystemC analysis.....	55
7.3.	RTL analysis.....	56
7.4.	Call tree, Instance tree.....	58
7.5.	Overflow check.....	58
8.	Constraint file generation	60
8.1.	Use of standard BLIB and standard FLIB.....	60
8.1.1.	Specifications of project creation.....	61
8.1.2.	Change of standard BLIB and standard FLIB value	61
8.2.	Automatic generation of Basic library file (BLIB).....	63
8.3.	Automatic generation of functional unit related constraint file (FLIB/FCNT)	64
8.3.1.	Automatic generation of Functional Unit Library file (FLIB)/Functional Unit Count file (FCNT)	64
8.3.2.	Linking of standard FLIB settings and FLIB automatic generation.....	65
8.4.	Delay area setting of Functional unit library file (FLIB).....	66
8.5.	Automatic generation of Memory library/constraint file (MLIB/MCNT)	67
8.6.	Editing of constraint file.....	68
8.7.	Constraint file icon display.....	69
9.	Behavioral synthesis	71
9.1.	Option settings of bdltran.....	71
9.2.	Execution of bdltran.....	72
9.3.	Handling of _E.FCNT file generated by synthesis.....	73
10.	RTL generation	75
10.1.	Verilog generation.....	75
10.2.	VHDL generation.....	76
10.3.	Generation of logical synthesis script.....	77
10.4.	Generated Verilog/VHDL export.....	77
11.	Supporting functionality for Synthesis	79
11.1.	Synthesis support window overview.....	79
11.2.	Synthesis execution.....	80
11.3.	Graph.....	80
11.3.1.	Plot (trade-off graph)	80
11.3.2.	Bar Graph	84
11.3.3.	Loop latency display	87
11.4.	Error table.....	88

11.5.	Circuit quality display.....	88
11.6.	Result Display.....	89
11.6.1.	Hierarchical history tab.....	89
11.6.2.	Process name tab.....	90
11.7.	Behavior of Save history and Restore results.....	91
11.7.1.	Save history.....	91
11.7.2.	Delete history.....	92
11.7.3.	Restore results.....	92
12.	Functionality to analyze synthesis results.....	94
12.1.	Display of linkage of source file and RTL etc.....	94
12.1.1.	Display of linkage of source file and Verilog.....	94
12.1.2.	Display of linkage of source file and VHDL.....	95
12.1.3.	Display of linkage of source file and cycle level BDL.....	95
12.1.4.	Display of linkage of source file and signal state and its handling table.....	96
12.1.5.	Notes for display of linkages.....	96
12.2.	State transition display in editor.....	97
12.3.	Signal state handling.....	98
12.3.1.	Port tab reference and operation method.....	100
12.3.2.	Reference and operation method of Register tab....	101
12.3.3.	Reference and operation method of Memory signal...	102
12.3.4.	Reference and operation method of functional unit tab.....	103
12.3.5.	Reference and operation method for Others signal ..	104
12.3.6.	Display of Pipeline state.....	105
12.4.	Linkage with delay report.....	106
12.4.1.	Basic functionality.....	106
12.4.2.	False path display.....	107
12.5.	Linkage with other analysis tools.....	108
12.5.1.	Linkage with Data path viewer.....	108
12.5.2.	BDL display by BDL viewer.....	109
12.5.3.	Circuit quality display.....	110
13.	Bus interface automatic generation	112
13.1.	cybus.....	112
13.2.	CybusM.....	112
13.3.	Usage method of bdef editor.....	114
14.	Top Module Generation	117
14.1.	Option-Settings and execution of topmodgen.....	117
14.2.	Structural Level topmodgen.....	119
14.3.	Behavioral level topmodgen.....	119
15.	Operation related to LOG file	121
15.1.	Import of LOG file and its expansion.....	121

15.2.	LOG file export	122
16.	Others	123
16.1.	Tool options	123
16.1.1.	General Tab	123
16.1.2.	Keyboard tab	124
16.1.3.	Editor tab	124
16.1.4.	Menu tab	125
16.1.5.	Signal table tab	125
16.1.6.	Execution control tab	125
16.1.7.	Tool path tab	126
16.2.	Attributes Palette	127
16.3.	Error filter	128
17.	Command list	130
17.1.	Drop down menu	130
17.1.1.	File	130
17.1.2.	Edit	130
17.1.3.	Display	131
17.1.4.	Architecture design	131
17.1.5.	Synthesis	131
17.1.6.	Tools	132
17.1.7.	Window	132
17.1.8.	Help	133
17.2.	Toolbar	133
17.2.1.	General Toolbar	133
17.2.2.	Operations toolbar	134
17.3.	Project Management context menu	134
17.3.1.	Workspace	134
17.3.2.	Intermediate project - RTL project	135
17.3.3.	src, include, testbench	136
17.3.4.	Process	137
17.3.5.	Simulation scenario	139
17.3.6.	File	140
18.	Index	142

Introduction

This section describes the usage method of Cyber Integrated Development Environment (CWB-gui).

Hardware development through Cyber uses behavioral description language BDL (which is an extended form of C language, used for hardware description) and SystemC. CWB-gui provides various analysis functionalities and user support functionalities. Various kinds of information that cannot be obtained by implementing the behavioral synthesis in command line, can be obtained by executing the behavioral synthesis in BDL/SystemC in CWB-gui.

For details of each tool operated from CWB-gui, please refer to their respective manuals.

In this document, CWB and CWB-gui are distinguished as follows:

- CWB: Abbreviation of Cyber Work Bench
- ASIC-GUI, ASIC: Integrated Development Environment for ASIC license
- FPGA-GUI, FPGA: Integrated Development Environment for FPGA license
- CWB-gui: Integrated Development Environment

1. Overview of CWB-gui

CWB-gui is an integrated development environment, used at the time of hardware development by using Behavioral Synthesis System (CWB). Figure 1-1 shows the basic design flow of CWB.

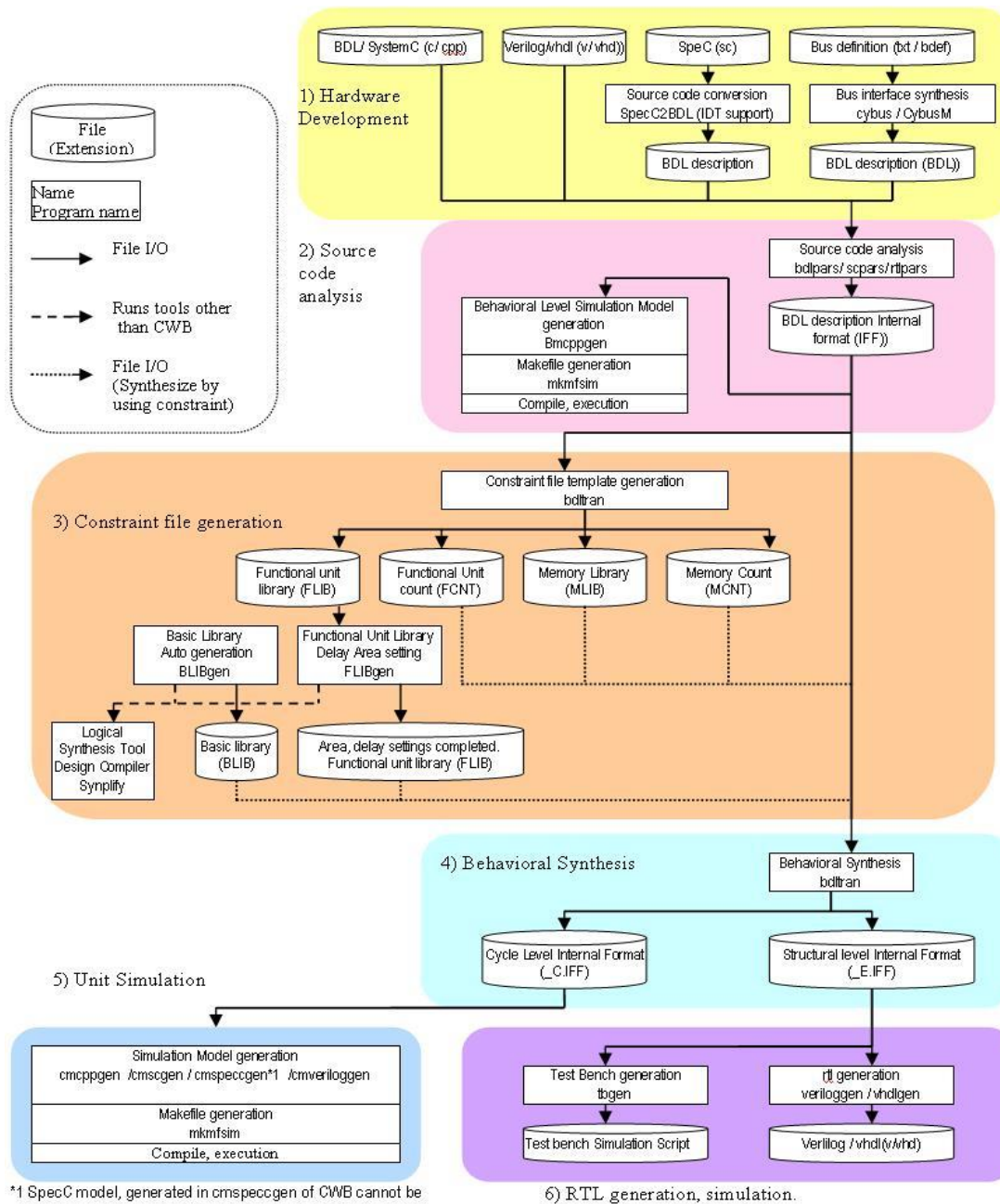


Figure 1-1 CWB Design Flow

Design Flow Overview

The overview of the design flow is as follows.

- 1) **Hardware Development:** Hardware is described at behavioral level by using behavioral description language (hereafter will referred as BDL), SystemC. Since both Verilog and VHDL can be used for input in CWB, so already existing designs can also be utilized. Further, bus interface BDL can be generated automatically from text file having bus specifications.
- 2) **Source file analysis:** Internal format files used as input for behavioral synthesis are generated by analyzing BDL, SystemC, Verilog, and VHDL files. In case of BDL input, description contents are verified quickly by generating C++ pin accuracy behavioral level simulation model and compiling it.
- 3) **Constraint file generation:** Constraint file, used during behavioral synthesis, is generated. Some constraint files contain specifications of functional units used during synthesis and memory, register etc, and some specify the maximum number of circuits that can be used. Constraint files are not always necessarily required depending upon the hardware description. Sometimes they are necessary to provide specifications such as area and delay for functional units and registers etc.
- 4) **Behavioral Synthesis:** Behavioral synthesis is done using internal format file generated in point 2) and if required, each type of constraint files generated in point 3).
- 5) **Model Simulation:** Compilation is executed by automatically generating SystemC/Verilog simulation model from behavioral synthesis result file.
- 6) **Rtl Generation:** Verilog/VHDL file is generated from behavioral synthesis result file. Further, since test bench and script file for RTL simulation can be auto generated hence simulation is done easily.

Hardware Development Steps

General steps for hardware development using CWB-gui are as given in Figure 1-2 below.

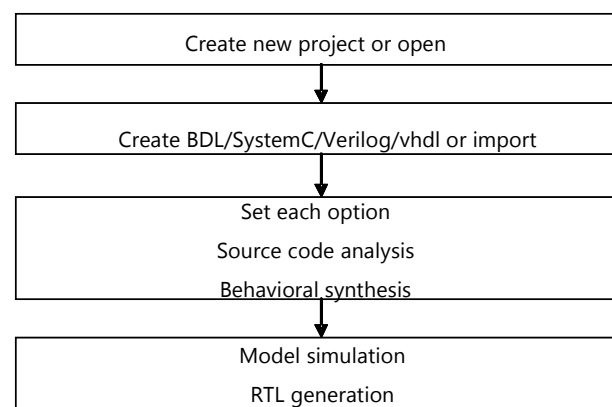


Figure 1-2 General Hardware development flow using CWB-gui

In CWB-gui, I/O file, intermediate file, each option etc are managed as a part of a project. Therefore, it is necessary to create the project in the beginning. Once the project is created, BDL/SystemC/Verilog/VHDL source files are entered (sourced) in project. Then, by setting the options for each program, source file analysis, behavioral synthesis, model simulation and RTL generation is performed.

This document gives explanation regarding all the above mentioned functionalities.

2. CWB-gui Startup method

This section explains about startup method and options for CWB-gui.

2.1. Windows

Startup method for Windows version is provided below.

- Double click on the executable file "cwb.exe" (cwb-fpga.exe for FPGA) under CWB installation directory.
- Double click on the shortcut of "cwb.exe" (cwb-fpga.exe for FPGA) which is created on desktop during installation of CWB.

2.2. Linux

It is run by applying \${CYBER_PATH}/bin in PATH variable as

```
> cwb [options] [workspacefile(.cyt)]  
(Binary name of FPGA is "cwb-fpga")
```

Starting with \${CYBER_PATH}, for details of environment variable settings, please refer to INSTALL file which is bundled with CWB package.

Options are mentioned below.

- -v, -h : display CWB-gui version number
- -nosplash : do not display image while CWB-gui startup
- -EE / -nosplash : startup as English version
- -EJ : startup as Japanese version
- -trace : Display message in standard error output
- -log : Create executable log file directly under start directory and save message.
- -force_read : Enables reading of newer version of workspace file, project file that what is supported by CWB-gui. However, in this case behavior of project file which is read is not guaranteed in CWB-gui. (Section 4.6)

Project managed by CWB-gui is termed as "Work Space". Specified work space opens automatically on CWB-gui startup, if Workspace file (*.cyt) is provided as a parameter.

On CWB-gui startup, following files are created under home directory.

~/qt/cwbrc

This file has CWB-gui information like window structure, user settings etc. saved in it. In Windows version, same information is saved in registry. The behavior of this file is not guaranteed if it is edited by using any editor.

And, according to the limitations of Qt used as framework in CWB-gui, the behavior may become unstable if it is done partly through x server. Especially, multi window mode of astec-X is deprecated.

2.3. Post startup settings

Below mentioned two points can be setup as per requirement. In case functionalities such as editing source file using external editor or browsing manual are not implemented in CWB-gui, then its settings is not required.

External text editor settings

Text file displayed in GUI can be opened using external editor. It requires setting up the location of binary file of external editor to be used (Section 16.1.3). Setup procedure is as following.

- 1) Select 'Options' from 'Tools' drop down menu.
- 2) Select 'Editor' tab.
- 3) Specify binary file in 'Text File Editor'.

Pdf viewer settings

Each type of manual bundled in CWB tool kit can be viewed using GUI. It requires setting up the location of binary file of pdf viewer to be used (Section 16.1.7). Setup procedure is as following.

- 1) Select 'Options' from 'Tools' drop down menu.
- 2) Select 'Tool path' tab.
- 3) Specify binary file in 'pdf file viewer'.

3. Appearance of CWB-gui

When project is opened by running CWB-gui, a screen will open as shown in Figure 3-1(Appearance of CWB-gui 4.2).

CWB-gui consists of drop down menu, tool bar, project management window, browser window, work screen, console window and command palette. This section explains about each of these sections. Further, browser window has been removed in CWB 5.4.

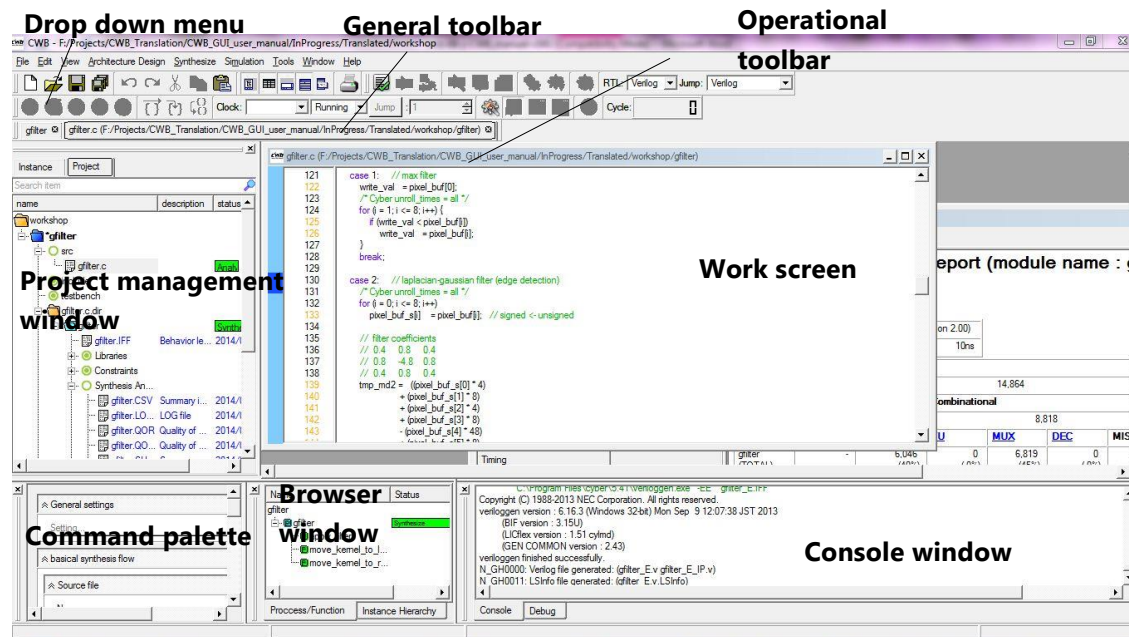


Figure 3-1 Appearance of CWB-gui 4.3

3.1. Drop Down Menu

There are 10 types of menus under Drop down menu as shown in Figure 3-2.

File Edit View Architecture Design Behavior Level Synthesize Property Check Equivalence Check Simulation Tools Window Help

Figure 3-2 Drop Down Menu

Brief description of each menu is explained in Table 3-1. Please refer to section 4 for file, section 13 for automatic generation of bus interface, section 14 for top module generation, section 5, 8, 9, and 10 for synthesis and behavioral level simulation, and section 16 for tools. Please refer Appendix 17.1 for details of commands of each menu.

Item Name	Description
File	Project management operations such as create new project, open, save, close, create new file, open, and save, and close
Edit	File editing operations such as modify, copy, paste, undo, redo etc
View	view and hide each window
Architecture - Design	Automatic generation of bus interface and top module generation
Synthesis	Source code analysis, behavioral synthesis, RTL generation and option setting for these programs
Property Check	Property Check It cannot be selected if property check is not installed
Simulation	simulation model generation, makefile generation etc unit simulation related and option setting
Tools	Various tool option setting
Window	Window display setting in work screen
Help	display of help, and version information

Table 3-1 Items of Drop Down Menu

3.2. Tool Bar

There are two types of tool bars available in CWB-gui.

1. General tool bar
2. Operation tool bar

3.2.1. General tool bar

General tool bar (Figure 3-3) consists of project management related icons, file editing related icons window view/hide setting icons, and file print icons.

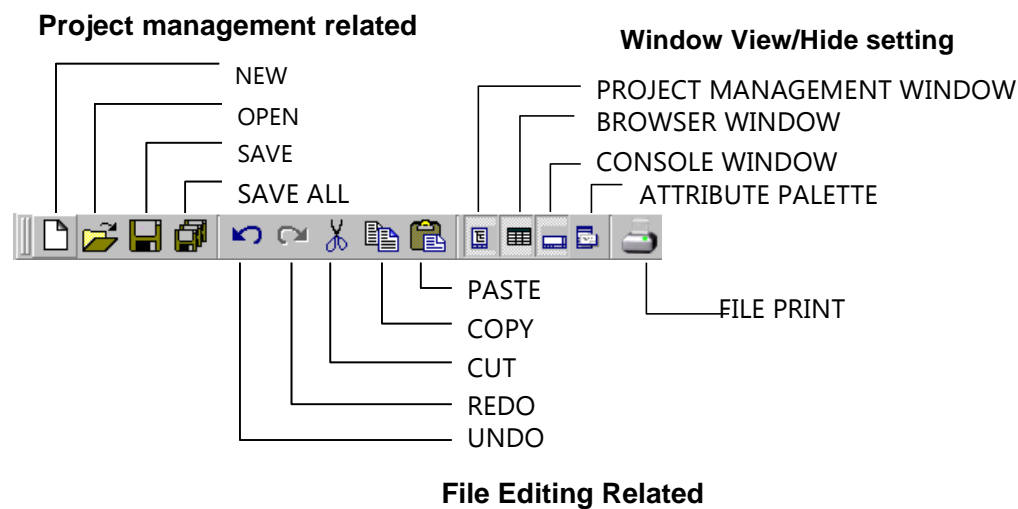


Figure 3-3 General tool bar

Description of each icon of general tool bar is shown in Table 3-2. Please refer to section 4 for project management related commands.

Item Name	Description
New	Create new project or file
Open	Open project or file
Save	Save the specified file
Save all	Save all open projects and files
Undo	Undo the edited portion
Redo	Redo the undone portion
Cut	Cut the specified character string
Copy	Copy the specified character string on clip board
Paste	Paste the character string of clip board
Project management window View/Hide	View / Hide the project management window
Browser Window View/Hide	View / Hide the browser window
Console Window View/Hide	View/Hide the console window
Attribute Palette View/Hide	View/Hide the attribute palette
File Print	Print the text file opened in the gui editor

Table 3-2 General tool bar description

3.2.2. Operation tool bar

Operations toolbar (Figure 3-4) consists of analysis, synthesis, RTL generation, model simulation, language selection related icons, and stop icon.

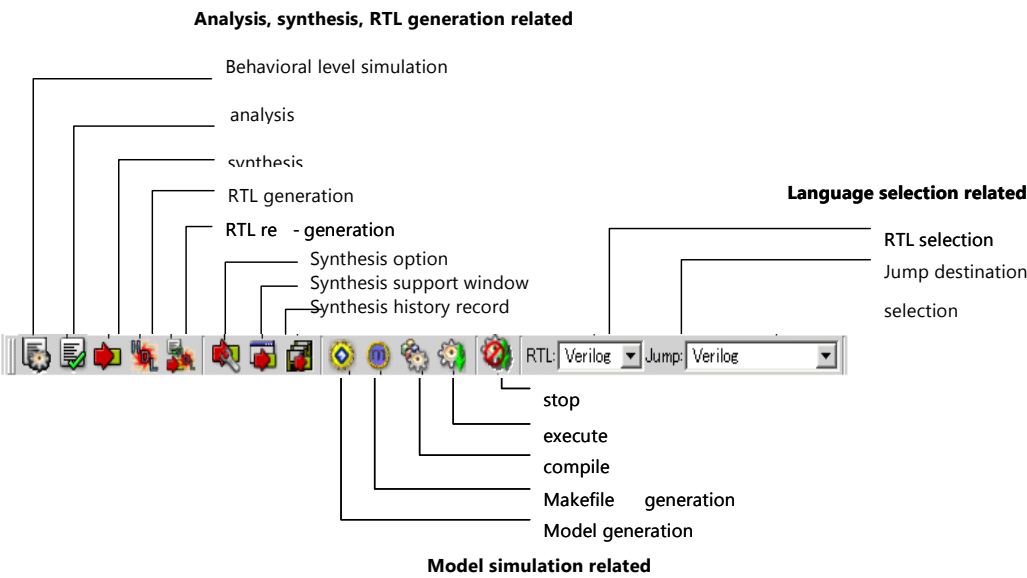


Figure 3-4 Operation toolbar

Description of each icon of operation tool bar is given in Table 3-3. Please refer to section 7, section 9 for synthesis and synthesis option, section 109 for rtl generation, section 0 for synthesis support window.

Analysis	Implement source code analysis for specified source code
Synthesis	Implement behavioral synthesis for specified process
Generate RTL	Implement RTL generation for specified process
Regenerate RTL	Forcibly execute analysis, behavioral synthesis, RTL generation for specified process
Synthesis option	Start the synthesis option setting screen
Synthesis support window	Start the synthesis support window
Save synthesis history	Manually save synthesis results as history
Generate Model	Implement the model generation for specified process
Generate Makefile	Implement the Makefile generation for specified process
Compile	Carryout compile for specified process
Execute	Implement unit simulation for specified process
Stop	Stop the running program
Select RTL	Select RTL output from either verilog or vhdl By default, it is verilog
Select jump destination	Select the jump destination when clicking the source code displayed in CWB-gui editor.

Table 3-3 Operation toolbar explanation

3.3. Project management window

Project management window shows the files and folders in a project, opened in CWB-gui. Please refer to section 4 regarding project management window. Right click on each icon in Project Management Window to display menu for executable command corresponding to that icon. Refer to Appendix 17.3 for details of each command within menu.

3.4. Command palette

You can execute the basic commands in CWB-GUI itself in command palette. Executable command of respective icon can be entered by selecting the icon on project management window.

3.5. Work screen

Work screen (Figure 3-5) shows the source file, signal and corresponding chart displaying the state after synthesis. When source file, include file etc of project management window are double clicked, then work screen shows the file in CWB-gui internal editor. If “Signal Table” is selected in context menu for processes provided in project management window, then chart corresponding to signal and state is displayed in work screen. Please refer to section 12 for details regarding CWB-gui internal editor and Signal Table.

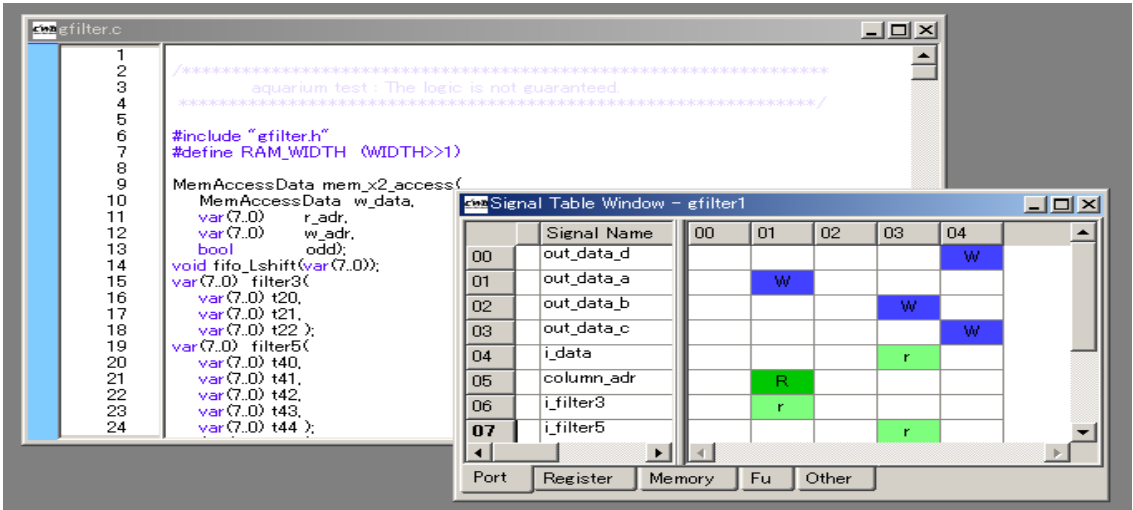


Figure 3-5 Work screen

You can display multiple files stored in process directory by clicking on “Process wise display” from “Window” drop down. Figure 3- shows display of CSV, error file, circuit

performance. The tab can switch display of each file.

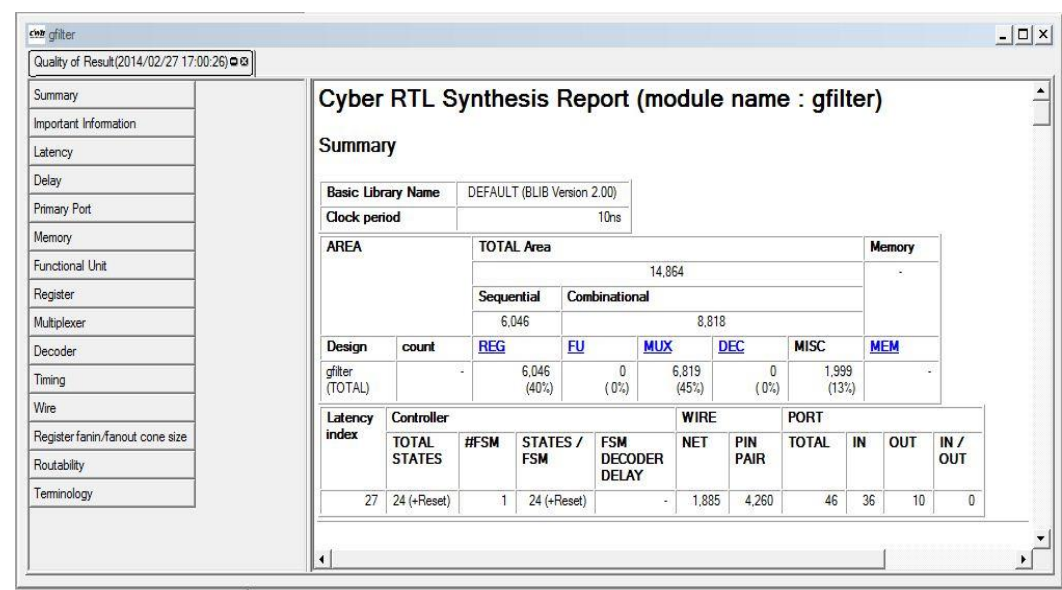


Figure 3-6: Process wise display

3.6. Console window

Console window shows the execution log of each program executed in CWB-gui. Further, it is possible to save or delete the log.

4. Project Management

This section explains about project management. Project management is mainly done through project management window. There are two types of views of project management window, you can switch the view by tab. They are named as "Instance" tab and "Project" tab. In "Instance" tab, hierarchy structure of process instances is displaying, this is called as instance view. Figure 4-1 shows example of instance view. By default, the project management window shows instance view.

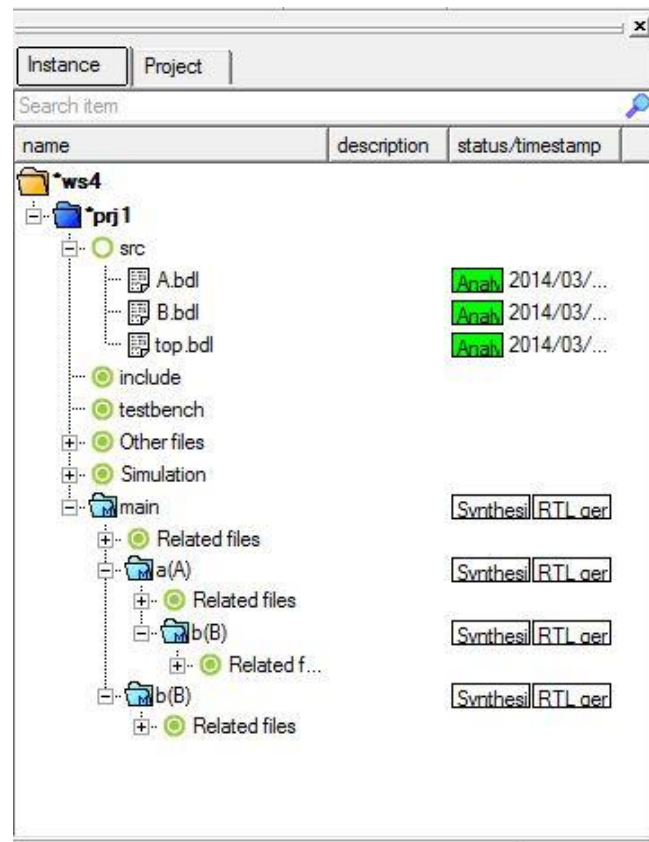


Figure 4-1 Instance view of project management window

In process view also, process instances are displayed in hierarchy. However, the difference is that, synthesis option etc. is saved for each process and not for each instance. For instance, if the synthesis option for instance b1 of process B changes, synthesis option for other instances of process B will also change. Hence, take a note of it.

The directory hierarchy displaying below to the work sheet under "Project" tab is the "Project view". Example of the same is given in Figure 4-2. Example is same as the instance view however, it is the view if displayed from directory hierarchy. In ASCII version, project view was the default view of project management window of CWB 5.4 and older. Hereafter, it explains project management method focusing on project view.

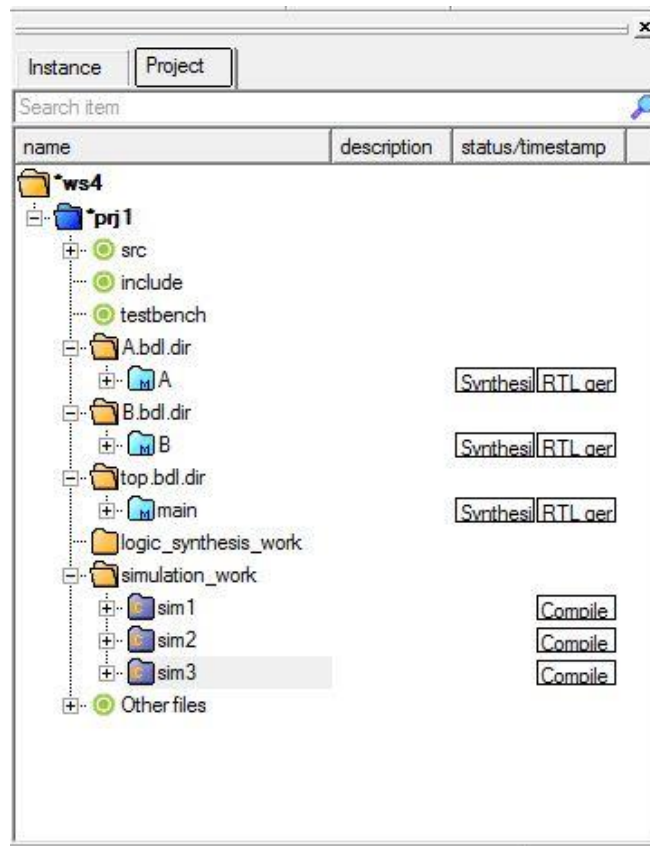




Figure 4-2: Project view of project management window








Initially when CWB-gui is started, the project management window is blank. In this state, if new workspace is created or existing workspace is loaded, workspace configuration of project management window is displayed.

The extension of workspace file of CWB-gui is .cyt. CWB project allows hierarchical structures. The entire sub-tree of file system, which considers directory of project file as root directory, is equivalent to single CWB-gui workspace.

Types of icons

The project shown in project management window is displayed through a tree configured by various icons. The configuration of directory in file system and tree displayed in project management window are almost having same configuration. The icons shown in project management window are as follows.

- 1)  Workspace (refer to section 4.1.1. It is equivalent to the root of CWB-gui project. In the actual file system a directory corresponding to same name exists.
- 2)  Intermediate project (refer to section 4.1.2): It is intermediate hierarchy node of CWB-gui project. It is a location to store source files of BDL and SystemC and it is a base point for executing various operations for stored source files. In the actual file system a directory corresponding to same name exists.

- 3)  RTL project (refer to section 4.1.3): Same as point 2), it is an intermediate hierarchy node of CWB-gui project. It is a location for storing Verilog, VHDL source files.
- 4)  Process folder (refer to section 4.1.6): It is a folder to indicate process. A process is a single synthesis unit and on it various operations such as synthesis, rtl generation etc are performed in this process folder. In the actual file system a directory corresponding to same name exists.
- 5)  Folder: It indicates the intermediate nodes other than project and process folder. In the actual file system a directory corresponding to same name exists.
- 6)  Virtual folder: It indicates the intermediate nodes other than project and process folder. In the actual file system a directory corresponding to same name does not exist.
- 7)  Simulation scenario folder : It is an executable directory to carry out Model simulation.
- 8)  Constraint file: It is an icon to indicate constraint file.
- 9)  General file: It is an icon to indicate files other than constraint files.

Further, the detailed information of constraint file like status of its registration as synthesis option etc are expressed by icon. The details are explained in section 8.

Next section onwards, explanation is provided regarding details of each component (section 4.1), handling of project management window and actual file system (section 4.2), and series of commands (section 4.3) for carrying out project management.

4.1. Details of each component

This section explains the details of each component of project management window.

4.1.1. Workspace

Project root of CWB-gui is referred as workspace. As shown in Figure 4-1, it is equivalent to icon (gfilter) located at the top.

Workspace is equivalent to a single directory in actual file system. If workspace is created in CWB-gui, corresponding new directory is generated in file system. File path of actual file system of this directory is mentioned in CWB-gui application label (Figure 4-3) .



Figure 4-3 CWB application label

Project file (.cyt) is generated under this directory. Information regarding projects created under workspace is stored in Cyt file. Operation is not secured when this file is directly edited by editor etc.

Open and close of CWB-gui workspace are used for this project file. Section 4.3.1 gives detailed explanation of operations concerning workspace.

In case of ASIC, Intermediate project, RTL project can be generated under workspace; however, files such as source file etc. cannot be kept. In case of FPGA, only one intermediate project is auto generated under workspace.

4.1.2. Intermediate project

Project nodes other than RTL project ((Section 4.1.3)) are referred as Intermediate projects. In **Figure 4-1 Instance view of project management window** prj1 folder is the intermediate project.

Workspace plays the role of only project root of CWB. On the contrary, design data such as source files and include files can be kept in intermediate project.

Characteristic of intermediate project prj is given below.

- In intermediate project, there is src folder that keeps source files of BDL, SystemC, include folder that keeps include files, as well as testbench folder that keeps testbench files. Details of src and include are explained in section 4.1.4. Details for testbench are explained in section 4.1.4. Virtual folder icon is used as shown in Figure 4-2, because there is no directory corresponding to src, include and testbench in actual file system. When a new source file, include file, testbench file is created or imported by copy format, these files exist under directory corresponding to intermediate project (section 4.2). In case files are imported by reference format of relative path or full path, the files do not exist under project and whenever corresponding files in CWB-gui are accessed, imported files are sequentially referred (section 4.3.3).
- Simulation_work folder exists in intermediate project. Directory with same name

exists under directory corresponding to the project in actual file system. Work directory is created under this directory to execute simulation that is called simulation scenario, and various simulations are executed here.

- In intermediate project, there is "Other files" folder that keeps files other than source files and include files. Virtual folder icon as shown in Figure 4-2 is used because directory corresponding to the "Other files" folder does not exist in the actual file system. All files shown in "Other files" exist (section 4.2) under the directory corresponding to intermediate project.
- Work folder of each source file exists in intermediate project. In Figure 4-2 there are three source files under src i.e. sub1.c, top1.c, top2.c. At this time, work folder sub1.c.dir, top1.c.dir, top2.c.dir, corresponding to these source files, are generated under prj. Work folders corresponding to source files are generated under directory equivalent to intermediate project even in actual file system (Section 4.2).
- In case of ASIC, it is possible to have another different intermediate project under a certain intermediate project, as sub-hierarchy. Figure 4-2 shows that intermediate project sub is kept under intermediate project prj, as sub project. Even in actual file system, sub-hierarchy intermediate project is generated under directory equivalent to intermediate project (section 4.2) .

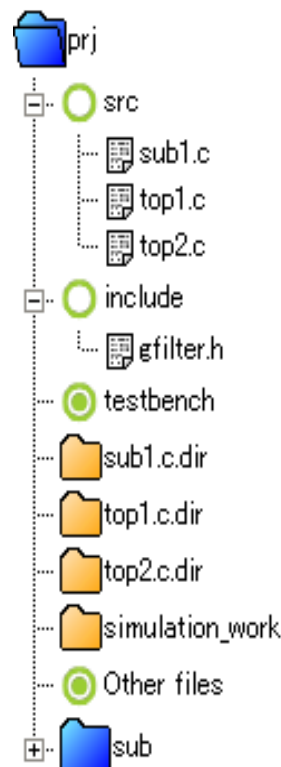


Figure 4-2 Intermediate projects

Management file (.cyp) of applicable intermediate project is generated under directory

corresponding to intermediate project. Information related to project created under applicable project, source file information of applicable project, information of behavioral synthesis target, information of various option settings etc. is stored in cyp file. Cyp file maintains information related to corresponding directory only. In case project has sub projects, then cyp file in the sub project maintains the information of the sub project. The operation is not secured when this file is directly edited by editor etc.

4.1.3. RTL project

RTL project is one of the type of intermediate project. Figure 4-3 shows the example of RTL project. In Figure 4-3 prj folder is RTL folder.

Source file of Verilog, VHDL can be stored in RTL project. Similar to intermediate file, BDL, SystemC source files cannot be stored in it. Further, it cannot keep intermediate project and RTL project, as sub hierarchy. Conceptually, as shown in Figure 4-15 RTL project is used to store existing design data like IP described in RTL.

Taking the example of intermediate project prj as shown in Figure 4-3 , given below is the list of features of RTL project.

- In RTL project, there exists a folder src, which keeps source files of Verilog and VHDL. include folder does not exist. Details of src are explained in section 4.1.4. Virtual folder icon is used as shown in Figure 4-3, because there is no directory corresponding to src folder in actual file system. When new source files and testbench files are created or imported by copy format, these files exist under directory corresponding to RTL project (section 4.2). In case files are imported by reference format of relative path or full path, the files do not exist under project and whenever corresponding files in CWB-gui are accessed, imported files are sequentially referred (section 4.3.3).
- Simulation_work folder exists in RTL project. Directory with same name exists under directory corresponding to the project in actual file system. Work directory is created under this directory to execute simulation that is called simulation scenario, and various simulations are executed here.
- In RTL project, there is an "Other files" folder that keeps files other than source files. Virtual folder icon as shown in Figure 4-2 is used because directory corresponding to the "Other files" folder does not exist in actual file system. All files shown in "Other files" exist (section 4.2) right under the directory corresponding to RTL project.
- In intermediate project, there is a work folder corresponding to each source file but, in RTL project, there is only one work folder rtlpars_work that is common for all source files. In actual file system also rtlpars_work is generated under directory corresponding to RTL project (section 4.2).
- It is not possible to keep another intermediate project or RTL project as sub hierarchy under an RTL project, but in case of ASIC, lower hierarchy can be specified by adding a link (section 4.3.2).

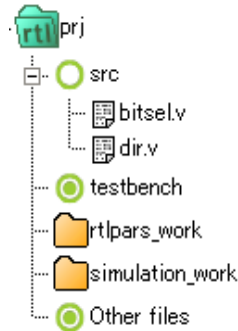


Figure 4-3 rtl project "prj"

Management file (.cyp) of applicable RTL project is generated under directory corresponding to RTL project. Information related to project created under applicable project, source file information of applicable project, information of behavioral synthesis target, information of various option settings etc. is stored in cyp file. cyp file maintains information related to corresponding directory only. The operation is not secured when this file is directly edited by editor etc.

4.1.4. Source files, include files

In an intermediate project, BDL and SystemC source file `s` can be placed in their respective `src` folder. Similarly, in RTL project, Verilog, VHDL source files can be placed in their respective `src` folder. Source files are used for source file analysis (section 7) and behavioral level simulation.

Further, BDL, SystemC include file `s` can be placed in include folder only in case of intermediate folder.

In the actual file system, there does not exist a directory equivalent to `src` or `include`. In case new source files and include files are created or imported with copy format, these files exist under directory corresponding to the project that they belong to (section 4.2). In case these files are imported with reference format of relative path or full path, the files do not exist under project and whenever the corresponding files in CWB-gui are accessed, imported files are sequentially referred (section 4.3.3).

When BDL, SystemC source files (example: `foo.c`) are either generated or imported in intermediate project, a work folder (example: `foo.c.dir`) is automatically generated. As a result of source file analysis, one or more behavioral synthesis units (hereafter, referred as process) are generated, however, work folders of all processes pertaining to analyzed source files are generated in work folder of the pertinent source file.

Further, when RTL project is generated, automatically, work folder `rtlpars_work` is generated. As a result of analyzing Verilog and VHDL source files, entered in RTL project, work folder is generated under `rtlpars_work` for all processes.

Details of process generation are explained in section 5. In actual file system, work folder and `rtlpars_work` folder of source files are generated under their respective intermediate project and RTL project (section 4.2).

4.1.5. Test bench

Testbench file used in execution of simulation can be kept in testbench folder. This file can be used for compiling simulation model.

Directory corresponding to testbench does not exist in actual file system. When new testbench files are created or imported by copy format, these files exist under the directory corresponding to the project they belong to (Section 4.2). In case these files are imported with reference format of relative path or full path, the files do not exist under project and whenever the corresponding files in CWB-gui are accessed, imported files are sequentially referred (section 4.3.3).

4.1.6. Process

One behavioral synthesis unit is referred as Process. Many programs of CWB that include behavioral synthesis are executed as a unit of this process.

Because of analyzing source codes, entered in intermediate project and RTL project, process folder is generated under source code work folder or rtlpars_work folder.

Figure 4-4 shows that as a result of analyzing source code tmp.c, two process work folders func1 and module1 are generated under work folder tmp.c.dir of tmp.c.

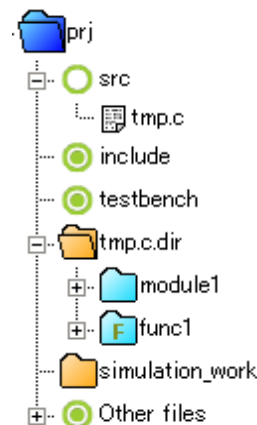


Figure 4-4 Process

Process work folder is generated under work folder (rtlpars_work in case of RTL project) of corresponding source codes even in actual file system (section 4.2).

If the process work folder is expanded, files related to the process are displayed. Figure 4-5 shows the expanded work folder of func1 process of Figure 4-4 .

Process work folder mainly consists of the following files and folders.

- Source code analysis result file (tmp.IFF in Figure 4-5)
- **Libraries folder:** a folder to store library files, which are used as input for behavioral

synthesis. (tmp_func1-auto.FLIB in Figure 4-5)

- **Constraints folder:** A folder to store constraint files that are used as input for behavioral input. (tmp_func1-auto.FCNT, tmp_func1-auto.MLMT in Figure 4-5)
- **Synthesis Analysis folder:** A folder to store result of behavioral synthesis. (tmp_func1.QOR.HTML, tmp1_func1_E.IFF etc in Figure 4-5)
- **RTL-script folder:** A folder to store RTL generated by behavioral synthesis (tmp1_func1_E.v etc in Figure 4-5)
- **Other files folder:** A folder to store CWB program output files, which are not frequently used.
- **Property Check folder:** A folder to store files related to property check. This folder is not displayed if property check functionality is not installed separately.

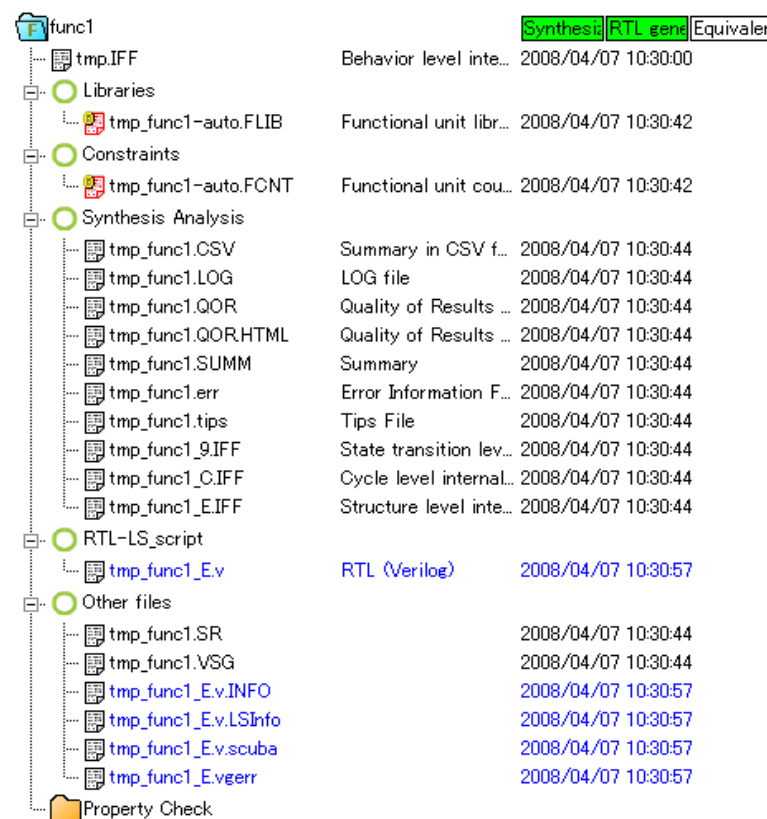


Figure 4-5 Process folder configuration

In the actual file system, all files, which are displayed under process work folder of project management window, and which are displayed in each folder except Property Check, exist in directory equivalent to process work folder (Section 4.2).

4.2. Support for actual file system

This section gives brief explanation about relation between project management window

and actual file system, as mentioned in section 4.1. is used for explaining this relation. This is explained with the example shown in Figure 4-6.

- **Workspace:** The following files and directories exist in directory (test_ws), which is equivalent to a workspace.
 - Directory (prj), which is equivalent to intermediate project and RTL project and is located under workspace
 - CWB-gui project file (test_ws.cyt)

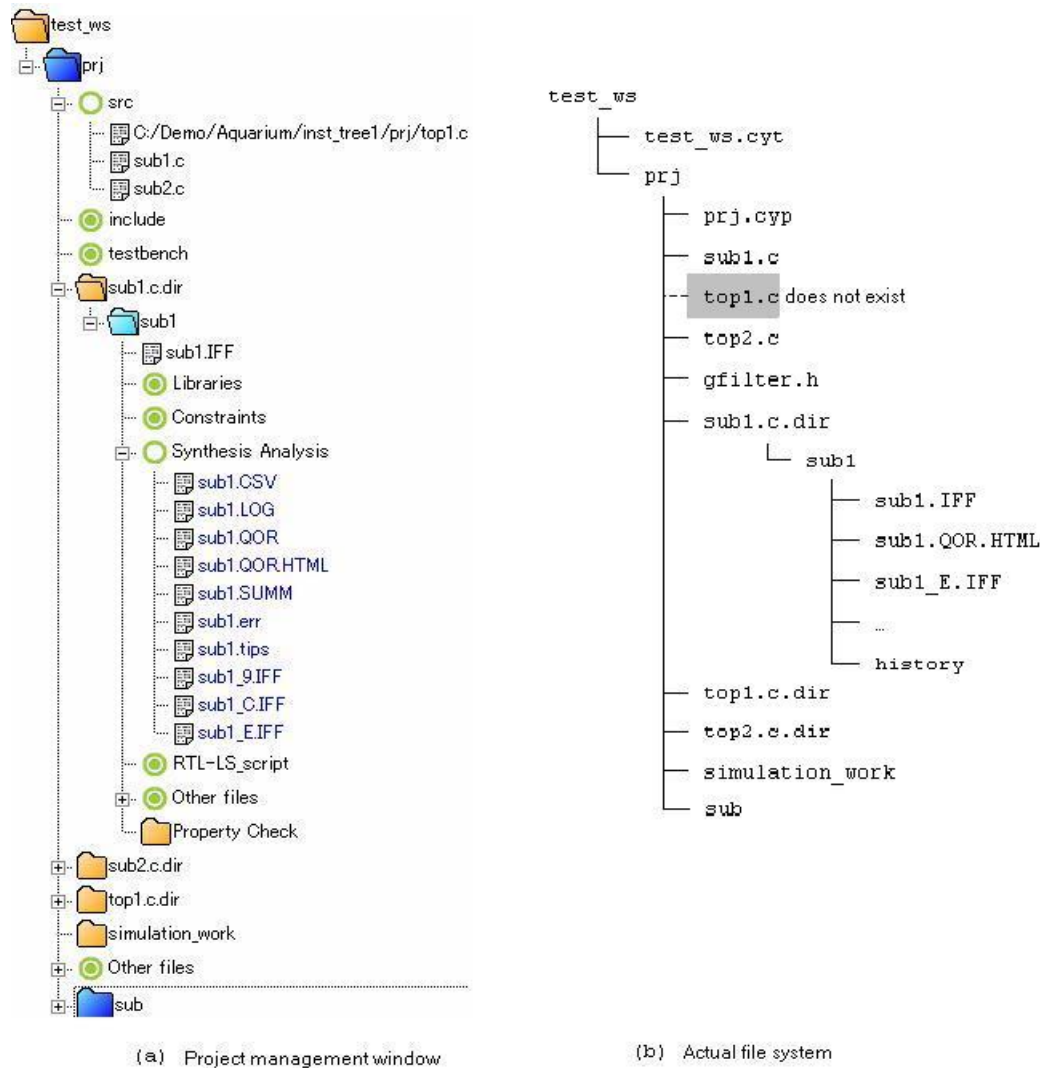


Figure 4-6 Project management window and actual file system

- **Intermediate Project:** The following files/directories exist in directory (prj), which is equivalent to intermediate project.
 - Project file of intermediate project (prj.cyp): It provides the information about

- intermediate projects.
- BDL, SystemC source files (sub1.c, top2.c). top1.c does not exist under directory as it is imported by reference formatting of full path.
- Include files (gfilter.h)
- Work directory of source files (sub1.c.dir, top1.c.dir, top2.c.dir)
- Sub-project (sub)
- "Other files" folder shown under intermediate project of project management window does not exist in the actual file system. All files in "Other files" folder exist under intermediate folder.
- **RTL project:** The following files/directories exist in RTL project directory.
 - Project files of RTL project: It provides information about RTL project.
 - Verilog, VHDL source files (excluding when reference import is being done by full path, relative path)
 - Include files do not exist in RTL project
 - Work directory of source files (rtlpars_work is the fixed name)
 - Sub project cannot be generated in RTL project.
 - "Other files" folder is treated same as intermediate folder
- **Source file work directory (In case of intermediate project):** In case of intermediate project, the following directory exists in work directory (sub1.c.dir) of source file.
 - Work directory (sub1) of all processes generated by analyzing the corresponding source files.
- **Source file work directory (For the RTL project) :** In case of RTL project, the following directories exist in work directory (rtlpars_work) of source files.
 - Work directory of all processes generated by analyzing source files entered in RTL project.
- **Process work directory :** The following files/directories exist in directory (sub1) corresponding to process work.
 - Behavioral synthesis input files (sub1.IFF)
 - Output file of each CWB program (sub1.QOR.HTML, sub1_E.IFF etc)
 - History: This directory has the history of behavioral synthesis successfully completed for relevant process.
 - Each virtual folder, excluding Property Check, displayed just under process work folder of project management window, doesn't exist in actual file system. All the files of such folders exist under process work folder.

4.3. Project editing command

This section gives brief description about project management related commands. A project is created by using each command mentioned in this section.

4.3.1. Commands related to workspace

Commands related to workspace are given below.

- **Create workspace**
 - Contents: A new workspace is created, when no project is opened in CWB-gui.
 - Implementation method: When workspace is not opened in CWB,
 - ✧ Select "new" in drop down menu "file". Specify "type" in "workspace" in opened window.
 - ✧ Select "new" from general tool bar. Specify "type" in "workspace" in opened window.
 - ✧ Select "new" from context menu of project management window. Specify "type" in "workspace" in opened window.
 - Notes: In case of ASIC, once the workspace is created, window is opened to generate intermediate project, RTL project automatically. For this operation, please refer items related to creation of intermediate project, RTL project in section 4.3.2.
- **Open Workspace**
 - Contents: When an existing workspace is opened, while there is already an opened workspace, then if the existing workspace has been saved, a new workspace will be opened automatically by closing the existing one. In case it is not saved, a dialog box will appear asking to save or close without save.
 - Implementation method:
 - ✧ Select "open" from drop down menu of "file", then select project file (.cyt) of CWB-gui.
 - ✧ Select "open" of general tool bar, then select project file (.cyt) of CWB-gui.
 - ✧ Select "open" in context menu of empty part of project management window, then select project file (.cyt) of CWB-gui.
 - Notes: In workspace opened through CWB-gui, lock file (.lock) is generated inside the directory of project file (.cyt). Please refer section 4.4.2 for details of Lock file.
- **Recently opened workspace**
 - Contents: A workspace can be selected from the history of recently opened workspaces. By default, the number of workspaces that can remain in history is 5 however, this number can be changed from Tools option. (Refer to section 16.1).
 - Implementation method: select "recent workspace" from "file" drop down menu.
- **Save all**
 - Contents: All contents of workspace and project opened in CWB-gui are saved.
 - Implementation method: When workspace is opened in CWB-gui,
 - ✧ Select "save all" in "file" drop down menu.
 - ✧ Select "save all" in general tool bar.
- **Close workspace**
 - Contents: Workspace opened in CWB-gui is closed. If it is not saved, a dialog will appear asking to save or close without saving.
 - Implementation method: When workspace is opened in CWB-gui,
 - ✧ Select "close workspace" in "file" drop down menu
 - ✧ Select "close workspace" in context menu of project management window.
- **Import project**
 - Contents: Project exported from other workspace or project within other

workspace can be embedded after copying.

- Implementation method: Select "import" in workspace of project management window, or in context menu of intermediate project, and specify the cyp file of the project to be used.
- This feature is not available in FPGA.
- **Import Hierarchy project**
 - Contents: Set up data that has hierarchical structure is imported to CWB-gui project. For details please refer Section 4.3.4.
 - Implementation method: select "import hierarchy project" in workspace of project management window, or in context menu of intermediate project.
 - This feature is not available in FPGA.

4.3.2. Intermediate project, RTL project related command

This section gives brief explanation about commands related to Intermediate project and RTL project.

- **Generate intermediate project**
 - Contents: intermediate project is generated. Intermediate project cannot be generated under RTL project.
 - Implementation method: When workspace is opened in CWB-gui,
 - ✧ Select "new" in "file" drop down menu. Specify, "Type" as "project" in window, then select parent project (workspace or intermediate project) for intermediate project that creates "generation destination".
 - ✧ Select "new" from general tool bar. Specify "type" as "project" in the window, then select parent project (workspace or intermediate project) for intermediate project that create "generation destination".
 - ✧ Select "new" in context menu of project management window. Specify "type" as "project" in window, then select parent project (workspace or intermediate project) for intermediate project that create "generation destination".
 - ✧ This feature is not available in FPGA.
- **Generate RTL project**
 - Contents: Generates RTL project. Intermediate project or RTL project cannot be created under RTL project.
 - Implementation method: When workspace is opened in CWB-gui,
 - ✧ Select "new" in "file" drop down menu. Specify the "type" as "RTL project", then select parent project (workspace or intermediate project) for RTL project that create "generation destination".
 - ✧ Select "new" from general tool bar. Specify the "type" as "RTL project", then select parent project (workspace or intermediate project) for RTL project that creates "generation destination".
 - ✧ Select "new" in context menu of project management window. Specify the "type" as "RTL project", then select parent project (workspace or intermediate project) for RTL project that creates "generation destination".

- ✧ This feature is not available in FPGA.
- **Detach from project**
 - Contents: Project will be detached from project tree of workspace.
 - Implementation method: Select "detach from project" from context menu of project.
 - This feature is not available in FPGA.
- **Delete**
 - Contents: Project is deleted.
 - Implementation method: Select "delete" from context menu of project to be deleted.
 - This feature is not available in FPGA.
- **Move project**
 - Contents: Any intermediate project or RTL project in open workspace, can be moved under workspace or other intermediate project. In case there are reference imported files by relative path in src, include and testbench folders, then relative path is corrected by matching it with project location after moving.
 - Implementation method: Select "move project" from context menu of intermediate project, RTL project in project management and specify the project where it is to be moved.
 - Constraints:
 - ✧ RTL project cannot be specified as destination for moving as there can be no project under it.
 - ✧ In option settings of each sub system, in case there is a location that sets files etc. by relative path, then these path descriptions are not corrected.
 - ✧ Moreover, sometimes the project cannot be moved when there are conflicts with constraints of section 4.4.
 - ✧ This feature is not available in FPGA.
- **Project link**
 - Contents: Link can be generated for any intermediate project and RTL project in open workspace under other intermediate project or RTL project. For details of results, please refer section 4.4. Figure 4-7 shows link to project prj placed under project rt. Link prj does not exist under the directory corresponding to project rt. In case of lower hierarchy etc., when project prj contents are referred with project rt, then project prj that is to be used is directly referred.
 - Implementation method: Select "project link" in context menu of intermediate project, RTL project in project management and specify the project that needs to be linked.
 - Constraints:
 - ✧ Sometimes the projects cannot be linked when there are conflicts with constraints of section 4.4.
 - ✧ This feature is not available in FPGA.

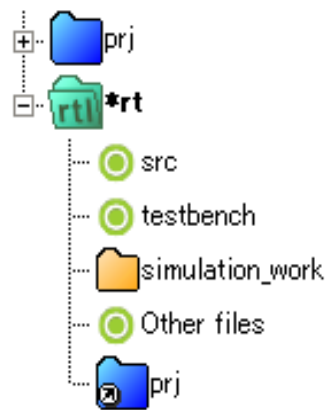


Figure 4-7 Project link

- **Import project**

- Contents: Project exported from other workspace projects or other workspace can be embedded after copying. In case there are reference imported files with relative path in src, include and testbench folders, relative path is corrected by matching project position after importing.
- Implementation method: Select "import" from context menu of intermediate project or in workspace of project management and specify cyp file of the project.
- Constraints:
 - ✧ RTL project cannot be specified as destination for import because there can be no project under it.
 - ✧ In option settings of each sub system, in case position of files etc. is set up with relative path, description of these paths is not corrected.
 - ✧ Moreover, sometimes import cannot be done when there are conflicts with constraints of section 4.4.
- This feature is not available in FPGA.

- **Project export**

- Contents: Copy the files under directory corresponding to specified project to any location.
- Implementation method: Select "export project" from context menu of RTL project or intermediate project of project management.
- This feature is not available in FPGA.

- **Import hierarchy project**

- Contents: Design data having hierarchy data is imported in CWB-gui project. For details, please refer to Section 4.3.4.
- Implementation method: Select "Import hierarchical project" in context menu of workspace or intermediate project of project management window.
- Constraints:
 - ✧ This command cannot be executed as there can be no projects under RTL

project.

- ✧ Moreover, sometimes import cannot be done when there are conflicts with constraints of section 4.4.
- This feature is not available in FPGA.
- **Show/Hide change of hidden attributes file**
 - Contents: Changes the show and hide status of source file of hidden attributes file in project management window.
 - Implementation method: Select "do not show hidden attributes file" from context menu of RTL project or intermediate project of project management window.

4.3.3. File related commands

This section gives brief explanation about file related commands.

- **Generate file**
 - Contents: A new file is created.
 - Implementation method: When workspace is opened in CWB-gui,
 - ✧ Click "Create new" on command palette.
 - ✧ Select "new" in drop down menu "file". Specify "type" as either one of "source file", "include file", "testbench file", "library file", "constraint file" or "text file", and then select the parent folder of file that creates "generation destination".
 - ✧ Select "new" from general tool bar. Specify "type" as either one of "source file", "include file", "testbench file", "library file", "constraint file" or "text file", then select the parent folder for folder that creates "generation destination".
 - ✧ Select "new" in context menu of project management window. Specify "type" as either one of "source file", "include file", "testbench file", "library file" or "constraint file", then select the parent folder for folder that creates "generation destination".
- **Open file**
 - Contents: File is opened in CWB-gui internal editor or constraint file editor.
 - Implementation method: Double click the icon of file to be opened in project management window. Relevant file contents are displayed by starting the CWB-gui internal editor in case of source files and include files, and constraint file editor (section 8.6) in case of constraint files. The files can be opened by external editor also, depending on the settings of tool options. (section 16.1.3)
- **Recently opened file**
 - Contents: A file can be opened from history of recently opened file. By default the number of files remained in history are 5 however, it can be changed by tool option (section 16.1).
 - Implementation method: Select "recent files" in drop down menu "file".
- **File import**
 - Contents: The file, which is not entered in project, is entered in a project by copying it to a location specified under project.
 - Implementation method: Select "Import files" from context menu of constraints

folder, library folder under src folder, include folder, testbench folder, processes of RTL project and intermediate project and process folders. Or, you can select "Import" on command palette. Then, specify folder of import destination in "import destination" and from where files are imported in "file name".

- While importing the source files and include files, any one of "copy", "reference by full path" or "reference by relative path" can be selected.
 - ✧ When "copy" is specified, file to be imported is copied in import directory. All GUI operations are executed on the copied file. All the files other than source files, include files and testbench files can be imported by copying. File imported through copy show the file name unit without the path description in GUI.
 - ✧ When "reference by full path" is specified, file to be imported is entered in GUI with full path. All the operations of GUI can be performed by referring files with full path for the files to be imported. Files imported by reference with full path, show the full path description in GUI.
 - ✧ When "reference by relative path" is specified, file to be imported is entered in GUI with relative path. All the operations of GUI can be performed by referring files with relative path for the files to be imported. Files imported by reference with relative path, show the relative path description in GUI.

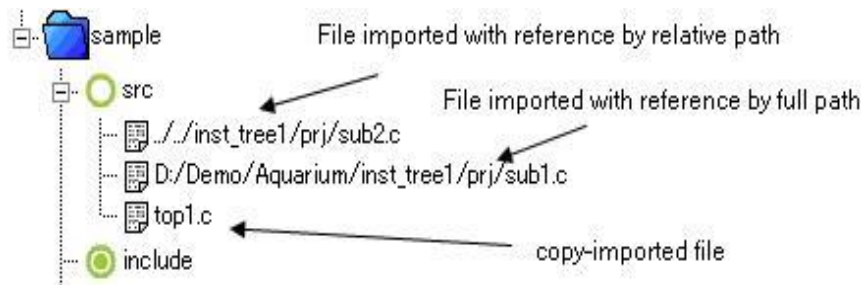


Figure 4-8 File display after import

- **Save file**

- Contents: File edited in CWB-gui internal editor is saved.
- Implementation method: when CWB-gui internal editor, in which file to be saved is opened, is active,
 - ✧ Select "save" in drop down menu "file".
 - ✧ Select "save" from general tool bar.

- **Save as**

- Contents: The file edited in CWB-internal editor is saved with different name.
- Implementation method: when CWB-gui internal editor, in which file to be saved is opened, is active, select "save as" in drop-down menu "file".

- **Close file**
 - Contents: CWB-gui internal editor is closed during editing.
 - Implementation method: when CWB-gui internal editor, in which file to be saved is opened, is active, select "close" in drop down menu "file".
- **Delete file**
 - Contents: file is deleted.
 - Implementation method: Select "Delete Files" from context menu of file to be deleted.
- **Detach from project**
 - Contents: source files, include files are detached from project.
 - Implementation method: Select "Detach from project" from context menu of the file to be deleted.
- **Properties**
 - Contents: Detailed information of source files can be viewed and attributes can be set.
 - Implementation method: Select "properties" from context menu of file to be used.
 - ✧ When "do not show this file when hide is specified" is selected, hidden attributes are added to the file and icon is grayed out. In this condition, for the project:
 - When "do not show hidden files"(Section 4.3.2) is selected, then relevant source file and corresponding work directory is not displayed.
 - When "do not show hidden files"(Section 4.3.2) is selected again, then relevant source file and corresponding work directory is displayed again.

4.3.4. Import directory

In case existing designs are used, import can be done by following methods.

1. To import the files in project one by one by using the file import functionality as explained in section 4.3.3.
2. To import directory-wise as project.

This section gives explanation about the second way.

This feature is not available in FPGA.

Conversion rule during import

In the existing design, when source codes in a directory are divided in hierarchical order, as shown in Figure 4-9 (a), these source codes can be imported as a project of CWB-gui, as shown in Figure 4-9 (b), by applying the functionality to import a directory on the root of this directory. Following are the conversion rules while importing.

- Specified root directory and files under it are copied under workspace to be imported or intermediate project.
- Each directory is converted to intermediate project or RTL project.
- When there are one or more source codes in a directory and all of these source

codes are in Verilog/VHDL, RTL project is created.

- Hierarchical structure of original directory is retained.
- Intermediate project is created also for directories not having source files. However, no source code will be entered in src folder.

Import Method

Given below is the import method.

- When CWB-gui project is opened, select "Import hierarchical project" from the context menu of workspace or intermediate project.

Restrictions while importing

This functionality has few restrictions and notes based on restrictions of RTL project.

- This functionality creates a project for sub-hierarchy of specified workspace, intermediate project. Therefore, this functionality cannot be applicable for RTL project, which cannot keep a project as sub-hierarchy.
- In case original directory, which is to be imported, has mixed source files of BDL/SystemC and Verilog/VHDL, then that directory will become intermediate project. In this case, Verilog/VHDL source files are copied to a directory equivalent to intermediate project, however, not entered in the project.
- When a certain directory is converted in to RTL project, sub-directories of original directories are not copied in RTL project directory; therefore, it is not imported as a project.
- A directory cannot be imported, when directory to be imported is parent directory or child directory of workspace currently open.

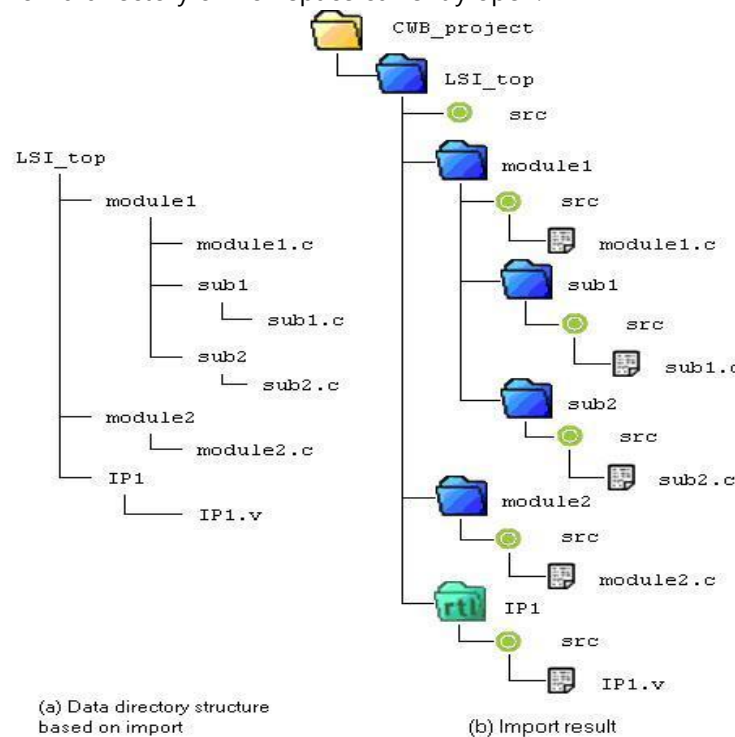


Figure 4-9 Import project

4.4. Policy and limitations of project management

Policies and limitations related to CWB-gui project structure and operations are explained in this section.

4.4.1. Policy related to project structure

CWB-gui analyzes the instance relationship between the processes entered in one workspace and then builds the instance tree. In case operations such as synthesis or model generation are executed for a certain process, then information of converting function into functional unit with this instance tree is used, and processing of synthesis is carried out automatically after searching related sub directories or functions.

In case of ASIC, Structural policy of instance tree is as following:

- Process of lower hierarchy that belongs to a certain intermediate project, is searched from project corresponding to the sub project in project management window as well as applicable project. For example, subprj1 and sub_subprj become options of lower hierarchy for processes that belong to intermediate project subprj1 as shown in Figure 4-10.
- Process of lower hierarchy that belongs to a certain RTL project, is searched from corresponding project. Since sub projects cannot be created for RTL projects, that is why processes of sub projects are not used. For example, process that belongs to rtpj becomes option of lower hierarchy for process that belongs to RTL project rtpj as shown in Figure 4-10.
- In case link to other projects is generated for intermediate projects and RTL projects, processes that belong to the applicable project and its sub projects also become options of lower hierarchy.
- Instance tree is created for every project (prj, prj1 in Figure 4-10) under workspace. Therefore, process with same name as the process that exists under the tree that has prj of Figure 4-10 as root can be created under the tree that takes prj1 as root. Details of this item are explained in section 4.4.3.

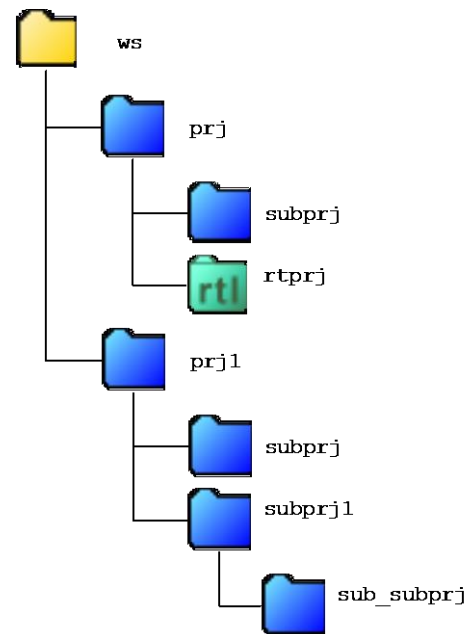


Figure 4-10 Structural policy of instance tree

For example in case it is required to build a project that uses instance tree as shown in Figure 4-11 (a), then Figure 4-11 (b) and Figure 4-11 (c) can be considered as an option for its format. The difference in (b) and (c) is that, in (b) process sub2 is in project prj in which top hierarchy process top exists and in (c) process sub2 is under sub project of prj. It is understood that tree (a) is built from the above mentioned policy of both (b) and (c).

Operations such as import and export can be carried out for the project. In case same designer executes the design of all these processes, then implementation method of (b) is easy. However, in case it is required that only process sub2 should be designed by another designer, then there can be an implementation method in which only sub2 is imported as a sub project as shown in (c), from the completed design workspace of other designer. In this way, CWB-gui provides the best structural methods according to the factors such as the kind of system to execute design, and the kind of configuration circuit to be used.

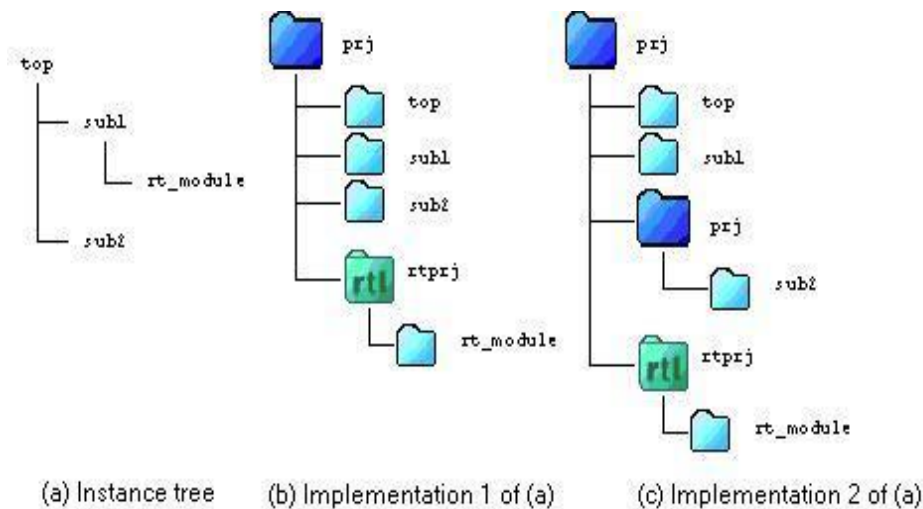


Figure 4-11 Example of project structure

4.4.2. Behavioral Policy

Behavioral policy for project management is as given below:

Expression for file path

Files are specified as arguments for options of various subsystems. In CWB-gui, path description for handling files in this way is in accordance with the policy given below.

- In case file exists in the location corresponding to sub directory of sub tree that takes the directory corresponding to open workspace in file system as root: relative path from work directory
- For others: Full path

When workspace is moved like this, the risk of not being able to resolve the path is minimized. That is, in case all related files exist under workspace, then by referring files with relative path workspace can be moved to any location, because these files are being used in closed form in workspace. Moreover, when there is a file that exists in a location other than workspace, then it is possible to move it to a workspace of same terminal and same environment because that file is referred with relative path. However, in case it was moved to a different environment, then settings for the files referred with full path should be done again after it is moved to workspace because there is a possibility that the path cannot be resolved.

In case source files, include files, and testbench files are imported, it is possible to select any one of the options for these files - get the files after copying or refer these file with full path, or refer these files with relative path (section 4.3.3). In this case, policy of this item is not applicable

Double expansion control

If a certain user opens the workspace, file with *.lock extension is generated under the directory corresponding to the workspace. When CWB-gui opens the workspace that has this lock file, then a dialogue box appears that has two options - open after deleting lock file? Or cancel this operation?

When lock file exists, there is a possibility that other users including the user himself has opened the workspace, and there is also a possibility that while opening corresponding workspace with CWB-gui, there is an abnormal exit of CWB-gui and as a result though nobody has opened the workspace but the lock file remains. In the above mentioned dialogue box, as the user name of the person who has opened the workspace, and terminal name is mentioned, so on the basis of this information please select either Open after deleting lock file? Or cancel this operation?

Save project

.cyt file is created under the directory corresponding to workspace, and .cyp file is created under directory corresponding to project. These files manage the information of workspace and project. This file is created one by one under the respective directory corresponding to the workspace or project, and only the corresponding information under each directory is managed. CWB-gui reads these files for all directories, and builds the project management window screen.

Information stored in .cyt and .cyp files is as following:

- Information of project that exists under applicable workspace/project.
- Information of files that exist under applicable project (Since there are no files shown in project management window under workspace, so .cyt file does not contain this information)
- Option settings condition of subsystem related to each process etc, or timestamp at the time of subsystem execution etc. (Since process does not exist under workspace, so .cyt file does not contain this information)

Timing of saving project

In case some changes are done for above mentioned .cyt and .cyp files, characters of corresponding workspace and project of project management window are displayed as bold characters and "*" is added in front of the characters. At the same time, back up file of project file corresponding to the applicable directory is created under directory corresponding to the applicable workspace or project.

Back up files are updated whenever user executes any operation for the applicable project. In case saving operation is done for the project, these back up files are overwrite saved as .cyt and .cyp files, and back up files are deleted.

In case there is an abnormal exit of CWB-gui due to application trouble and there were unsaved project files, then .cyt/.cyp files and back up files get mixed up under the applicable project. Please refer section 4.5 for restoring the project that used back up files.

4.4.3. Constraints

This section explains the constraints of project management.

Multiple entries of processes with same name

In CWB-gui, there cannot be multiple entries for same name processes in project tree that takes project under workspace as the root. Though Figure 4-12(a) shows multiple entries of same name processes mentioned as common in project tree that takes project topprj under workspace ws as the root, but actually this kind of project cannot be built. Depending on the various operations such as project import, analysis there is a possibility of a situation like this but in that case processing is stopped by displaying error message and due to this, process with a name same as the name of an existing process does not get entered.

It is same when there is no project in project tree. In Figure 4-13 process common does not exist in project tree that has topprj project as its root. However, in this tree, since the link is added respectively for processes (cmnprj_v1/common and cmnprj_v2/common) with same name but different project, so there is a possibility that name is not resolved normally. This kind of situation also conflicts with this limitation.

In order to avoid failure in name resolution of process, this kind of constraint is created in project tree that takes project under workspace as its root. For example, when model generation is executed for process top of (a), it cannot be judged that which option out of topprj/prj1/cmnprj/common and topprj/prj2/cmnprj/common should be used as process common. This constraint is created to avoid this kind of situation.

In case of ASIC, in case project tree corresponding to (a) needs to be built, then it will result in building of projects as shown in Figure 4-12(b) and Figure 4-12(c). Both are using link functionality of project.

- Figure 4-12(b): cmnprj under prj1 is same as (a) but, cmnprj under prj2 changes to link for prj1/cmnprj. In case synthesis or model generation is done for process sub2, prj1/cmnprj/common process that becomes link destination is referred.
- Figure 4-12(c): project cmnprj that has common process under workspace ws is newly setup and cmnprj under prj1, and cmnprj under prj2 are replaced by link to this project. In case synthesis or model generation is done for process sub1 and sub2, ws/cmnprj/common that becomes link destination is referred.

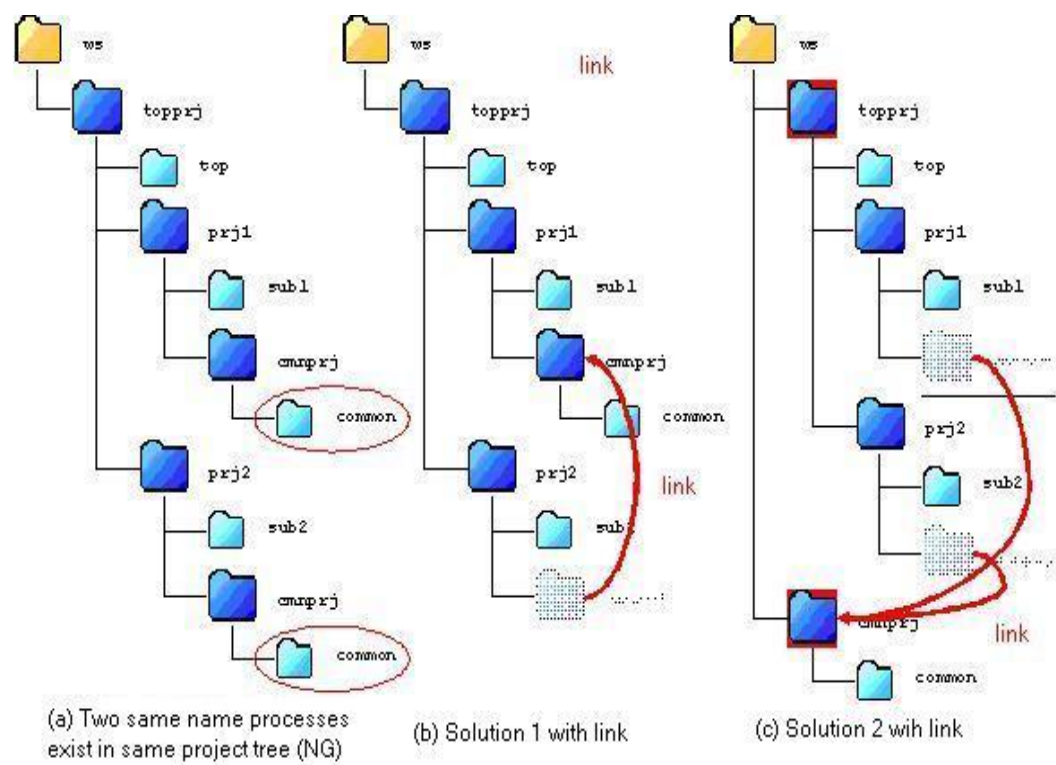


Figure 4-12 Multiple use of same name processes

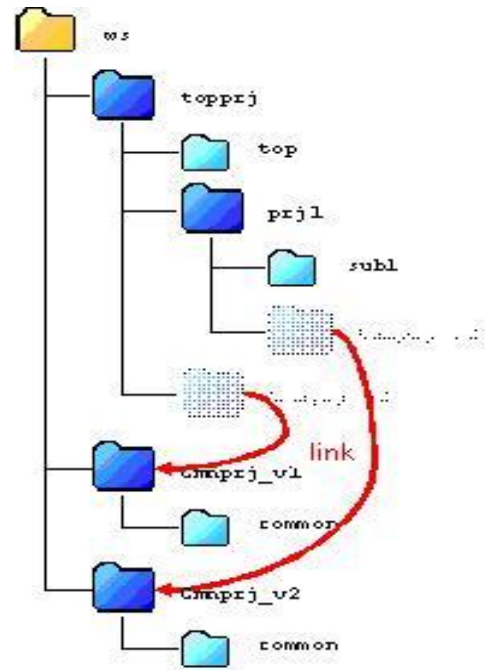


Figure 4-13 Project structure with link (NG)

Constraints related to operations of project

In case of ASIC, various operations like import, move, link etc. are performed for a project but there are constraints in these operations as well.

- When project is moved, sometimes in options of sub program of the process to be moved even if the file is specified with relative path, it does not execute recovery of relative path as per the moving destination of project automatically. Consequently, depending on the destination, reference of path breaks so it fails in path resolution when the sub program is executed. Please implement recovery of file path as per the requirement.
- Link to subproject and parent project in project tree cannot be fixed.
- In case operations where new processes can be generated such as moving of project, import, wincwb project import, jcyber project import etc. are in conflict with the constraints related to same name process of previous items in this section, then the analysis after the import process will fail regardless of the fact whether relevant operation could be executed or not.

Version management of source files etc.

The functionality to conduct version management of source files or project is not provided in CWB-gui. In future, it is being considered to provide version management mechanism that is linked with version management program such as CVS etc.

Following methodology can be considered to implement version management in present

conditions:

- Save all directories under workspace after compressing them
- Version management of only source files will be done by doing reference import of source files in location where CVS is managed.
- In workspace build a project that stores the project for which version management needs to be done, and link will be attached (Only ASIC, Figure 4-14) to project of version that needs to be referred.

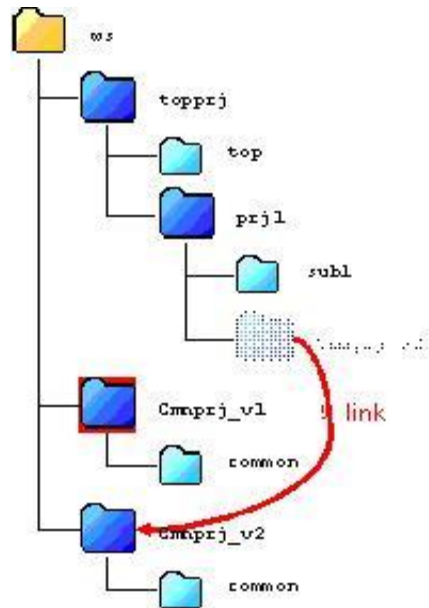


Figure 4-14 Example of version management

4.5. Restoration processing at time of abnormal exit

When any operation is performed in CWB-gui, Back up file is created for project file .cyp and workspace file .cyt (section 4.4.2). When project is saved, back up file is overwrite saved in current workspace file and project file, as workspace file and project file. Some operations are done in CWB-gui, and whenever information of workspace and project is updated, back up file is updated. Therefore, latest condition of project structure is already incorporated in back up file.

However, sometimes back up files exist even when the workspace has not been opened. This is applicable when CWB-gui has abnormal exit.

In case workspace in which back up files exist is opened in CWB-gui, a dialogue box appears that inquires whether to restore project or not. There are four buttons in the dialogue box. Respective functionality of buttons is as given below. Please select as per

these conditions:

- Yes: Build project management window by using back up file for project to be used.
- Yes to All: Build project management window by using back up files for all projects.
- No: Build project management window by using manually and explicitly saved project files and workspace files for the project. In this case, back up files of applicable project are discarded.
- No to All: Build project management window by using manually and explicitly saved project files and workspace files for all projects. In this case, back up files are discarded.

Notes

Processing related to deletion of files and directories was done but, in case that state is not explicitly saved, then the state is as following:

- Project file and workspace file: There is description related to deleted files.
- Back up files: There is no description related to back up files.
- Actual file system: Deleted files do not exist

Therefore, select "No" in above mentioned dialogue box, and even if it returns to the state before deletion of the file, then as the file does not exist in actual file system so a message appears that the file does not exist.

4.6. Reading of workspace of different versions

If the version of CWB toolkit is different, then version of workspace file and project file used by CWB-gui is also different. Also, version of intermediate file (IFF file) used by each program of CWB toolkit is also different. This section describes the behavior at the time of reading old project files.

In case old project files are read

Dialogue box appears that confirms whether to read the project file or not to read the project file. In case the option to read is selected, project file and workspace file is updated in version corresponding to CWB-gui of the operation to be performed.

In case new project files are read

When a version that is newer than workspace file and project file supported by CWB-gui is read, an error message appears and read operation cannot be performed.

When CWB-gui is started by adding "-force_read" option, the error message does not appear, and the dialogue box appears to confirm whether to read the file or not. In case the option to read is selected, project file and workspace file are updated in version corresponding to CWB-gui of the operation to be performed. However, in this case, please note that the following operation cannot be secured because sometimes the generated workspace has an abnormal format.

In case the project file created through FPGA-GUI is read in ASIC

A dialogue box appears to confirm whether to read the file or not. In case the option to

read is selected, project file and workspace file are updated in version corresponding to CWB-gui of the operation to be performed. **After that, the file cannot be read in FPGA-GUI.**

In case the project file created through ASIC-GUI is read in FPGA

You cannot read project file created through ASIC-GUI in FPGA.

In case project file containing IFF of different version is read

In this case also a dialogue box appears. Operation is selected from the following 3 buttons:

- Delete: Deletes all IFF files in workspace. Consequently, all sub programs are not executed.
- Analyze after deletion: Analyzes all source files automatically after deletion of all IFF files in workspace.
- Cancel: Stops reading of workspace.

4.7. Example of project structure

In ASIC same as above, you can create independent project structure.

Please refer Section 4.4 as the project will be built by referring some examples and policies indicated in this section. In this section hierarchical structure of design data is hypothetically shown with examples. In Figure 4-15(a), the hierarchy is illustrated. In this figure the structure is as following:

- Top most hierarchy is LSI_top
- module1, module2, module3, module4, bus1 of design module C exist under LSI_top.
- module 1 has sub1, sub2, sub3 as sub hierarchies
- LSI_top, has IP1, IP2 of IP that already exists and that is already mentioned in Verilog/VHDL.

Example of creating project in CWB-gui is shown in Figure 4-15 (b) for the project shown in Figure 4-15 (a). Hierarchical structure and characteristics of each model is reflected in project as shown in of Figure 4-15 (a).

Actually, since one person cannot design all these modules, so when this kind of project is built, project is exported with each module unit such as module1, module2, module3 etc. and is assigned to the designer in charge. When the designer finishes the designing it is imported and the design data is embedded. This flow is considered as one of the examples of operation forms.

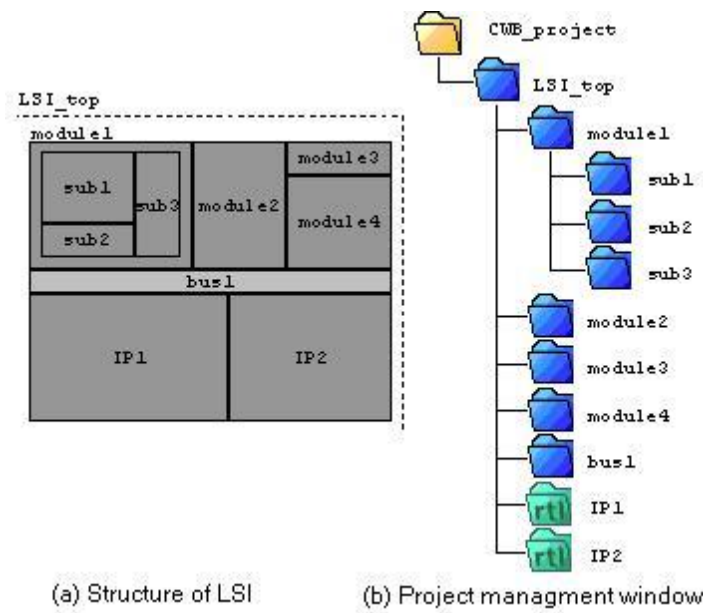


Figure 4-15 Operation example of actual design data

5. Properties setting of the project

In ASIC, you can set various attributes for intermediate and complete RTL projects. In case of FPGA, the attribute settings will be common for whole workspace.

Project attribute setting

Follow either of the below methods to start the attribute setting screen of the project.

ASIC:

- 1) It starts automatically at the time of creating intermediate project, RTL project.
- 2) Select intermediate project, RTL project on the project management window, select "Properties" from the context menu of intermediate or RTL project.

Select intermediate project, RTL project on the project management window, click "Settings" of the command palette.

FPGA:

- 1) Click "Settings" of the command palette.

Items which can be set on the properties window are described below.

5.1. Clock

You can set clock value and unit for all the processes related to the corresponding project in case of ASIC and for all the processes in case of FPGA.

These settings are used at the time of execution of behavioral synthesis using bdltran.

This value is reflected only if "Follow project property" check box is checked under the "Specify clock_constraint" in the synthesis option screen (Section 10.1) of respective process. If it is not checked, you can do other settings for the process. By default, this check box is selected.

If you change clock value of the project properties, the value for the process will also change if the check box is checked ON.

5.2. Device

You can choose either FPGA or ASIC. If you choose FPGA, you will be able to specify details of the device such as family name etc. These settings are used for options such as automatic generation of library file and constraint file (Chapter 8).

5.3. Basic Library

You can do settings of basic library for all the processes related to the corresponding project in case of ASIC and for all the processes in case of FPGA.

These settings are used at the time of execution of behavioral synthesis using bdltran.

If the setting is done, settings of basic library for all the processes related to the corresponding project in case of ASIC and for all the processes in case of FPGA will also change.

Follow the below steps to do above settings:

- "Settings" button
 - If you select "Generate": Start the option setting screen of "Auto generate basic

library file”(BLIBgen, section 8.2). The program is executed when you click “Execute” button. A dialog box will appear to confirm whether you want to use setting of temporary basic library until execution of BLIBgen completes.

- ✧ If you select “Yes”:
 - A file specification dialog box will appear to specify the file for temporary basic library. Please specify the file.
 - After the temporary file is specified, BLIBgen will start.
 - After completion of BLIBgen, a dialog box will appear to confirm whether you want to replace the basic library settings with the generated file. Click “Yes” to replace and “No” if you don’t want to replace.
- ✧ If you select “No”: Start BLIBgen. Now, you will not be able to change settings of generated basic library file under synthesis option.
- If you select “Select”: A file specification window will appear to specify the basic library file. Please specify the file.
- “Stop the generation” button: If you select “Generate”, by clicking this button you can stop the generation at any time during the execution of BLIBgen.
- “Delete” button: Deletes specification of the basic library file.

5.4. Functional unit library

You can do settings of functional unit library for all the processes related to the corresponding project in case of ASIC and for all the processes in case of FPGA.

These settings are used at the time of execution of behavioral synthesis using bdltran.

If the setting is done, settings of functional unit library for all the processes related to the corresponding project will also change.

Steps are same as the basic library. Please refer section 5.3 for further details.

5.5. Changes to the sub module

In ASIC, you can specify whether the properties of this project to be inherited to the sub projects or not by checking the “set options to the lower projects” check box.

By default, it is OFF. In this case the setting values of this project will not reflect in the sub module. For instance, in project structure of figure 4-15 (b), project properties of LSI top project is specified however, you can specify different properties to the sub projects module1 and module2.

If it is ON, settings of the project are reflected to all the sub projects. When this check box was checked and properties of LS1 top project are set, all the child projects of that project module1, module2 sub1 will inherit properties of LSI top project. You cannot separately set properties for sub projects.

6. Policy of Option settings screen

Settings of options can be done by starting option settings screen for each subsystem of CWB toolkit that can be started from CWB-gui. Behavior of option settings screen is based on a fixed policy. This policy is explained in this section.

Options screen of behavioral synthesis is shown in Figure 6-1.

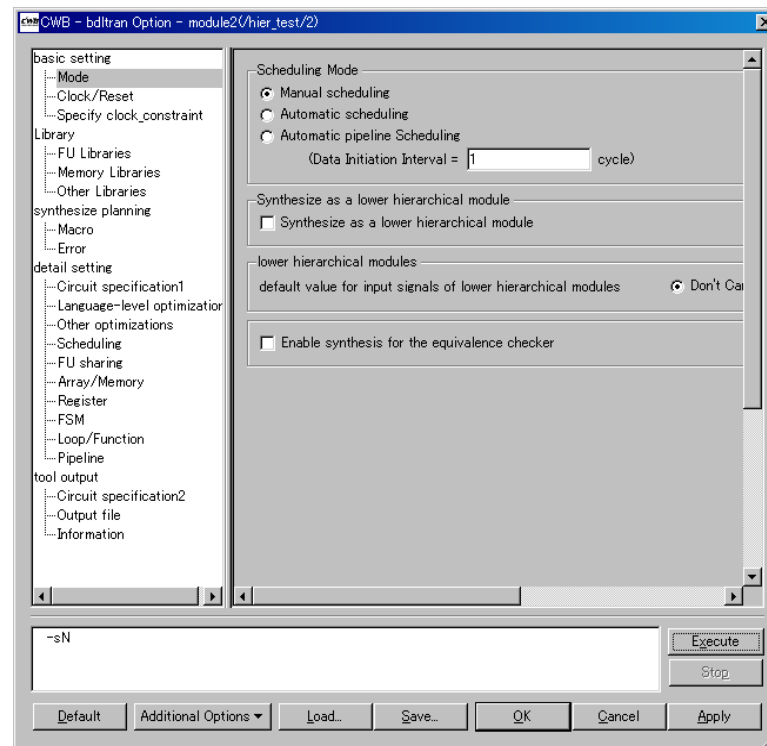


Figure 6-1 Options screen of operation synthesis

Each component of options screen is as following. However, there are some components that do not exist in options screen depending on the subsystem.

- Upper screen: It is the part that implements option settings for subsystem.
- Lower screen text box: According to the settings in the option settings part of the upper screen, related options are displayed as character string. At the time of executing subsystem, character string mentioned in this text box is passed as options in actual. In the text box, options are manually entered and by clicking "Execute" button, character strings that cannot be shown in the components that exist in option settings part of upper screen are entered, and it can be passed as option at the time of subsystem execution. However, since manually entered character string is not saved in the project file so manually entered character string

gets cleared once the options screen is exited.

- Execute button: Character string of text box in lower screen is given as argument to the subsystem and then subsystem is started.
- Stop button: In the option screen of bdltran, option screen is not closed even if subsystem is started by clicking execute button. In this case, the executed subsystem is stopped. For other subsystems, since options screen closes when execute button is clicked, it can be stopped with stop functionality of Operations toolbar or drop down menu.
- OK button: Option settings screen is closed after incorporating option settings. When option settings screen is opened and the changes in options is not implemented until OK is clicked, then even if the option settings screen is closed by clicking OK, it is considered that the option settings were not changed. In case changes of the options are implemented, it is considered that option settings were changed and as a result execution status (for example even if it is executed) of target subsystem becomes unexecuted and even when operation related to execution of subsystems hereafter is executed, execution is not skipped.
- Apply button: Option settings are incorporated but option screen is not closed.
- Cancel button: Option settings screen is closed. When operations are carried out for option screen after option screen is opened or after the last time options were saved, a dialogue box appears to inquire to save those changes.
- Default button: Returns option settings to default.
 - Default options of program can be changed with "Tools" drop-down menu in "default option settings" for each program of Bdlpars, scpars, bdltran, veriloggen, vhdngen. In case it is changed, changed value is used as the default value.
- Additional options button: By clicking this button a text box appears in which any character string can be entered. After that, entered character string can be given as an argument of program by clicking checkbox of "set following character string to options". Due to this, an option that cannot be assigned in GUI component operation of upper part of option screen can be provided in the program.
- Save button: Character string of the options of text box in upper screen is saved as text file.
- Load button: Text file in which options are described is loaded and analyzed, and is incorporated in the option screen. In case there is an option in the text file that does not have the component to be specified in option screen, the applicable option is ignored.

In option screen multiple pages can be opened simultaneously. However, there are following limitations:

- Simultaneously 5 pages can be opened in option screen.
- In case synthesis option screen is opened in a certain process:
 - Option screen of relevant process, and other processes in the hierarchy of the relevant process cannot be opened.
 - Option screen of related simulation scenario cannot be opened.
 - Option screen of related source file cannot be opened.
- When the option screen of simulation scenario and source file is open, option

screen of related processes etc cannot be opened in the same way.

For few options, the value can be shared between processes under the project by setting it in the project properties. For details please refer section 5.1.

7. Source file analysis

Source files analysis can be done for BDL/SystemC/Verilog/VHDL. Further, behavioral level simulation can be performed for BDL. This section gives explanation about these operations.

7.1. BDL analysis

BDL source files should be analyzed before implementing behavioral synthesis. This analysis is done by executing a program, bdlpars.

bdlpars option settings

bdlpars option settings screen can be opened by any of the following ways. (Figure 7-1)

- 1) When BDL source file (.c and .cpp are extensions) is selected in project management window, select "analyze options" from context menu of source code.
- 2) When BDL source file (.c and cpp are extensions) is selected in project management window, select "analyze options" from drop-down menu "synthesize".
- 3) Select source file on project management window and click "Analysis Options" of the command palette.

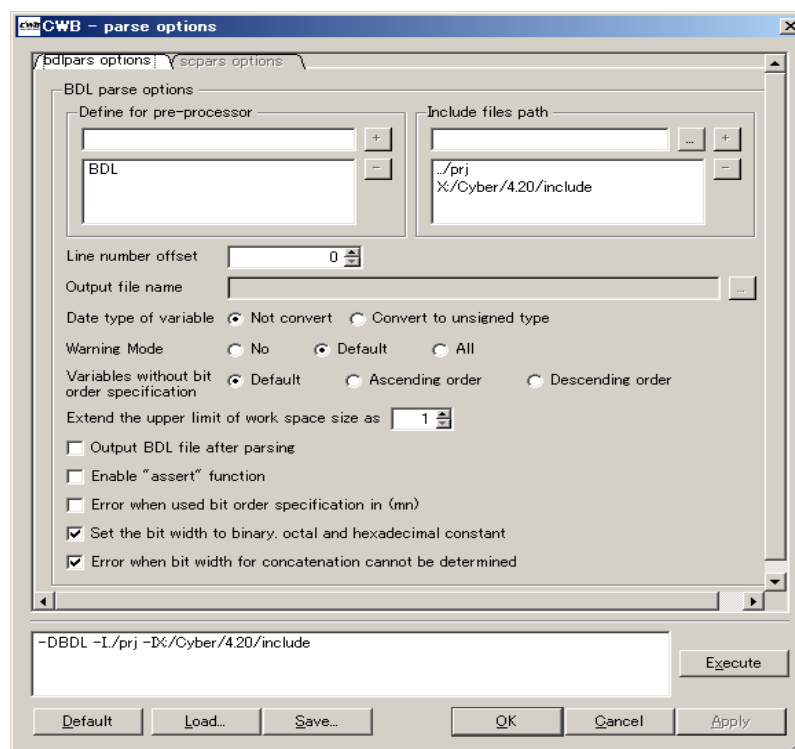


Figure 7-1 bdlpars option setting screen

At the top of option setting screen, there are two tabs, "bdlpars options" and "scpars options". "bdlpars options" is an option settings screen for bdlpars. "scpars options " will be described later in section 7.2.

bdlpars execution

bdlpars can be executed by any of the following ways.

- 1) When BDL source file (.c and .bdl are extensions) is selected in project management window, select "analyze" from context menu of source code
- 2) When BDL source file is selected in project management window, select "analyze" from drop-down menu "synthesize".
- 3) When BDL source file is selected in project management window, select "analyze" from operations toolbar.
- 4) Open bdlpars option setting screen for BDL source files and press "execute" button.
- 5) Select source file on project management window and click "Analysis Options" of the command palette.

Error file display of bdlpars

Error file at the time of bdlpars execution can be displayed by selecting "open error file" from context menu of source code in the state where BDL source file is selected in project management window.

In case of any errors, the applicable row of source file can be highlighted by selecting the row number in error file.

Generated process

By executing bdlpars, process work folder is created inside source file work folder.

In BDL, one process is described in one file. Therefore, fundamentally, one process is generated by executing bdlpars. Definition of process module1 in BDL is shown in Figure 7-2 (a). Figure 7-2 (b) is the output of the corresponding bdlpars execution displaying under "Project" tab of the project management window. Work folder module1 of process is created under work folder module1.c.dir that corresponds to source file module1.c.

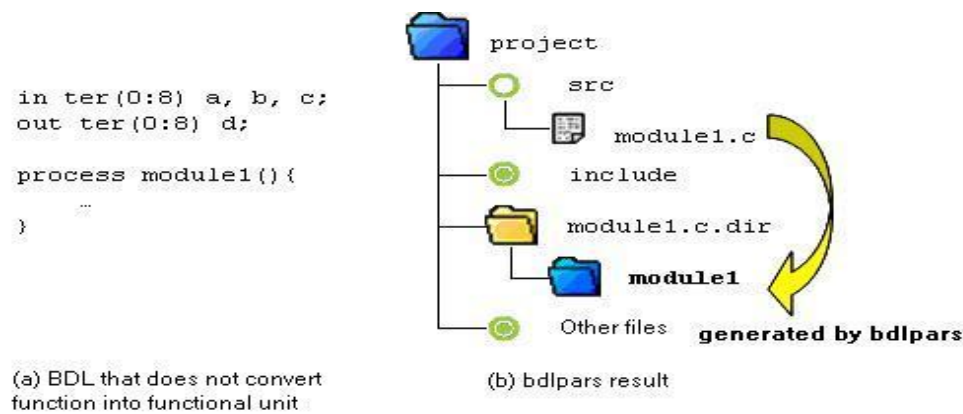


Figure 7-2 Behavior of bdlpars against BDL, that does not convert function into functional unit

If functionality of Converting function into a functional unit is being used in BDL, process folder is created for each process function and function converted to functional unit, under source file of work. In the BDL given in Figure 7-3 (a), module1 process is defined. At the same time, module1 calls function func1, and operator attribute is given to this func1. Figure 7-3(b) is the output of the corresponding bdlpars execution displaying under "Project" tab of the project management window. Process work folder module1 and work folder func1 for function converted to functional unit are generated under work folder module1.c.dir that corresponds to source file module1.c. Regarding the details of functionality of functional unit conversion, please refer to Cyber Reference manual.

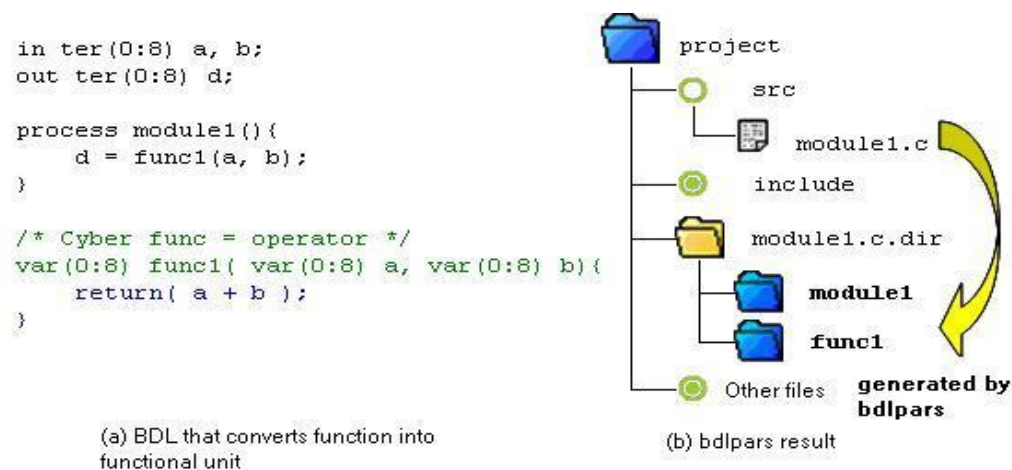


Figure 7-3 Behavior of bdlpars against BDL, that converts function into functional unit

Execution control

When there is normal exit of bdlpars, status of applicable source file becomes analyzed. As a result, when bdlpars is executed again, it is skipped. In the following operation analyzed

status is analyzed and it becomes analysis pending.

- After running the options in bdlpars option setting screen press OK.
- Edit source file and include file.

7.2. SystemC analysis

SystemC source file should be analyzed before implementing behavioral synthesis. Analysis can be carried out by executing the program scpars.

scpars option setting

Option setting screen of scpars can be executed by any of the following ways.

- 1) When SystemC source file (.cpp and .sc are extensions) is selected in project management window, select "analyze options" from context menu of source code.
- 2) When SystemC source file (.cpp and .sc are extensions) is selected in project management window, select "analyze options" from drop down menu "synthesize"
- 3) Select source file, process on the project management window and click "Analysis option" of the command palette.

Scpars execution

scpars can be executed by any of the following ways.

- 1) When SystemC source file (.cpp and .sc are extensions) is selected in project management window, select "analyze" from context menu of source code.
- 2) When SystemC source file is selected in project management window, select "analyze" from drop-down menu "synthesize".
- 3) When SystemC source file is selected in project management window, select "analyze" from operation toolbar.
- 4) Open scpars option setting screen for SystemC source files and press "execute" button.
- 5) Select source file on project management window and click "Analysis option" of the command palette.

Display Error file of scpars

Error file at the time of scpars execution can be displayed by selecting "open error file" from context menu of source code in the state where System C source file is selected in project management window.

In case of any errors, applicable row of source file can be highlighted by selecting row number in error file.

Generated process

If scpars is executed, one or more processes are generated. Under "Project" tab of project management window, process work folder (to be executed with scpars) is generated under work folder of source files. (Figure 7-4).

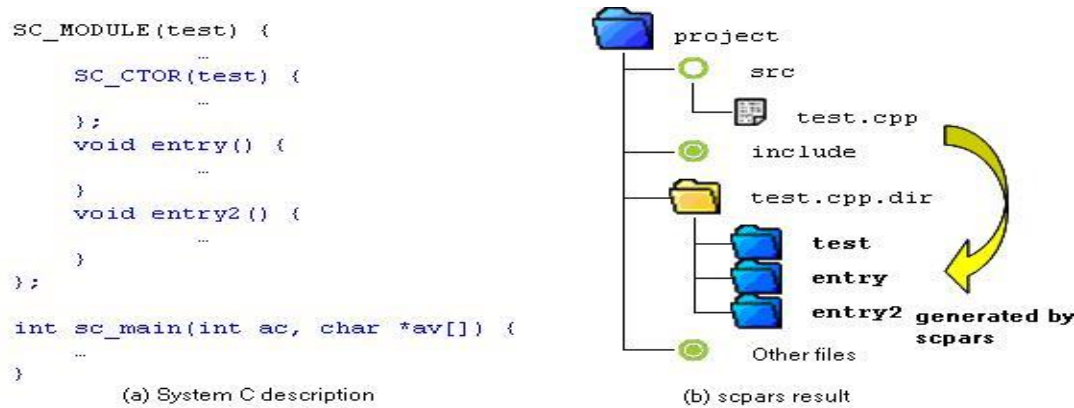


Figure 7-4 Behavior of scpars

Execution constraints

When there is normal exit for scpars, status of applicable source file becomes analyzed. As a result, when scpars is executed again, it is skipped. In the following operations analyzed status is analyzed and it becomes analysis pending.

- After running the options in scpars option setting screen, press OK.
- Edit source file and include file.

7.3. RTL analysis

Verilog/VHDL source file should be analyzed before implementing behavioral synthesis. Analysis is done by executing rtlpars program.

RTL registration (FPGA only)

In FPGA, in order to save the RTL, "RTL Module" must be created and RTL source file corresponding to RTL module must be registered.

- Create RTL module: Under "Source file" tab of project management window. Select "Add RTL Module" from context menu of "RTL".

Source file registration: Select "Create new" or "Import" from the context menu of RTL module. Or, select the RTL module and click "Create New" or "Import" on the command palette.

rtlpars option setting

There is no option setting screen to set options for rtlpars.

rtlpars execution

Execute rtlpars by selecting "Analysis" from the context menu of RTL module in case of FPGA and of src folder in case ASIC. You can also start the execution by clicking "Analysis" of command palette after selecting src folder or RTL module.

If rtlpars is executed, file sequence selection screen will open. In case of VHDL source codes,

sequence of inputting the file is important, so file sequence is decided in this screen. After this, top module selection screen is displayed. Now, process is generated by selecting top module, and starting rtlpars by pressing "Output IFF".

rtlpars re-execution

Re- execute rtlpars by selecting "Re-analysis" from the context menu of RTL module in case of FPGA and of src folder in case ASIC.

When reanalysis is implemented, input to file selection screen and top module selection screen is provided from most recent execution history of rtlpars and then rtlpars is executed.

Error file display of rtlpars

Error file at the time of rtlpars execution can be displayed by selecting Open error file module in case of FPGA or src folder in case of ASIC after selecting RTL module in project management window.

In case of any errors, applicable row of source file can be highlighted by selecting row number in error file.

Generated process

One or more processes are generated when rtlpars is executed. In case module-having sub-hierarchy is selected in highest module selection screen, multiple processes are generated.

If rtlpars is executed, in "All output" tab of project management window, process work directory is generated under rtlpars_work (Figure 7-5)

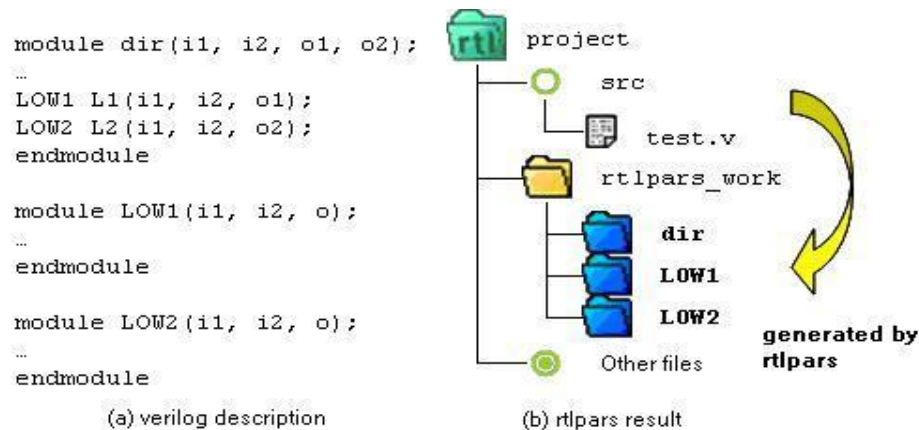


Figure 7-5 Behavior of rtlpars

Simultaneously generating files

When rtlpars is executed, there are instances when functional unit library file (FLIB), and

functional unit count file (FCNT) are generated in the process work folder at the same time. In the option setting of behavioral synthesis, it is not necessary to explicitly provide these files as argument. During behavioral synthesis, these files are automatically passed as options (Section 9).

Restrictions

Please make a note of the following restrictions.

- 1) In case files are added or deleted in an RTL project, execute rtlpars at least once by selecting "analyze" from context menu of src folder.
- 2) In case module definition etc are deleted by editing the source file manually, execute rtlpars at least once by selecting "analyze" from context menu of src folder.

Execution control

Even if rtlpars is executed again after the normal exit of rtlpars, rtlpars is executed without getting skipped like other analysis programs.

7.4. Call tree, Instance tree

Instance tree shown in instance view of the project management window is generated after the analysis of source file. After the analysis of source file, a process is executed to find the sub hierarchical processes in intermediate project/RTL project of that process, when it is ascertained that the process has a sub process. Further, in case of intermediate project, a process is executed to find the sub hierarchical processes from intermediate processes/RTL project one layer below.

In addition to this, in case function definition part or function call part of source code is encrypted, corresponding part is not displayed in call tree.

7.5. Overflow check

Occurrence of overflow or underflow can be investigated according to the operation described in the source file. The program used for this purpose is check_overflow.

Overflow check can be executed any time if done after analysis.

Option settings of check overflow

Option setting screen of check_overflow can be opened by selecting "check overflow" from context menu of the process of project management window (Figure 7-6).

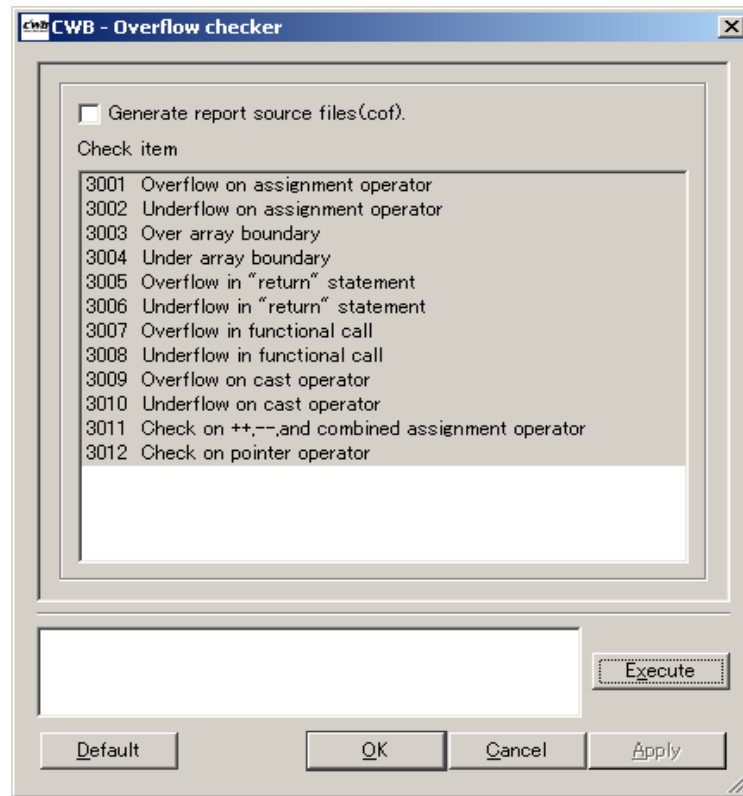


Figure 7-6 Overflow check option screen

Execution of check overflow

check_overflow can be executed by opening check_overflow option setting screen and pressing "execute button".

Error file display of check overflow

Error file at the time of check_overflow execution can be displayed by selecting "overflow check" in "open error file" from context menu in the state where process is selected in project management window.

In case of any errors, applicable row of source file can be highlighted by selecting row number in error file.

Generated file

When option of "create file (.conf) that adds symbol to source file" is enabled in check_overflow option screen, and check_overflow is executed, file with ".cof" extension is generated in "other files" under process for check_overflow execution. This file adds symbol to the part that was in conflict with overflow check for the entered source file.

8. Constraint file generation

There are 9 types of constraint files. Please refer to Cyber Reference Manual for details.

- **Functional unit constraint file (LMT):** It specifies the information and restrictions related to functional unit used in synthesis. In terms of functionality, it will be the subset of FLIB/FCNT. Basically, FLIB/FCNT will be used in place of LMT.
- **Memory constraint file (MLMT):** It specifies the information and constraints related to memory used in synthesis. In terms of functionality, it is a subset of MLIB/MCNT. Basically, MLIB and MCNT will be used in place of MLMT.
- **Port constraint file (PLMT):** It specifies the information and constraints related to input output ports of module used in synthesis.
- **Port related file (PREL):** It specifies the relation between input output ports of module and allocated input output variables used in synthesis. It is used with PREL.
- **Basic library (BLIB):** It specifies the information related to register, multiplexer, and decoder etc used in synthesis.
- **Functional unit library file (FLIB):** It specifies the information of functional unit used in synthesis.
- **Functional unit count file (FCNT):** It specifies the constraint of functional unit used in synthesis. It is used with FLIB.
- **Memory library file (MLIB):** It specifies the information of memory used in synthesis.
- **Memory count file (MCNT):** It specifies the constraints of memory used in synthesis. MCNT will have information of memory, register, and register array. If any memory related information exists in description of MCNT, it will be used with MLIB. If that is not the case, it will be used by MCNT.

Hereafter, constraint file names may be referred by notations (LMT, MLMT etc) as given in the parenthesis above.

Sometimes these constraint files become a requirement while behavioral synthesis. When operations such as addition, subtraction in source file are carried out, LMT or FLIB/FCNT are required. When memory is to be accessed, MLMT or MLIB/MCNT are required. When area or delay value needs to be specified for register and multiplexer, BLIB is required. Some of these constraint files can automatically output template files and specify delay value and area value. This section provides explanation regarding operations related to constraint files.

8.1. Use of standard BLIB and standard FLIB

Sample of FLIB & BLIB can be automatically generated by using the program inside CWB Tool kit, so there is no need to prepare it in advance in case the user wants to carry out behavioral synthesis for the time being.

However, while using CWB for designing the product, as it is desirable to standardize BLIB & FLIB used at each location and so it is recommended that standardized BLIB & FLIB

should be prepared in advance and distributed for use in each department within the company and offices. These files are referred as Standard BLIB and Standard FLIB.

Standard BLIB and Standard FLIB is bundled as sample in \$CYBER_LIB.

Settings of Standard BLIB and Standard FLIB are done for intermediate project and RTL project. Operation method and behavior is different as per the specified timings. This is described in the section given below.

8.1.1. Specifications of project creation

When new intermediate project and RTL project is created, standard BLIB and standard FLIB can be specified. The specifications of creation of project, and standard BLIB and standard FLIB at the time of project creation are implemented in the following way:

Project creation

When "new" command is selected from "file" drop-down menu, new window appears and specifications of project to be created are specified. Click "OK" after doing the settings mentioned below.

- Select "project" or "RTL project" in "Type"
- Input any name in "project name"
- In "generation destination" specify the parent project or workspace of the project to be created

Select "yes" in the dialogue box when "OK" is pressed.

Specifications of standard BLIB and standard FLIB

"Project attributes settings" window is displayed. Standard BLIB and standard FLIB that are prepared in advance, are specified here. For setting methods, see sections 5.3, 5.4.

When standard BLIB and standard FLIB are specified in project creation, and if the process is created under the corresponding project, standard BLIB and standard FLIB specified in the project are automatically assigned as synthesis options of that process.

Value can be changed and deleted for standard BLIB and standard FLIB options that are automatically specified, by setting the synthesis options for each process separately.

8.1.2. Change of standard BLIB and standard FLIB value

In case it is required to specify the value of standard BLIB and standard FLIB again because it was not specified at the time of creating new project, or in case it is required to specify a different value for existing standard BLIB and standard FLIB then it is possible to reset that value. This operation and policy is explained here.

Change of project properties

It is possible to change the value of standard BLIB and standard FLIB in "project attributes settings" screen that appears when "properties" is selected from context menu of intermediate project or RTL project.

Operation method and setting items are basically same as mentioned in section 8.1.1.

When standard BLIB and standard FLIB are specified in properties of the project, and if process is generated under the corresponding project, standard BLIB and standard FLIB specified in the project are automatically assigned as synthesis options of that process.

In case there is already a process under the corresponding project, the synthesis option of that process also changes automatically. The behavior in this case is as explained below:

- In case value is specified when the value is not already specified in properties of the project
 - Standard BLIB: The value set for all processes under the corresponding project, is added as synthesis option. Even if any BLIB is already specified in synthesis option, it can be replaced with the value specified in properties.
 - Standard FLIB: The value set for all processes under the corresponding project, is added as synthesis option. In case any FLIB is already specified in synthesis option.
 - ✧ In case FLIB of existing settings is different from FLIB that was specified in properties in the file system: Adding to the FLIB of existing settings, new FLIB set in properties is added to synthesis option.
 - ✧ In case FLIB of existing settings is same as FLIB that was specified in properties in the file system: synthesis option does not change.
- In case value is deleted when the value is already specified in properties of the project
 - Standard BLIB : All settings related to BLIB are deleted from synthesis option for all processes under the corresponding project.
 - Standard FLIB : In case any FLIB is already specified in synthesis option for process under corresponding project,
 - ✧ In case FLIB of existing settings is different from FLIB deleted in properties in the file system: synthesis option does not change.
 - ✧ In case FLIB of existing settings is same as FLIB that was specified in properties in the file system: Only applicable FLIB is deleted from synthesis option.
- In case value is changed when value is already set up in properties,
 - Standard BLIB: The value after change is added as synthesis option for all processes under corresponding project. Even if any BLIB is already specified in synthesis option, it can be replaced with the value set up in properties.
 - Standard FLIB: The value after change is added as synthesis option for all processes under corresponding project. For respective FLIB specified in properties screen here,
 - ✧ When new FLIB is added: It is same as "in case value is specified when the value is not already specified in properties of the project"
 - ✧ When FLIB of existing settings is deleted: It is same as "in case value is deleted when the value is already specified in properties of the project".

This change is not incorporated in the subproject of project for which value is changed. The change of value becomes effective only for the project that is to be changed.

Moreover, the values of Standard BLIB and Standard FLIB options that were automatically specified, can be changed or deleted by setting synthesis options for each process individually.

8.2. Automatic generation of Basic library file (BLIB)

BLIB specifies the information related to register, multiplexer, decoder etc used in synthesis. For details, please refer to Cyber Reference Manual.

Accuracy of behavioral synthesis is improved by providing BLIB as input during behavioral synthesis.

BLIB can be described manually from scratch, however, in CWB, there is functionality (program BLIBgen) to generate BLIB template and set the information such as delay value, area value etc of each part simultaneously.

BLIBgen usage conditions

BLIBgen requires the following logical tools as external tool.

- Windows : Synplify, Xilinx ISE, Quartus II
- LINUX : Design Compiler or Synplify

BLIBgen option settings

BLIBgen option setting screen can be started by any of the following methods.

- 1) Select "automatic generate basic library" from context menu of intermediate project or RTL project in project management window.
- 2) Select "automatic generate basic library " from context menu of process in project management window.

Further, you can also start it without opening the project by selecting "Generate Basic Library (BLIB)" from the context menu at any blank space of the project management window.

BLIBgen execution

For executing the BLIBgen, open option setting screen of BLIBgen and press "execute" button.

Generated file

If BLIBgen execution has a normal exit, a new BLIB file is created and is registered in the project.

If BLIBgen is executed by above mentioned method 1, BLIB file gets stored in "Other files" folder of intermediate project/RTL project, on which execution is performed in project management window.

If BLIBgen is executed by above mentioned method 2, BLIB file gets stored in Library folder of process to be executed in project management window.

8.3. Automatic generation of functional unit related constraint file (FLIB/FCNT)

FLIB specifies the information of functional unit used in synthesis. FCNT specifies the functional unit constraints used in synthesis. FLIB and FCNT are used together. For details, please refer to Cyber Reference Manual.

FLIB/FCNT can be described manually from scratch, however, in CWB, there is a functionality to output FLIB/FCNT template. This section gives explanation about such functionalities.

8.3.1. Automatic generation of Functional Unit Library file (FLIB)/Functional Unit Count file (FCNT)

Automatic generation of FLIB/FCNT, can be executed by providing "-Zflib_fcnt_out" option for bdltran. By executing "bdltran -Zflib_fcnt_out" for intermediate file generated after analysis of source code, intermediate file is analyzed and type of functional units required in behavioral synthesis are determined and FLIB/FCNT template is output.

bdltran -Zflib fcnt out option setting

bdltran -Zflib_fcnt_out option setting screen can be started by any one of the following

- 1) Select "automatic generate functional unit library/constraint (FLIB/FCNT_" from context menu of process in project management window.
- 2) When process to be executed is selected in project management window, Select "automatic generate functional unit library/constraint (FLIB/FCNT)" from drop down menu "synthesize".
- 3) Select "Generate Constraint file" from the command palette after selecting the process to be executed on the project management window.

bdltran -Zflib fcnt out execution

For executing bdltran -Zflib_fcnt_out, start option setting screen, and press "execute" button.

Generated file

If bdltran -Zflib_fcnt_out execution has a normal exit, FLIB and FCNT are generated in Library folder, constraint folder of the process respectively. Generated FLIB and FCNT automatically get set as an argument of behavioral synthesis option.

Option settings can be changed for selection of the option of generating the respective files in FLIB and FCNT, and also for contents of generated FLIB. Please refer section 8.3.2 for details.

Notes

- In automatically generated FLIB,
 - Appropriate value is not set for functional unit delay and area. Automatic setting

processing of delay and area should be executed as mentioned in section 8.4 or FLIB should be edited by referring section 8.6.

- In automatically generated FCNT,
 - Appropriate value is not set for functional unit constraint count. FCNT needs to be edited by referring to section 8.6.
- If scheduling mode is changed in option setting screen, scheduling mode of behavioral synthesis and memory library/constraint automatic generation is changed in conjunction with each other.
- In case function is converted into functional unit also, FCNT file is required but even if automatic generation of FLIB/FCNT is used, FCNT description is not generated for converting function into functional unit. In FCNT for converting function into functional unit, behavioral synthesis program generates a sample at the time of behavioral synthesis. Please refer section 9 for details.

8.3.2. Linking of standard FLIB settings and FLIB automatic generation

The operation of automatic generation screen of FLIB/FCNT of section 8.3.1 differs on the basis that whether standard FLIB is set for the project or not.

As shown in figure 8-1, at the time of opening automatic generation screen of FLIB/FCNT, you can select output for both "FLIB" and "FCNT" in the "Output file".

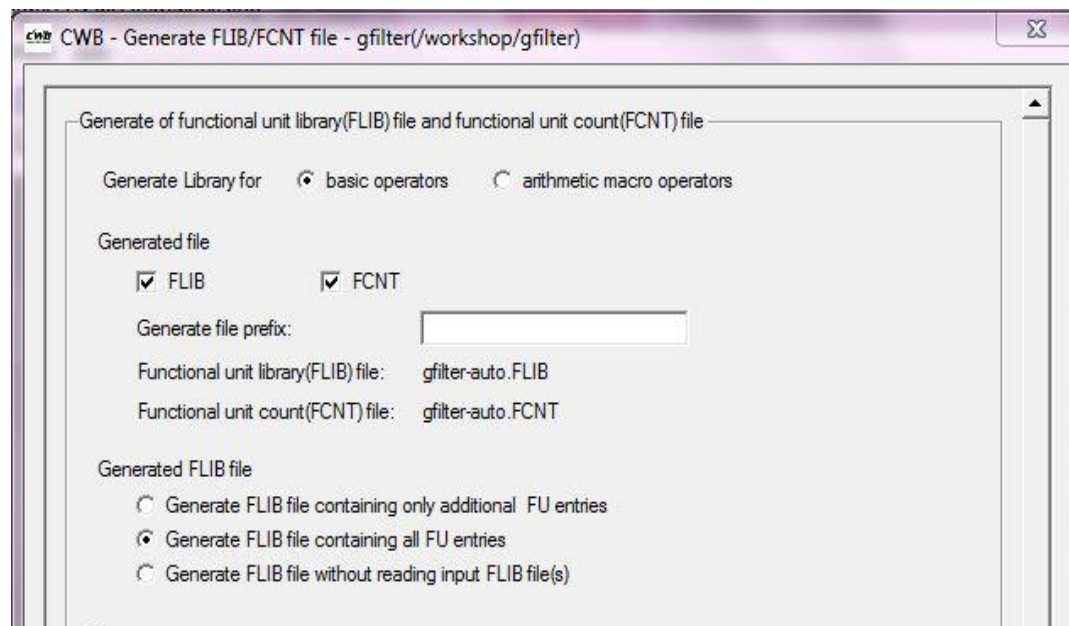


Figure 8-1 Screen where standard FLIB is not setup

When FLIB is output, the generated FLIB is different depending on the specification of items of "FLIB file" in the following way:

- Select "Generate FLIB file containing only additional FU entries":
 - ✧ Standard FLIB and newly created FLIB are specified as synthesis option. FLIB of other basic functional unit are removed from option.

- FLIB for basic operators has the following description in header of FLIB file
 - #@KIND{BASIC_OPERATOR}
- ✧ Already set FLIB used for arithmetic macro functional unit cannot be removed from the synthesis option.
 - FLIB for arithmetic operators has the following description in header of FLIB file
 - #@KIND{ARITH_MACRO_OPERATOR}
- Select "Generate FLIB file containing all FU entries":
 - ✧ FLIB file is generated as one file under process that concludes all functional units described in FLIB of all basic operators in synthesis option.
 - ✧ FLIB of all basic operators of existing settings are deleted from synthesis option and only the new FLIB is specified as synthesis option for FLIB of basic operators.
 - ✧ FLIB of arithmetic operators of existing settings is not deleted from synthesis option.
- Select "generate FLIB file containing only additional FU entries" :
 - ✧ FLIB that has description of only functional units that lack details of FLIB of existing settings in synthesis option, is generated under process.
 - ✧ All FLIB of existing settings and all newly created FLIB are specified as synthesis option.
- Select "Generate FLIB file without reading input FLIB file(s)" :
 - ✧ Regardless of the contents of FLIB of existing settings in synthesis options, new FLIB is generated on the basis of only described contents under the process.
 - ✧ All FLIB of existing settings and all new FLIB are specified as synthesis options.
- When "arithmetic operators" is selected in "Generate library for"
 - FLIB is generated under the process as one file that concludes all functional units that are described in FLIB of all arithmetic operators in existing settings in synthesis option.
 - FLIB of all arithmetic operators of existing settings are deleted from synthesis option and only the newly created FLIB is specified as synthesis option for FLIB of arithmetic operators.
 - FLIB of basic operators of existing settings is not deleted from synthesis option.

8.4. Delay area setting of Functional unit library file (FLIB)

In FLIB, there are items to specify the delay value and area value of the functional unit in use. In these items, values are not set up when FLIB is automatically generated. Therefore, for implementing high quality behavioral synthesis, it is mandatory to either specify the value for delay and area by editing (section 8.6) FLIB or doing automatic settings by executing the delay and area settings program as explained in this section.

Setting of value for delay and area of FLIB is carried out by using FLIBgen program. This program inputs FLIB and synthesizes each functional unit specified in FLIB by using external logical synthesis tool. Then it searches for delay, area, NET count and PINPAIR count and overwrites the inputted FLIB.

Further, if function is converted into functional unit by behavioral synthesis, then _E.FCNT file that provides information related to relevant function after synthesis is generated. FLIBgen can be executed for _E.FCNT in order to set detailed information of this function. E.FCNT is generated even if process function is synthesized but if FLIBgen is executed for this _E.FCNT file it can not be opened because an error message appears.

FLIBgen usage condition

FLIBgen requires the following logical synthesis tools as external tool.

- Windows : Synplify, Xilinx ISE, Quartus II
- LINUX : Design Compiler, Synplify

FLIBgen option setting

FLIBgen option setting screen is started by selecting "Characterize Functional Unit Library (FLIB)" from context menu of FLIB, _E.FCNT in project management window.

FLIBgen execution

For executing FLIBgen, start FLIBgen option setting screen, and press "execute" button.

Generated file

If execution of FLIBgen has a normal exit, FLIB, _E.FCNT files that are to be executed are overwritten by file having delay, area, NET count, PINPAIR count specification.

Notes

- One FLIBgen option is saved for a process. Accordingly, in case there are multiple FLIB and _E.FCNT under a single process, then option setting value in option screen opened from respective FLIB, _E.FCNT will be same.

8.5. Automatic generation of Memory library/constraint file (MLIB/MCNT)

MLIB specifies the information of memory used in synthesis. MCNT specifies the constraints of memory used in synthesis. MLIB and MCNT are used together (As an exception, sometimes MCNT is used independently). MLIB, MCNT can be described manually from scratch, however, in CWB, there is a functionality to generate MLIB, MCNT template.

Automatic generation of MLIB, MCNT is carried out by executing it after providing "-Zmlib_mcnt_out" option in bdltran. By executing "-Zmlib_mcnt_out" for intermediate file generated after source code analysis, type of memory required in behavioral synthesis is

determined after intermediate file analysis, and MLIB, MCNT template is generated.

bdltran -Zmlib mcnt out option setting

Option setting screen of bdltran -Zmlib_mcnt_out can be opened by any one of the following ways:

- 1) Select "automatic generate memory library/constraint (MLIB/MCNT)" from context menu of process in project management window.
- 2) Select "automatic generate memory library/constraint (MLIB/MCNT)" from drop down menu of "synthesize" in the state where process to be executed is selected in project management window.

bdltran -Zmlib mcnt out execution

Open option setting screen and click "execute" button to execute bdltran -Zmlib_mcnt_out.

Generated file

If execution of bdltran -Zmlib_mcnt_out has a normal exit, new MLIB and MCNT will be created in process constraint folder. Generated MLIB and MCNT is automatically set as argument of behavioral synthesis option.

Notes

- Appropriate value is not specified for delay of memory in automatically generated MLIB and MCNT. It is required to edit MLIB and MCNT by referring section 8.6.
- If scheduling mode is changed in option setting screen, scheduling mode of behavioral synthesis and automatic generation of functional unit library/constraint is changed after conjunction.
- If option related to usage of memory is changed in option setting screen, applicable option of behavioral synthesis is changed after conjunction.

8.6. Editing of constraint file

Since each constraint file is a text file, generally, it can be edited in text editor however, if CWB-gui inbuilt constraint editor is used, constraint file can be edited without recognizing the precise formatting of each constraint file.

Start constraint file editing screen

Constraint file editing screen can be started by any of the following.

- 1) Double click the constraint file in project management window (however, it is necessary to specify FLIB, MLIB, PLMT, which act as behavioral synthesis option, for FCNT, MCNT, PREL).
- 2) Open new screen, select constraint file and set extension for file name same as the constraint file to be created.
- 3) When behavioral synthesis (bdltran) option screen is opened (section 9),
 - **LMT:** When existing LMT is set for "functional unit constraint file" of "functional unit library" tab, press "edit" button.
 - **MLMT:** When existing MLMT is set for "memory constraint file" of "memory

library" tab, or press "new".

- **PLMT:** When exiting PLMT is set for "port constraint file" of "memory library" tab, either press "edit" button or press "new".
- **PREL:** When existing PREL is set for "port related file" of "memory library" tab, either press "edit" button or press "new".
- **BLIB:** When existing BLIB is set for "basic library file" of "memory library" tab, either press "edit" button or press "new".
- **FLIB:** When existing FLIB is set for "functional unit library file" of "functional unit library" tab, either press "edit" button or press "new".
- **FCNT:** When existing FCNT is set for "functional unit count file" of "functional unit library" tab, either press "edit" button, or press "new".
- **MLIB:** : When existing MLIB is set for "memory library file" of "memory library" tab, press "edit" button or press "new".
- **MCNT:** : When existing MCNT is set for "memory count file" of "memory library" tab, either press "edit" button or press "new".

When constraint file editing screen is opened in the state where it has becomes modeless, synthesis can be carried out in project management window or synthesis support window while changing the contents of constraint file.



Furthermore, multiple pages can be opened in constraint file editing screen. However, there are following limitations:

- In constraint file editing screen 5 pages can be opened at the same time.
- In constraint file editing screen opened with the method of above mentioned point 3), other operations can not be done while editing because it becomes a modal dialogue
- When constraint file editing screen is opened with method of above mentioned point 1), same constraint file can not be opened with the method of point 3).
- When constraint file editing screen is opened with method of above mentioned point 1), if same constraint file is opened with same method again, the already open constraint file editing screen pops up. However, sometimes it does not pop up depending on environments like X server.
- When FLIB is opened in constraint file editing screen, FCNT of same process cannot be opened. Similarly, when FCNT is opened FLIB of same process cannot be opened. There is similar limitation for MLIB and MCNT.
- FLIB in which "@PROCESS" entry is described cannot be opened in FLIB file editing screen.

8.7. Constraint file icon display

Constraint file, depending upon its status, shows different icons in the project management window.

-  : Constraint file icon

-  : It is present inside work folder of process having constraint files and in case it is entered as synthesis option of relevant process, "O" will be marked on the left top of constraint file icon.
-  : In _E.FCNT file (refer section 8.4), on which delay and area setting (FLIBgenare executed, "F" sign will appear on top-right of constraint file icon. This sign is removed when _E.FCNT file is overwritten. Please be careful while synthesizing sub functions as new _E.FCNT is created. (Refer section 9).

Sometimes both signs ("O", "F") appear on icon.

9. Behavioral synthesis

When source file analysis is carried out and constraint files are created, behavioral synthesis can be carried out. Behavioral synthesis is executed through bdltran program.

9.1. Option settings of bdltran

Options of bdltran can be set in the following way.

bdltran option setting

bdltran option setting screen can be started by any one of the following methods.

- 1) Select "synthesize options" from context menu of process in project management window
- 2) When process is selected in project management window, select "synthesize options" from drop-down menu "synthesize".
- 3) When process is selected in project management window, select "synthesize options" from operation toolbar.
- 4) When process is selected in project management window, select "Synthesis options" of command palette.

Search option screen settings from option name

Settings for bdltran option setting screen, supporting the options given in Cyber Reference Manual etc, can be searched through help. Usage method of help is as follows.

- 1) Select "behavioral synthesis" from drop down menu "help"

In this Help screen, when options displayed in blue color are clicked, setting method of that option appears in the lower part of Help screen.

Things to be noted during option settings



Sometimes few of the bdltran options are automatically specified.

- 1) In case constraints are automatically generated (section 8.2, section 8.3, section 8.5):
 - When constraint file is automatically generated for a process, the generated processes are automatically set in bdltran option. When a process that becomes sub hierarchy or function that is converted in functional unit is synthesized, sometimes constraint file is added and updated in options because information of sub hierarchy process and function is passed to call based process and function through _E.FCNT file (Refer section 9.3).
- 2) In case function is converted into functional unit during BDL source file analysis (section 7.1):
 - To convert function in to functional unit, "-Ztop=function name" option is set as default option . Regarding the details of this option, please refer Cyber Reference Manual.
- 3) In case clock cycle and standard FLIB, standard BLIB is set up in properties of project during generation of intermediate project and RTL project:

- These values are reflected in behavioral synthesis option of process under relevant project.
- 4) If .cyber file is created under home directory and behavioral synthesis option is mentioned, these options will be specified as next priority during behavioral synthesis. At this time, options such as deletion cannot be specified from CWB.

Change of process icon by option settings

Figure of process icon will change in the following way by changing the value of scheduling mode:

- When "manual scheduling" is set : "M" is displayed in the bottom right corner of the icon like .
- When "automatic pipeline scheduling" is set: "P" is displayed in the bottom right corner of the icon like .

9.2. Execution of bdltran

Bdltran can be executed by either of the following procedure

Bdltran execution

bdltran can be executed by any of the following steps.

- 1) Select "synthesize" from context menu of process of project management window
- 2) When process is selected in project management window, select "synthesize" from drop down menu "synthesize"
- 3) When process is selected in project management window, select "synthesize" from operation toolbar.
- 4) Open bdltran option setting screen and press "execute" button.
- 5) When process is selected in project management window, select "Synthesize/Generate RTL" on command palette.

Error file display of bdltran

Error file at the time of bdltran execution can be displayed by selecting "synthesize" in "open error file" from context menu in the state where process is selected in project management window.

In case of any errors, the applicable row of source file can be highlighted by selecting the row number in error file.

Generated file

Files generated by bdltran are mainly stored under Synthesis Analysis folder of process. Various files are output, however out of these also especially, structural level internal format file (_E.IFF) and cycle accuracy level internal format file (_C.IFF) are important intermediate files because these will be required in further processes such as RTL generation or simulation model generation. Please refer Figure 1-1 for the flow.

Execution constraints

During bdltran execution, in case analysis of the process to be used is in analysis pending status, then bdltran is executed after it is automatically analyzed.

Moreover, in case following process is in pending behavioral synthesis state, then behavioral synthesis is automatically executed similar to the process for which behavioral synthesis will be done.

- Functional unit conversion: Synthesis of applicable process is executed before synthesis of process to be used without fail because it has an impact (_E.FCNT file is generated and passed) on the synthesis of that process.
- Sub hierarchy described by defmod: In this case, synthesis execution sequence is not controlled because synthesis of applicable process does not impact the synthesis of process to be used.

GUI automatically searches these sub hierarchies, either from the project in which there is a process for behavioral synthesis or from its subproject in project tree of project management window.

In case sub hierarchy described in defmod calls the conversion of function into functional unit, synthesis execution sequence of these processes take as standard the policy explained in section 9.3 and synthesis of converting function into functional unit is implemented first.

When bdltran has a normal exit, RTL generation is executed automatically (Chapter 10). Further, status of applicable process becomes synthesized. As a result, if bdltran is executed, it is skipped. In the following operations, synthesized status is analyzed and then status becomes synthesis pending.

- In bdltran option setting screen, press OK button after running the options.
- Reanalysis is done for source file to be used.

9.3. Handling of _E.FCNT file generated by synthesis

If behavioral synthesis is implemented, functional unit count file (_E.FCNT) is generated. Details (area, port configuration etc.) of the synthesized function or process are described in the _E.FCNT file. At the time of behavioral synthesis of function and process of parent hierarchy, estimation of area containing sub hierarchy and matching validation of port configuration can be done by providing this _E.FCNT file as option.

CWB-gui controls this _E.FCNT file with the following policy:

- In case same name file is not in work directory of call based parent process and functions in instance tree: Mandatory copying of the file is done and it is provided as synthesis option. As per the settings mentioned later, in case it is set as do not copy then copying is not done.
- In case there is same name file in work directory of call based parent process and functions in instance tree:
 - When contents of the file are completely same: Nothing is done.
 - When contents are different: A dialogue box appears that asks whether to overwrite or not and the process of the selected option is followed. If overwrite is

selected, the file is automatically set as synthesis option after it gets overwritten. In case operation such as edit or delay area setting (FLIBgen) is executed for overwriting functional unit count file, please be careful as execution result and editing result will be lost due to overwriting. Furthermore, "F" sign is added to the icon of _E.FCNT file that executed delay area settings, so please identify it with this sign while selecting (Refer 8.7).

- If you select "Yes to ALL", synthesis is continued for series of synthesis without displaying the confirmation dialog box.

Moreover, the setting can be done for not displaying this dialogue box from the tool option settings. In this case the behavior depends on the settings of the tool option, and one can select - always copy _E.FCNT file or always do not copy _E.FCNT file or display dialogue box for every synthesis. Please refer section 17.1.6 for details.

10. RTL generation

Synthesis result can be output as Verilog and VHDL by carrying out RTL generation for process where behavioral synthesis is normally completed.

RTL output is Verilog by default. This can be changed by "RTL selection" combo box of operation tool bar.

10.1. Verilog generation

The program that generates synthesis result as Verilog is veriloggen.

veriloggen option setting

veriloggen option setting screen can be started by any of the following, when "RTL selection" combo box of Operations toolbar is set to "Verilog",

- 1) Select "RTL generation options" from context menu of process in project management window.
- 2) Select "RTL generation options" from drop down menu "synthesize", when process is selected in project management window.
- 3) When process is selected in project management window, select "Generate RTL options" on command palette.

Veriloggen execution

When "RTL selection" combo box of operation tool bar is set to "Verilog", veriloggen can be executed by any of the following methods.

- 1) Select "Generate RTL" from context menu of process in project management window
- 2) When process is selected in project management window, Select "Generate RTL" from drop down menu "synthesize"
- 3) When process is selected in project management window, Select "Generate RTL" from operation tool bar
- 4) Start veriloggen option setting screen, and press "execute" button.

Generated file

The file generated by veriloggen is stored under synthesis RTL and script folder of process.

Execution control

In case the status of analysis and synthesis of the process to be used is pending at the time of veriloggen execution, veriloggen is executed after these processes are automatically executed. Also, in case the status of behavioral synthesis and RTL generation for sub hierarchy mentioned in defmod or functional unit conversion is pending, behavioral synthesis and RTL generation will be automatically executed similar to the process for which behavioral synthesis is to be done. GUI automatically searches the sub hierarchies, either from the project in which there is a process for RTL generation or from its sub

project in project tree of project management window.

If there is a normal finish of veriloggen, the status of applicable process becomes RTL generated. As a result, if veriloggen is executed again, it is skipped. The RTL generated status is analyzed in the following operations and the status becomes RTL generation pending.

- In veriloggen option setting screen, press OK after running the options.
- Reanalyze the source file to be used
- Re-synthesize the process to be used

10.2.VHDL generation

vhdlgen is a program to generate synthesis result as VHDL.

vhdlgen option setting

Vhdlgen option setting screen can be started by any of the following when "RTL selection" combo box of Operations toolbar is set to "VHDL".

- 1) Select "RTL generation options" from context menu of process in project management window
- 2) When process is selected in project management window, select "RTL generation options" from drop down menu "synthesize"
- 3) When process is selected in project management window, select "Generate RTL options" on command palette.

vhdlgen execution

When "RTL selection" combo box of operation tool bar is set to "VHDL", vhdlgen can be executed by any of the following methods.

- 1) Select "Generate RTL" from context menu of process in project management window.
- 2) When process is selected in project management window, Select "Generate RTL" from drop-down menu "synthesize".
- 3) When process is selected in project management window, Select "Generate RTL" from operation toolbar.
- 4) After opening the vhdlgen option setting screen, press "execute" button.

Generated file

File generated by vhdlgen, is stored under synthesis RTL and script folder of process.

Execution control

In case the status of analysis and synthesis of the process to be used is pending at the time of vhdlgen execution, vhdlgen is executed after these processes are automatically executed. Moreover, in case the status of behavioral synthesis and RTL generation for sub hierarchy mentioned in defmod or functional unit conversion is pending, behavioral synthesis and RTL generation is automatically executed similar to the process for which

behavioral synthesis is to be done. GUI automatically searches the sub hierarchies, either from the project in which there is a process for RTL generation or from its sub project in project tree of project management window.

If there is a normal finish of vhdlgen, the status of applicable process becomes RTL generated. As a result, if vhdlgen is executed again, it is skipped. The RTL generated status is analyzed in the following operations and the status becomes RTL generation pending.

- In vhdlgen option setting screen, press OK after running the options.
- Reanalyze the source file to be used
- Re-synthesize the process to be used

10.3.Generation of logical synthesis script

LSscrgen program outputs the template (file name:_E.dcsr) of logical synthesis script from file group after behavioral synthesis and RTL generation execution. For details, please refer Cyber reference manual.

LSscrgen option setting

LSscrgen option setting screen is started by selecting "logical synthesis options/Generate script" from context menu of process of project management window after behavioral synthesis and RTL generation execution.

It can also be started by clicking "Logical synthesis options/Generate script" on the command palatte after selecting the process.

LSscrgen execution

When LSscrgen option setting screen is opened, LSscrgen can be executed by pressing the execute button.

Generated file

File generated by LSscrgen, is stored under synthesis RTL and script folder of process.

Execution control

At the time of LSscrgen execution, even if the status of analysis, synthesis and RTL generation is pending for the process to be used, the processes are not executed automatically.

10.4.Generated Verilog/VHDL export

Verilog/VHDL generated by veriloggen/vhdlgen is kept under work directory of process to be executed.

These files can be obtained by directly accessing the directory having files. However, if RTL export functionality of CWB-gui is used, RTL file can be extracted much easily.

Starting method of RTL export

RTL export can be executed by any of the following methods.

- 1) Select "Export RTL" from context menu of workspace or project in project management window
- 2) Select "Export RTL" from context menu of process in project management window
- 3) Select "Export file" from context menu of RTL generated by RTL generation in project management window.
- 4) When process is selected in project management window, select "Export generated RTL files" on command palette.

If "create new directory" checkbox in the lowermost part of "Export RTL" screen is checked, then in case of workspace or project and process for which RTL export is to be executed, directories with name of workspace or project and process is created under specified export destination directory. Under that directory, all RTL files are exported directly below the directory to be executed.

At this time, in case multiple RTLs to be exported have same file-name, then file name from second RTL onwards is generated after changing the file name (index is provided in file name). The modification information of this file name is generated in CWB-gui console.

If "create new directory" checkbox in the lowermost part of "Export RTL" screen is unchecked, all files are generated directly under specified directory.

11. Supporting functionality for Synthesis

There are various options in behavioral synthesis program bdltran. In order to generate the circuit by behavioral synthesis that has the speed and area fulfilling the specifications, sometimes architecture search should be carried out by trying various combinations of these options and group of constraint files,

In CWB-gui, there is synthesis support window that supports this architecture search for unit process. This section provides explanation about synthesis support window.

11.1. Synthesis support window overview

Figure 11-1 shows the appearance of synthesis support window.

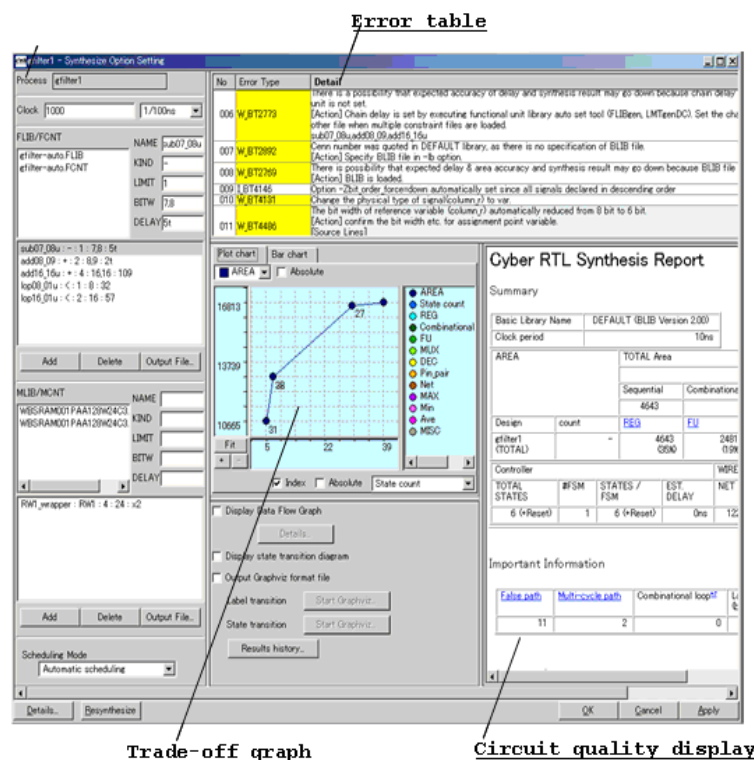


Figure 11-1 Synthesis support window

Synthesis support window consists of the following components.

- Graph (section 11.3): Displays the graphical representation of result of executed behavioral synthesis.
- Error table (section 11.4): Table of errors generated in any particular behavioral synthesis trial.

- Circuit quality display (section 11.5): Displays circuit quality of any particular behavioral synthesis trial result.

Start up methods

Synthesis support window can be started by any of the following methods.

- 1) Select "synthesize window" in "synthesis result display" from context menu of process in project management window
- 2) When process is selected in project management window, select "synthesize window" in "synthesis result display" from drop down menu "synthesize"
- 3) When process is selected in project management window, select "synthesize window" from operation tool bar

11.2. Synthesis execution

There are following two buttons in the bottom left of synthesis support window.

- Details: Starts the bdltran option setting screen.
- Resynthesize: carries out synthesis by using specified behavioral synthesis options.

If this "Details" button is pressed and bdltran option setting screen is opened, then all the options of bdltran can be set, however, values can be specified for display settings such as data flow graph and state change diagram, in synthesis support window even if the bdltran option setting screen is not opened.

Option settings of bdltran can be implemented from bdltran option settings screen that is opened from drop down menu, toolbar and project management window and it can also be implemented from bdltran option settings screen opened from Details button mentioned previously. Saved values are same in these option screens.

11.3. Graph

When behavioral synthesis is executed, its result is displayed in graphical form. Quality comparison can be done for generated circuits by using this graph.

By changing the tab provided on the top of graph, 3 types of graphs can be displayed.

Graph can be opened by any one of the following methods.

- 1) Select "Graph" in "synthesis result display" from context menu of process in project management window
- 2) When process is selected in project management window, select "Graph" in "synthesis result display" from "synthesize" drop down menu.

11.3.1. Plot (trade-off graph)

Trade-off graph can be displayed by selecting the tab "plot chart". Figure 11-2 shows the

example of trade-off graph.

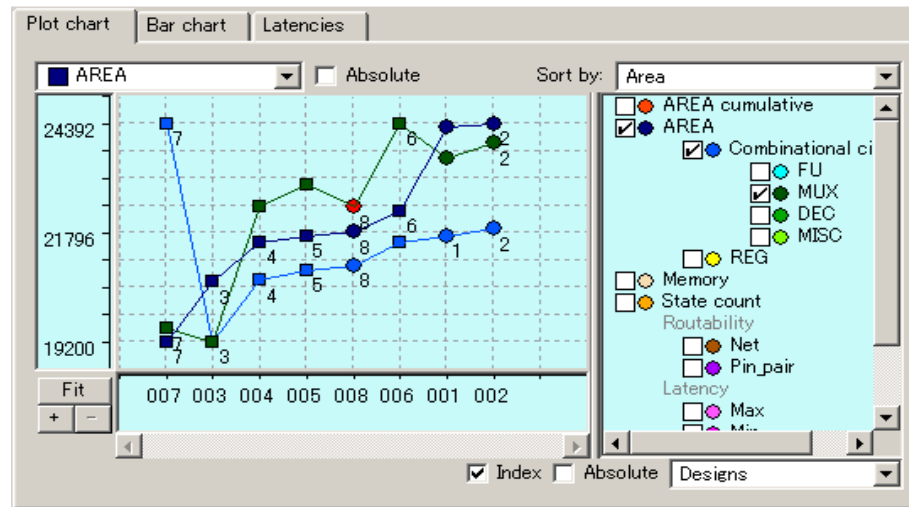


Figure 11-2 Trade-off graph

When behavioral synthesis is implemented, result gets plotted in trade-off graph. Better synthesis parameters can be searched through quantitative comparison of synthesis result.

The following operations can be performed for trade-off graph.

- By clicking the points plotted in graph, result of corresponding behavioral synthesis trial will be displayed in the form of error table and circuit quality display.
- In case two or more points are overlapping, or exist in very close proximity, then by clicking the point, a window to select the synthesis result will appear (Figure 11-3). By clicking the row inside the window, result of corresponding synthesis trial will be displayed in the form of error table and circuit quality display. The horizontal axis details are as follows. Please refer the following points for operations that can be performed in this window.
 - ID : Serial number of behavioral synthesis trial
 - Time : Time duration that entered history of behavioral synthesis
 - Third column (AREA): Index for vertical axis of displayed trade-off graph
 - Fourth column (Net): Index for horizontal axis of displayed trade-off graph
 - Comments: Comments for synthesis result

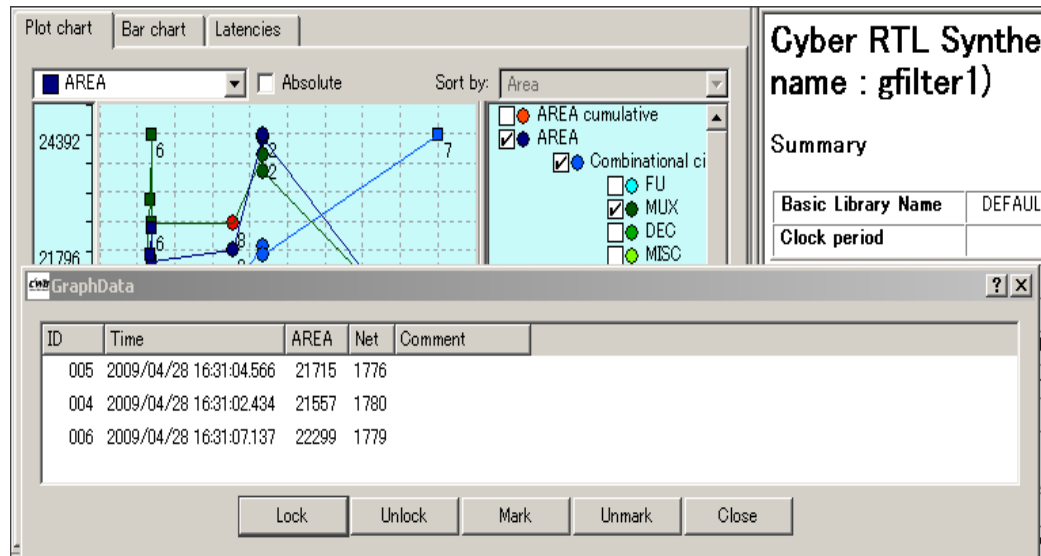


Figure 11-3 Synthesis result selection screen

- By right clicking the point plotted in graph, a window to select synthesis result will appear (Figure 11-4). Horizontal axis details are same as Figure 11-3.
 - If row is clicked in this table, "display" button and "delete" button, at the bottom of the window can be selected. At this time,
 - ✧ When "Lock" button is pressed: By locking the relevant synthesis result, it will not be deleted when all synthesis result in project window is deleted in one go (section 11.7.2).
 - ✧ When "Unlock" button is pressed: Removes the locked state
 - ✧ When "mark" button is pressed: Adds a mark to the graph of relevant synthesis history.
 - ✧ When "Unmark" button is pressed: Removes the mark.
 - ✧ When "Restore results" button is pressed: Environment at the time of execution of relevant synthesis can be restored in project management window. Please refer section 11.7 for details.
 - ✧ When "display" button is pressed: result of corresponding behavioral synthesis trial will be displayed in the form of error table and circuit quality display.
 - ✧ When "delete" button is pressed: result of corresponding behavioral synthesis trial is deleted.
 - By double clicking the Comments column, comments can be inserted about each synthesis result.

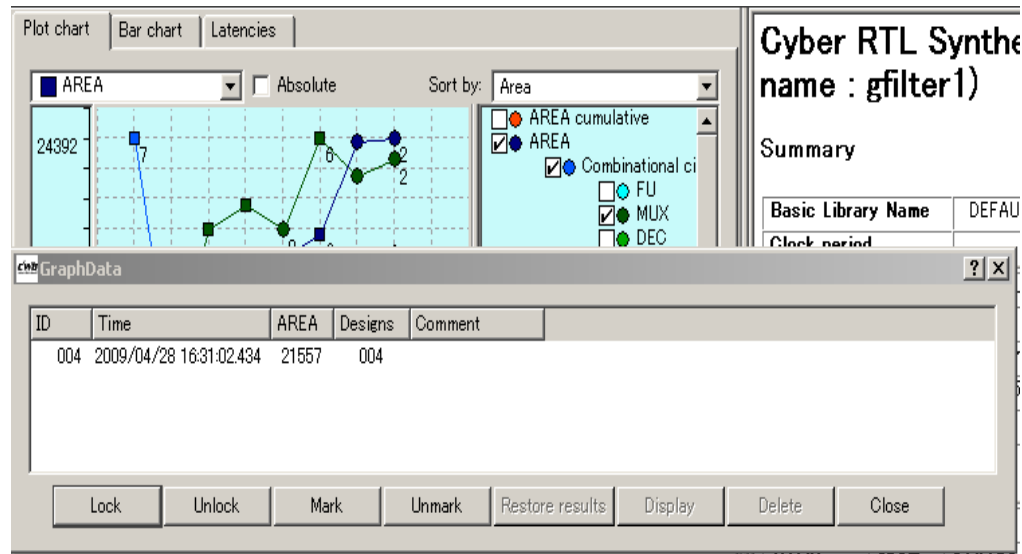


Figure 11-4 Synthesis result selection screen (right click)

- The following display changes are possible by operating each check box.
 - Check box at the top of the screen "Absolute": If it is check marked, vertical axis display will be displayed for absolute value with minimum value as zero.
 - Check box at bottom of screen "Absolute": If it is check marked, horizontal axis display will be displayed for absolute value with minimum value as zero.
 - "Index" check box: If it is check marked, synthesis ID will be displayed for each plotted point. In case relevant point shows the multiple synthesis results, ID of biggest value is displayed as index.
- Explanatory Notes (Key) have been displayed on the right of plot graph that represent the color corresponding to the Index. Explanatory Note (key) can be hidden by dragging the Frame that is located on the left-side of this Explanatory Note (Key) display area.
 - Explanatory Note (key) can be hidden by dragging the Frame that is located on the left-side of this Explanatory Note (Key) display area to right side of the screen.
 - Index that is displayed as graph can be selected by checking every checkbox of explanatory notes.
- By operating combo box, scale index of each axis can be changed.
 - Combo box at the bottom of the screen: horizontal axis index can be selected. Any index can be selected.
 - Combo box at the top of screen: vertical axis index can be selected. Index selected in checkbox of explanatory notes is selected.
 - Combo box at top right side of the screen: In case combo box in the lower part of the screen is "Synthesis history", index that aligns synthesis history in x axis is specified.
- Screen can be expanded as well as reduced.
 - Expand: By pressing "+" button at the bottom left of screen and specifying

(dragging and dropping both end points that forms rectangular shape) the area in graph, specified area can be displayed in the expanded form.

- Reduce: By pressing "-" button at the bottom left of screen, the screen will be reduced to actual shape, before last expansion.
- Scroll: At the time of expanded display, by left clicking the graph background, specified coordinates will be focused and scrolling can be done. Scrolling can also be done by arrow keys.
- Full screen display: By pressing the "Fit" button at the bottom left of screen, display will return to initial state.

11.3.2. Bar Graph

By selecting the "Bar chart" tab, bar graph is displayed. Figure 11-5 shows the example.

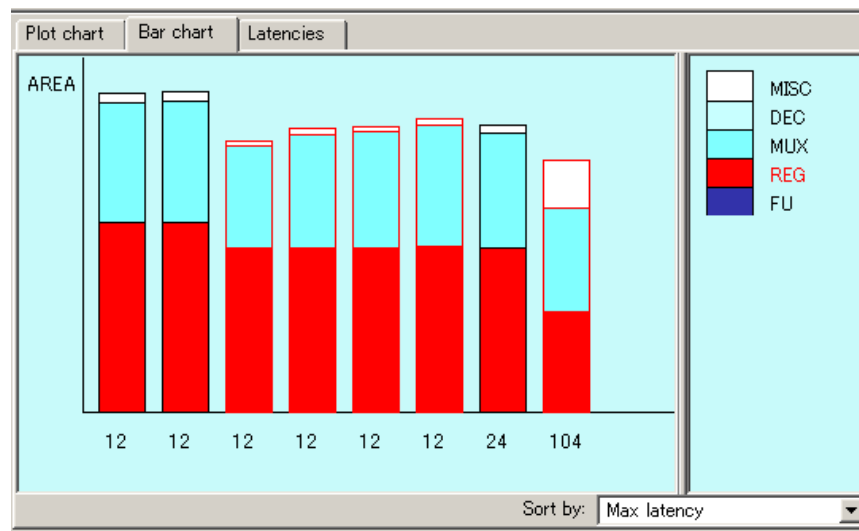


Figure 11-5 Bar graph

When behavioral synthesis is implemented, result is plotted in bar graph. Better synthesis parameters can be searched by carrying out the quantitative comparison of synthesis result, .

In bar graph, vertical axis represents area, horizontal axis represents maximum cycle count after behavioral synthesis, and both are fixed. Bars are arranged from left side with respect to maximum cycle count after behavioral synthesis. Each bar is divided into five different colors, representing different area, starting from bottom i.e. functional unit area, register area, multiplexer area, multiplexer area with decoder, and other area. Detailed value of each area is given in circuit quality that is displayed by clicking the bar, as respective FU, REG, MUX, DEC, MISC. (Figure 11-6)

Cyber RTL Synthesis Report

Summary

Basic Library Name	DEFAULT (BLIB Version 2.00)						
Clock period	10ns						
AREA		TOTAL Area					
		12330					
		Sequential		Combinational			
		4440		7890			
Design	count	REG	FU	MUX	DEC	MISC	
#filter1 (TOTAL)	-	4440 (36%)	1493 (12%)	5743 (46%)	0 (0%)	654 (5%)	

Controller				WIRE		PORT			
TOTAL STATES	#FSM	STATES / FSM	EST. DELAY	NET	PIN PAIR	TOTAL	IN	OUT	IN / OUT
14 (+Reset)	1	14 (+Reset)	0ns	1275	2272	82	45	37	0

Figure 11-6 Circuit Quality

The following operations can be performed by bar graph.

- By clicking the bar, result of corresponding behavioral synthesis trial is displayed in the form of error table and circuit quality display.
- By right clicking the bar plotted in graph, a window (Figure 11-7) to select synthesis result appears. By clicking the row in table, "Delete" and "Restore results" buttons at the bottom of window become selectable. In this state, by pressing "delete" button, result of corresponding behavioral synthesis is deleted. By pressing "Restore results" button, environment of applicable synthesis execution can be restored in project management window. Please refer section 11.3.1 for "Lock", "Unlock", "Mark" and "Unmark". For details please refer section 11.7. Horizontal axis details are as follows. Please refer Plot graph (section 11.3.1) for operations that can be performed in this window.
 - ID : Serial number of behavioral synthesis trial
 - History registration time : Time registered for the history of behavioral synthesis
 - Third column (AREA): vertical axis index of trade-off graph in plot graph display
 - Fourth column(State count): horizontal axis index of trade-off graph in plot graph display
 - Comments: Comments for synthesis result

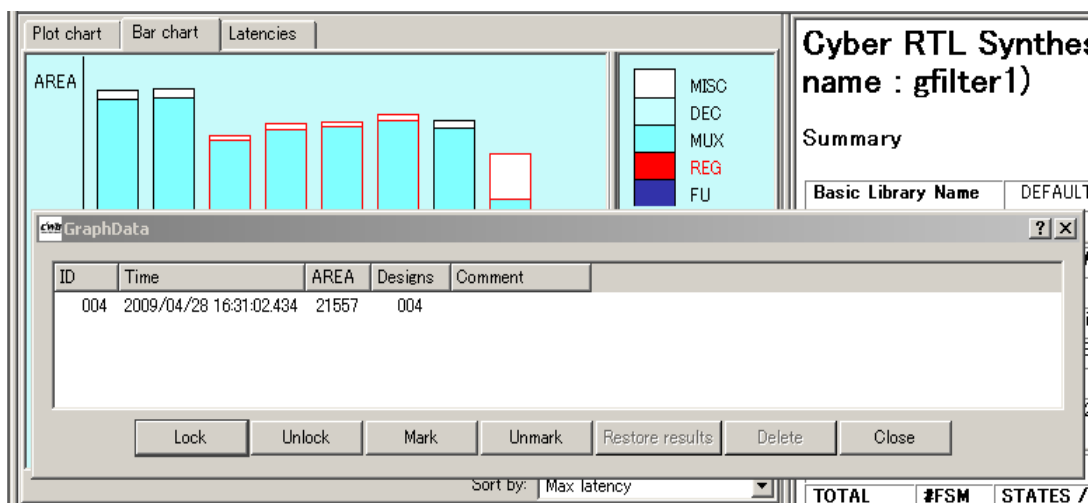


Figure 11-7 Synthesis result selection screen (right click)

- Explanatory Notes (Key) have been displayed on the right of plot graph that represents the color corresponding to the Index. Explanatory Note (key) can be hidden by holding the Frame that is located on the left side and dragging it to right side of the screen.
- Combo box at bottom right side of screen: Specifies the index that aligns synthesis history.
- Screen to select display data will appear by right clicking the background other than plotted bars in graph (Figure 11-8). The index selected in this screen, will be displayed as the lowest area of bar graph. Same effect is achieved by clicking explanatory notes also.

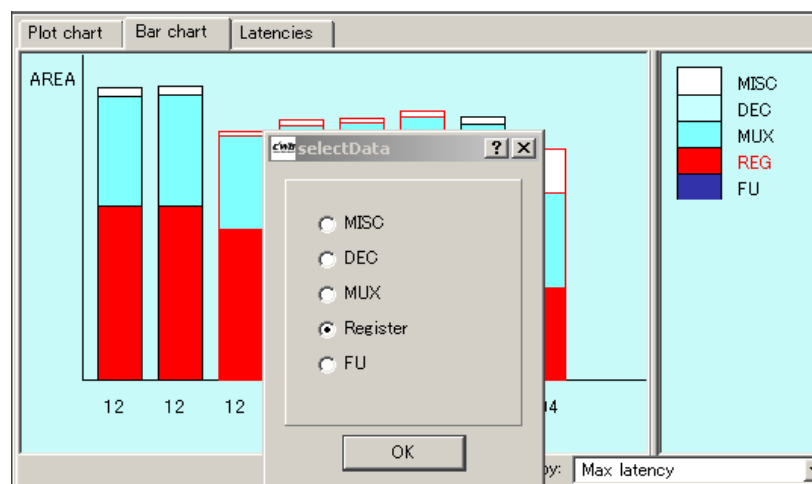


Figure 11-8 Display data select screen (right click background)

11.3.3. Loop latency display

When "Latencies" tab is selected, Latency table is displayed in which vertical axis is synthesis history and horizontal axis is Latency of loop. Example is shown in Figure 11-9.

Plot chart Bar chart Latencies			
Design	Loop		Total Latency
	L1	L2	
001	$[5, 4] * N1$	–	Reset + L1
002	$1 * N1$	$(3 + L1) * N2$	Reset + L2
003	$[5, 5] * N1$	–	Reset + L1

Figure 11-9 Latency

When behavioral synthesis is implemented, result is added to the table and displayed. The following operations can be performed in the table:

- By clicking the Synthesis history label, result of corresponding behavioral synthesis trial is displayed in the form of error table and circuit quality display.
- By right clicking the Synthesis history label, a window to select synthesis result appears. By clicking the row in this table, "Delete" and "Restore results" buttons at the bottom of the window become selectable. In this state, by pressing "delete" button, result of corresponding behavioral synthesis is deleted. By pressing "Restore results" button, environment of applicable synthesis execution can be restored in project management window. Please refer section 11.3.1 for "Lock", "Unlock", "Mark" and "Unmark". For details please refer section 11.7. Horizontal axis details are as following. Please refer Plot graph (section 11.3.1) for operations that can be performed in this window.
 - ID : Serial number of behavioral synthesis trial
 - History registration time : Time registered for the history of behavioral synthesis
 - Third column (AREA): vertical axis index of trade-off graph in plot graph display
 - Fourth column(State count): horizontal axis index of trade-off graph in plot graph display
 - Comments: Comments for synthesis result

When synthesis is implemented, different loop is added and deleted in the description to be entered. Moreover, even if input description is same it can be considered that multiple loops are merged as per specifications of options and attributes. In this kind of case, please be careful because there can be conditions such as that in each synthesis history count of loops is different, and loops displayed in same label show a different loop for every synthesis history.

executed and behavioral synthesis result is selected in graph, displayed contents will be changed.

11.6. Result Display

Synthesis result can be viewed in table form. Result list is displayed by pressing "Results history" button in synthesis support window.

Result list consists of "Hierarchy history" tab and a tab having process name. In case process to be synthesized is held as sub-hierarchy, tabs having name of all processes will appear.

Here details of all tabs will be explained.

Result history can be opened by any one of the following methods:

- 1) Select "Results history" in "Synthesis results display" from context menu of process of project management window
- 2) Select process in project management window and then select "Results history" in "Synthesis results display" from "synthesize" drop-down menu

11.6.1. Hierarchical history tab

Hierarchical history tab shows the detailed information including hierarchical information of process to be synthesized. Figure 11-11 shows the hierarchical history tab. The structure of this screen is as follows.

- First row: detailed information of process to be synthesized is displayed.
- Second row onwards: All processes that are kept as instance in the synthesis process and its sub hierarchy are enumerated. Total number of instance processes among processes to be synthesized are displayed in "instance count" column.
- SUM : From "Area" column to "Net" column, numeric characters will be displayed. The contents of these numeric characters shows the result (area of entire hierarchy with process to be synthesized at the top) calculated as follows by taking example of "Area".
 - $1^{\text{st}} \text{ row process} * 1^{\text{st}} \text{ row process instance count} + \dots + \text{nth row process} * \text{nth row process instance count}.$

Process	Instance	AREA	State	c	FU	REG	MUX	DEC	MISC	Memory	Pin_pail	Net	MAX	Min	Ave
gfilter1	1	155(100.	5		2606	14092	10716	0	541	-	3908	2088	5	5	5.00
SUM		27955	5		2606	14092	10716	0	541	0	3908	2088			

Figure 11-11 Result List Hierarchical History Tab

11.6.2. Process name tab

Process name tab shows the information of each process unit that does not include the hierarchical information. This tab is shown in Figure 11-12.

Each row of this table shows the trial of behavioral synthesis. “number” of Leftmost column indicates the serial number of behavioral synthesis trial; same as ID of graph (section 11.3).Remaining rows indicate the detailed result of relevant synthesis trial.

When row from this list is selected, and “restore results” button is pressed, then parameters and constraint files etc of relevant behavioral synthesis are reflected again in process option. For details, refer section 11.7

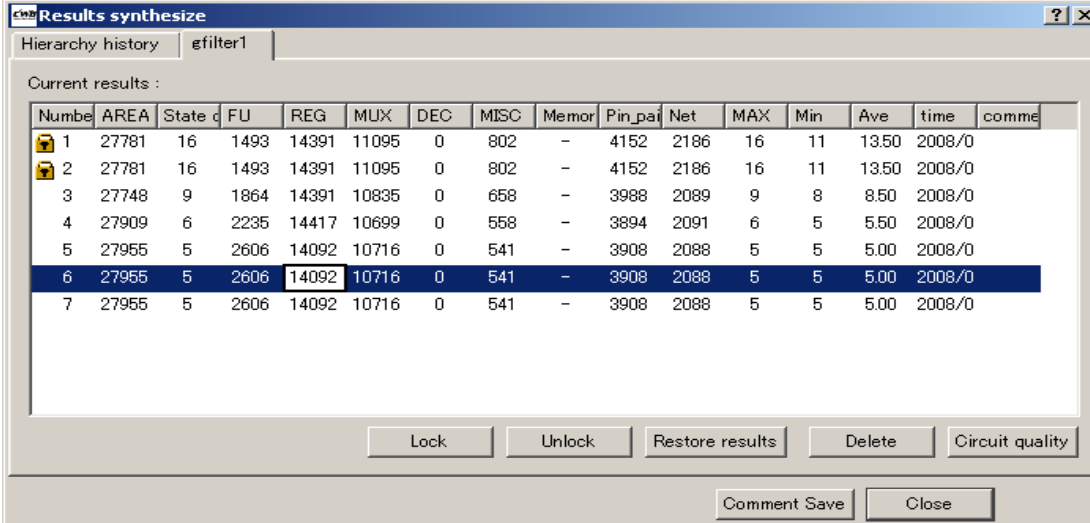
By pressing the “delete” button, relevant behavioral synthesis trial will get deleted.

By pressing “Circuit quality” button, result of relevant behavioral synthesis trial will be displayed as circuit quality in a separate window.

When “Lock” button is pressed, then while deleting (section 11.7.2) all synthesis history in operations of project window the relevant synthesis result will not be deleted by getting locked.

When “Unlock” button is pressed, the locked state is removed.

Any comments can be entered by double clicking comment of each row. Comments can be saved by pressing “Save comments” button.



Results synthesize

Hierarchy history gfilter1

Current results :

Number	AREA	State	FU	REG	MUX	DEC	MISC	Memor	Pin_pal	Net	MAX	Min	Ave	time	comment
1	27781	16	1493	14391	11095	0	802	-	4152	2186	16	11	13.50	2008/0	
2	27781	16	1493	14391	11095	0	802	-	4152	2186	16	11	13.50	2008/0	
3	27748	9	1864	14391	10835	0	658	-	3988	2089	9	8	8.50	2008/0	
4	27909	6	2235	14417	10699	0	558	-	3894	2091	6	5	5.50	2008/0	
5	27955	5	2606	14092	10716	0	541	-	3908	2088	5	5	5.00	2008/0	
6	27955	5	2606	14092	10716	0	541	-	3908	2088	5	5	5.00	2008/0	
7	27955	5	2606	14092	10716	0	541	-	3908	2088	5	5	5.00	2008/0	

Lock Unlock Restore results Delete Circuit quality

Comment Save Close

Figure 11-12 Result List Process Tab

11.7. Behavior of Save history and Restore results

This section explains about the management of history.

11.7.1. Save history

In case "Save environment of every synthesis" is enabled in "Synthesize" tab of window displayed from "Options" of "Tools" drop-down menu, then synthesis result is definitely saved as history when behavioral synthesis processing is implemented.

In case "Save environment of every synthesis" is disabled, the result is not automatically saved as history when behavioral synthesis processing is implemented. The history can be manually saved by executing the following operations. Moreover, these operations are effective only when "Save environment of every synthesis" is disabled.

- When process icon to be used is selected, press "Save synthesis results" button of Operations toolbar
- When process icon to be used is selected, elect "Save synthesis results" from context menu
- Open Synthesis support window for process to be used and then press "Save synthesis results" button

When history is saved, history directory is generated under the directory corresponding to process to be used. Under that directory folders ("001", "002") with names of synthesis history IDs and management file history.data is generated. Synthesis history is stored in the folder corresponding to each ID.

11.7.2. Delete history

Deletion of history can be done by pressing Delete button in Results list as well as in the Window that can be displayed from each graph of synthesis support window.

The following procedure is effective for deleting the history collectively:

- When process icon is selected, and "Delete synthesis history" is selected from context menu: all history that has not been locked is deleted in synthesis history of process to be used.
- When project icon is selected and "Delete synthesis history" is selected from context menu: all history that has not been locked is deleted in synthesis history of project to be used and the sub projects of that project.
- When workspace icon is selected and "Delete synthesis history" is selected from context menu: all history that has not been locked is deleted in synthesis history of all processes existing in workspace to be used

11.7.3. Restore results

History can be restored by pressing Restore results button in Results list as well as in the Window that can be displayed from each graph of synthesis support window.

Behavior of Restore results is explained as following:

In case source file is not restored

When "No" is selected in the dialogue box that appear at the time of Restore results, only synthesis environment is restored without restoring the source files. Actually behavior of restored files and behavior after restoring is different depending on the tool option settings.

- In case "Save synthesis results (IFF file)" is disabled in "Synthesize" tab of window displayed in "Options" of "Tools" drop-down menu, when history to be restored is saved:
 - Synthesis options and constraint files are restored.
 - Synthesis result of relevant history is not restored.
 - Even if synthesis is executed after Restore results, synthesis is not skipped.
- In case "Save synthesis results (IFF file)" is enabled in "Synthesize" tab of window displayed in "Options" of "Tools" drop-down menu, when history to be restored is saved:
 - Synthesis options and constraint files are restored.
 - Synthesis result of relevant history is also restored.
 - When synthesis is executed after Restore results, synthesis is not skipped.

When result is saved at the time of saving history, it saves the effort of resynthesizing at the time of restoring but this consumes a lot of disk space.

In case source file is restored

When "Yes" is selected in the dialogue box that appear at the time of Restore results, synthesis environment is restored along with source files. Actually restored files and behavior after restoring is different depending on the tool option settings. It shows the

following behavior.

- Source files, include files, analysis options, synthesis options, and constraint files are restored.
 - There is a constraint that -I option provided at the time of saving history in analysis options is not restored. It should be set up manually after restore processing.
 - When source files are being copy imported, and even when source files are reference imported by full path or relative path, they are overwrite restored.
 - For include files
 - ✧ When they are copy imported and when they are reference imported by full path or relative path also, include files are overwrite restored.
 - ✧ When include files are referred with -I option without being registered in the project, include files are overwrite restored under include folder of the project.
- Analysis is automatically executed after restoring. When synthesis is implemented after results restoration,
 - In case "Save synthesis results (IFF file)" is disabled in "Synthesize" tab of window displayed in "Options" of "Tools" drop-down menu, when history to be restored is saved, synthesis is not skipped.
 - In case "Save synthesis results (IFF file)" is disabled in "Synthesize" tab of window displayed in "Options" of "Tools" drop-down menu, when history to be restored is saved, synthesis is skipped.
 - As mentioned before since there is a constraint that -I option is not restored, so there is a possibility that either there will be failure in automatic execution of analysis or unintended include file is used.

Moreover, in case of rtl project, when source files are restored regardless of the value of "Save synthesis results (IFF file)" in "Synthesize" tab of the window displayed in "Options" of "Tools" drop-down menu, when history to be restored is saved, then synthesis and rtlpars should be manually started after restoration. As a result of that, Synthesis result such as IFF is saved in history and even if it is restored, the result is overwritten.

Restore constraint files

At the time of restoring constraint files, during synthesis if a constraint file that is not under process folder is referred, all constraint files are restored under process folder at the time of restoration. However, synthesis option is restored to the same option as at the time of implementing synthesis.

This is applicable when standard FLIB and standard BLIB are referred but it is a measure that CWB-gui does not overwrite standard FLIB and standard BLIB at the time of restore results. Please relocate the constraint file restored under the process folder to the desired location by exporting it from the context menu of the file as and when required.

12. Functionality to analyze synthesis results

CWB provides various methods to analyze the results of behavioral synthesis. This section covers such functionalities.

12.1. Display of linkage of source file and RTL etc

By double clicking the BDL, SystemC source files in project management window, the files can be opened in CWB internal editor. On left side of internal editor, there is a column showing row number but by clicking this row number, Verilog and VHDL rows corresponding to it can be highlighted. Further, on contrary, if Verilog and VHDL row numbers are clicked, row of the corresponding source file can be highlighted. Accordingly, most of the measures for conversion of description can be taken easily through behavioral synthesis.

12.1.1. Display of linkage of source file and Verilog

Linkage can be displayed between BDL, SystemC source files and Verilog, generated result of RTL generation, behavioral synthesis, and analysis from source code.

Conditions

For using this functionality, source file analysis (bdlpars /scpars), behavioral synthesis (bdltrn), RTL generation (veriloggen) should be completed.

Operation method

This functionality can be used by going through the following functionalities.

1. Select "Verilog" in jump destination select combo box in Operations toolbar.
2. Open BDL/SystemC source file or Verilog generated by RTL generation in CWB-gui internal editor.
3. Select the part highlighted in row number of CWB-gui internal editor.

Figure 12-1 shows how this functionality is used in actual. BDL source code is opened in left CWB-gui internal editor and Verilog file is opened in right CWB-gui internal editor. Some of the row numbers of both internal editors are indicated with color other than black (These colors can be customized. Refer section 16.1). By clicking here, corresponding row of Verilog file in case of BDL source code and row of BDL source code in case of Verilog file will be highlighted with green color.

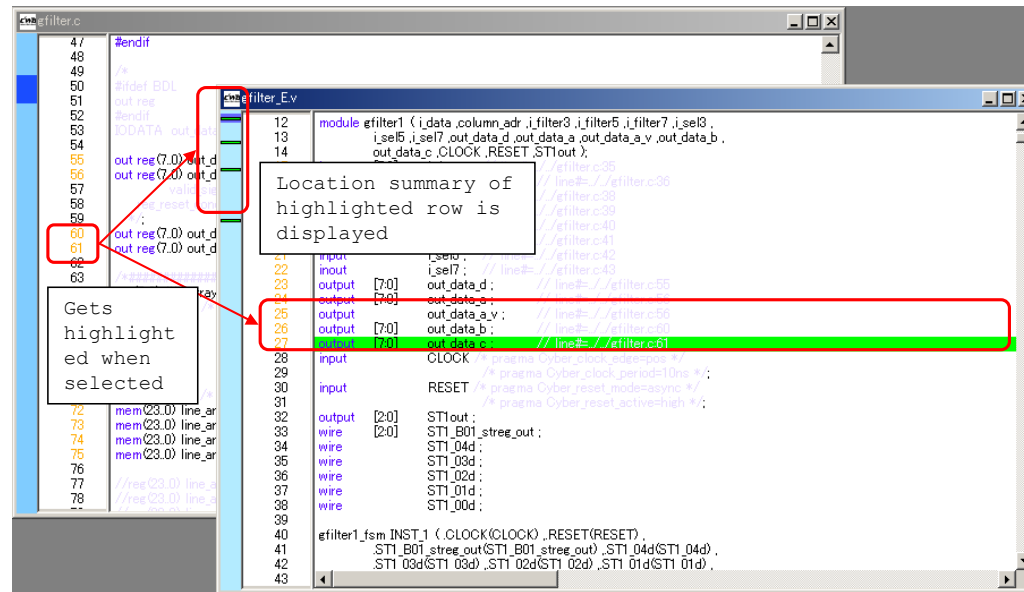


Figure 12-1 Display of linkage of BDL and Verilog

12.1.2. Display of linkage of source file and VHDL

Linkage can be displayed between BDL, SystemC source files and VHDL, generated result of RTL generation, behavioral synthesis, and analysis from source code.

Condition

For using this functionality, source file analysis (bdlpars /scpars), behavioral synthesis (bdltrn), RTL generation (vhdlgen) should be completed.

Operation method

This functionality can be used by going through the following functionalities.

1. Select "VHDL" in jump destination select combo box in Operations toolbar.
2. Open BDL/SystemC source file or VHDL generated by RTL generation in CWB-gui internal editor.
3. Select the part highlighted in row number of CWB-gui internal editor.

Please refer section 12.1.1 for example of this functionality.

12.1.3. Display of linkage of source file and cycle level BDL

Linkage can be displayed between BDL, SystemC source files and cycle level BDL(_C.BDL) generated as a result of behavioral synthesis.

Conditions

For using this functionality, source file analysis (bdlpars /scpars), behavioral synthesis

(bdltrn), should be completed. Further, cycle level BDL should be generated by any one of the following methods.

- When checkbox of "Output BDL file (*.BDL)" in behavioral synthesis options "Synthesis analysis result output" tab is checked, implement behavioral synthesis
- Execute "generate BDL" from context menu for Cycle level internal format file (_C.IFF) generated after behavioral synthesis implementation

Operation method

This functionality can be used by going through the following steps.

1. Select "cycle level BDL" in jump destination select combo box in Operations toolbar.
2. Open BDL/SystemC source files in CWB-gui internal editor.
3. Select highlighted items in row number of CWB-gui internal editor.

Please refer section 12.1.1 for the example of this functionality.

12.1.4. Display of linkage of source file and signal state and its handling table

Linkage can be displayed between BDL, SystemC source files, signal state handling table (section 12.3).

Condition

For using this functionality, source file analysis (bdlpars /scpars) and behavioral synthesis (bdltrn) should be complete.

Operation method

This functionality can be used by going through the following steps.

1. Select "signal table" in jump destination combo box in Operations toolbar.
2. Open BDL/SystemC source file in CWB-gui internal editor.
3. Select the highlighted items in row number of CWB-gui internal editor.

Above, state corresponding to the relevant row will be highlighted.

Figure 12-2 shows how this functionality is used in actual. BDL source codes are opened in signal table and CWB-gui internal editor. Some of the row numbers in internal editor are displayed in colors other than black (These colors can be customized. Refer section 16.1). By clicking here, state corresponding to signal table will be highlighted.

Further, by double clicking the label of following part of Signal table, row corresponding to the source code will be highlighted.

- Cell of Signal name (Port A of Figure 12-2 etc.)
- Label of state name (01 and 02 of Figure 12-2 etc.)
- Cell of access state (R and W of Figure 12-2 etc.) (

12.1.5. Notes for display of linkages

When source file is opened in CWB-gui internal editor, even on changing the contents of jump destination selection combo box, jump destination component of corresponding source code will not change.

For example, when option for jump destination select combo box is “Verilog” and the source code is opened in CWB-gui internal editor, then even after changing the option for jump destination select combo box as “signal table”, jump destination option for the editor remains Verilog

By executing the “Reload” from context menu of internal editor, jump destination information will be updated, and it will jump to signal table.

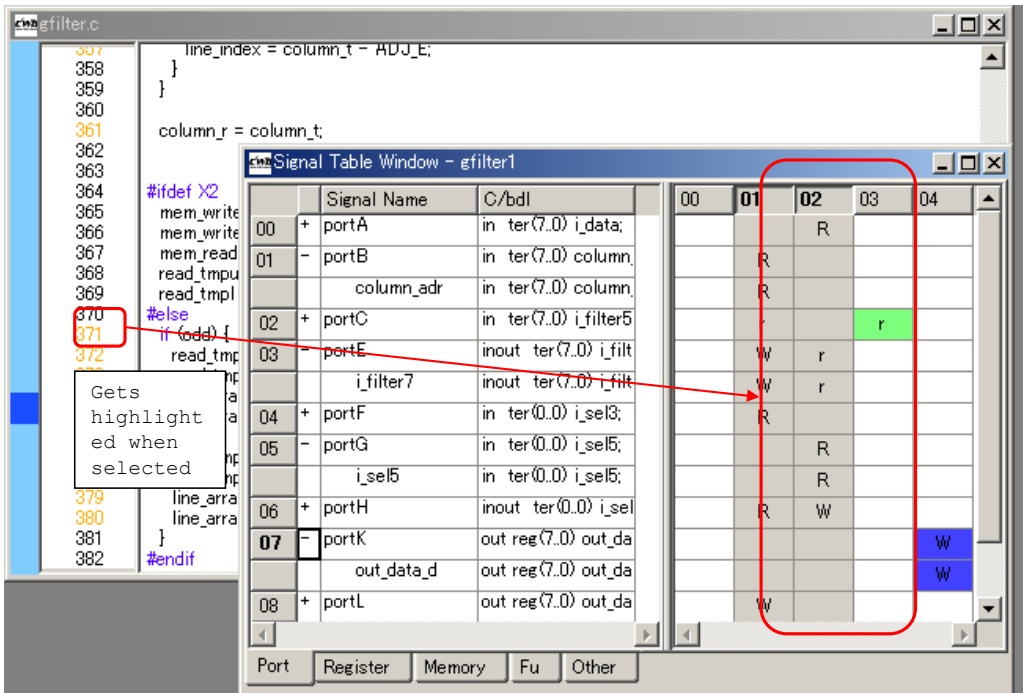


Figure 12-2 Display of linkage of BDL and signal table

12.2.State transition display in editor

State change and the corresponding relationship between row and the state can be displayed for source files in internal editor.

Conditions

For this functionality, source file analysis and behavioral synthesis should be complete.

Operation method

You can display state transition diagram by any one of the following methods.

- 1) Select relevant process from “State transition diagram display” slide menu in

context menu of source file in project management window

- 2) When source file is opened in internal editor, select relevant process from "State transition diagram display" slide menu in context menu of internal editor

In the "State transition diagram display" slide menu all processes that can be obtained from the relevant source code appear but only processes in the synthesis complete state can be selected.

In "State transition diagram display" slide menu, "Hide" can be selected. Displayed state transition can be hidden by selecting this.

Figure 12-3 shows the condition in which this functionality is used in actual. State change and handling of row and state is shown in the right side of the editor.

State 02 in the Figure is shown with blue color. This is the folding state and it shows that it is built from one loop FL1. Moreover, as loop FL1 is in stage 2, so it becomes a circuit in state 2 against the respective stage and it shows the existence of FSM that controls these state changes. For the folding state, display and undisplay of stage and state can be changed by clicking the state name.

Furthermore, by selecting the rectangle that shows the handling of row and state, it jumps to the corresponding part of Signal state handling table (section 12.3).

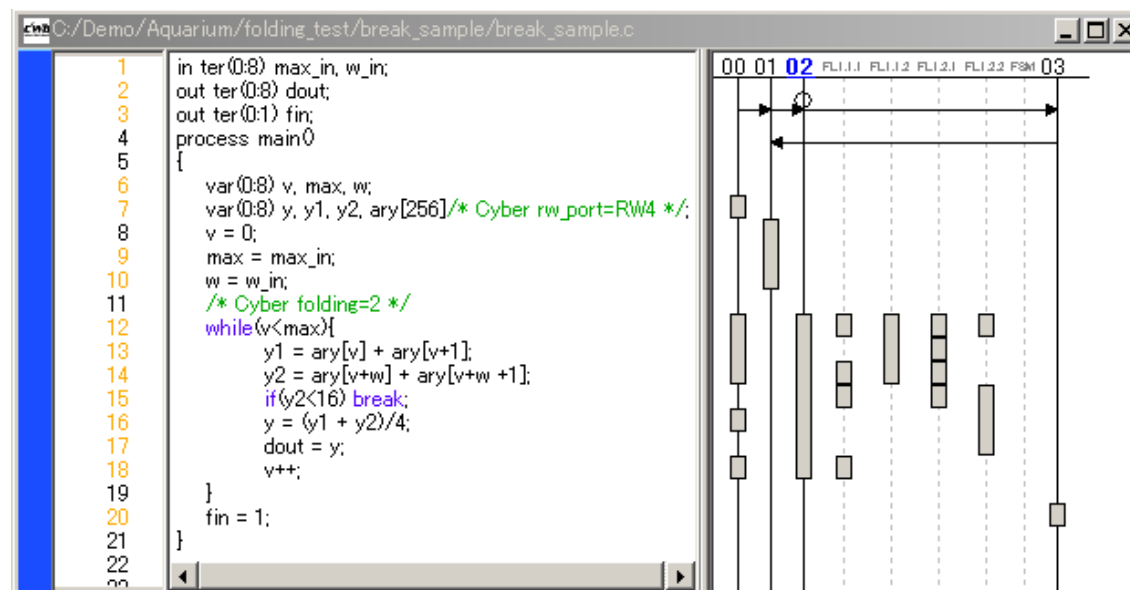


Figure 12-3 State transition display of source file

12.3.Signal state handling

Information of signals such as I/O signal and register, memory is displayed in table through behavioral synthesis showing state of each signal as whether they are being loaded, written, or both loading and writing is being performed. By referring this table, scheduling

of each signal can be understood, so it is helpful in tuning the source files.

There are 5 tabs in Signal state handling table i.e. port, register, memory, functional unit and others. Example of port signal is shown in Figure 12-4. Referring method and operation method of respective signals is explained from section 12.3.1 onwards.

Signal Name	C/bdl	00	01	02	03	04
column_adr	in ter(7.0) colu		R			
i_data	in ter(7.0) i_da				R	
i_filter3	in ter(7.0) i_filt		r			
i_filter5	in ter(7.0) i_filt				r	
i_filter7	inout ter(7.0) i		W	r		
i_sel3	in ter(0.0) i_se		R			
i_sel5	in ter(0.0) i_se				R	
i_sel7	inout ter(0.0) i		R	W		
out_data_a	out reg(7.0) out					
out_data_a_v	out reg(7.0) out		W			
out_data_b	out reg(7.0) out				W	
out_data_c	out reg(7.0) out					W
out_data_d	out reg(7.0) out					W

Figure 12-4 Signal state handling table Port tab

Opening condition and method

This table can be opened by selecting "signal state handling table" in context menu of process, only against process with normal completion of behavioral synthesis. You can also start it from "Signal table" of the command palette.

Common operation for all tabs

- **CSV output:** Contents of the tab opened can be output as CSV file. It can be executed by selecting "CSV output" from context menu of signal table.
- **Select:** By clicking the label of signal or state, corresponding signal or state can be selected. Multiple signal state selection can be done.
- **Search:** Signal matched by search is highlighted. It can be executed by any of the following methods.
 - Select "Find" from context menu of signal table.
 - Select "Find" from drop-down menu "edit", when signal table is active.
- **Non display of signal:** selected signal is not displayed. It can be executed by any of the following ways.
 - Select "hide signals" from context menu in access table in signal table.
 - Select "hide signals" from context menu of signal name.
 - Select "hide signals" from context menu of signal label.

- Do not display state: Sets the selection state as do not display. It can be executed by any of the following ways.
 - Select "hide states" from context menu of access table in signal table.
 - Select "hide states" from context menu of label of state.
- **Re-display:** It re-displays all the signals and states including the one set as do not display. It can be executed by selecting "redisplay" from context menu of signal table.
- **Display of related source codes:** Description of source codes corresponding to the selected part is highlighted. It can be executed by selecting "Display related source codes" from context menu of signal table.
- **Print:** Contents of the opened signal table can be printed. It can be executed by any of the following ways.
 - When signal table is set as active, Select "Print" from drop down menu "file".
 - When signal table is set as active, press "print" button from general tool bar.

12.3.1. Port tab reference and operation method

Access information related to Input output port can be displayed by pressing "Port" button in upper part of Signal table (Figure 12-4).

Each row

Input output signals are enumerated.

Each column

- Signal Name : input output signal name.
- C/BDL : description of declaration part of input output signal in source file
- 00, 01 ... : It shows the access condition of Input output signal in each state generated by behavioral synthesis. Symbol of each cell refers to as follows.
 - R : there is read access
 - r : there is read access in conditional clause
 - W : there is write access
 - w : there is right access in conditional clause
 - RW : there is read write access
 - rw : there is read write access in conditional clause

Operation

- Double click Signal Name, C/BDL: source file is opened and description of declaration part of input output signal is highlighted.
- Double click access display part of state: source file is opened, and corresponding access description part is highlighted.

When port constraint file and port relation file is used

When port constraint file (PLMT) and port relation file (PREL) is used.

Signal Name	C/bdl	00	01	02	03	04
Port						
out_data_a_v	out reg(7.0) out					
portA	in ter(7.0) i_da			R		
i_data	in ter(7.0) i_da			R		
portB	in ter(7.0) colu		R			
column_adr	in ter(7.0) colu		R			
portC	in ter(7.0) i_filt		r		r	
i_filter3	in ter(7.0) i_filt		r			
i_filter5	in ter(7.0) i_filt				r	
portE	inout ter(7.0) i		W	r		
i_filter7	inout ter(7.0) i		W	r		
portF	in ter(0.0) i_se		R			
i_sel3	in ter(0.0) i_se		R			
portG	in ter(0.0) i_se			R		

Figure 12-5 Port tab when using PLMT, PREL

Signal name will be displayed in Signal Name column after replacing it through PLMT/PREL. Further, "+" is displayed before table of each signal of Signal Name. By clicking it, signal before replacement will be indented and displayed. However, access timing of signal before replacement cannot be displayed.

12.3.2. Reference and operation method of Register tab

Access information related to Register can be displayed by enabling "Reg" tab in the upper part of Signal table (Figure 12-6).

Each row

Register will be enumerated.

Each column

- Signal Name : Register Name
- C/BDL : Description of signal declaration equivalent to register in source file
- 00, 01, ... : It shows the access condition of register in each state generated by behavioral synthesis. Symbol of each cell is same as in port signal.

Operation

- Double click signal Name, C/BDL: source file is opened and description of declaration part of corresponding signal will be highlighted.
- Double click state access display part: source is opened and corresponding access description part is highlighted.

Signal Name	C/bdl	00	01	02	03	04
Register						
out_data_a	out reg(7.0) out		W			
out_data_b	out reg(7.0) out				W	
out_data_c	out reg(7.0) out					W
out_data_d	out reg(7.0) out					W
RG_cclm_max_fifo_array_max_val2	var(15.0) cclm_		rw		RW	RW
RG_fifo_array	var(7.0) fifo_arr		R	W		RW
RG_fifo_array_1	var(7.0) fifo_arr					RW
RG_fifo_array_10	var(7.0) fifo_arr		R			RW
RG_fifo_array_11	var(7.0) fifo_arr					RW
RG_fifo_array_12	var(7.0) fifo_arr		R			RW
RG_fifo_array_13	var(7.0) fifo_arr					RW
RG_fifo_array_14	var(7.0) fifo_arr		R	W		RW
RG_fifo_array_15	var(7.0) fifo_arr					RW

Figure 12-6 Register tab

12.3.3. Reference and operation method of Memory signal

Access information related to Memory can be displayed by enabling “Mem” tab in the upper part of Signal table (Figure 12-7).

Each row

Memory will be enumerated.

Each column

- Signal Name : It is memory name.
It is displayed as memory instance name (memory type). For example, in first line of Figure 12-7, line_array_el is equivalent to memory instance name, and RW1_wrapper is equivalent to memory type. When “+”at the top of memory instance name is clicked, memory signal related to corresponding memory will be indented and displayed under memory instance name.
- C/BDL : It is signal declaration description equivalent to memory in source file. Nothing is displayed for memory signal.
- 00, 01, ... : It indicates access condition of memory in each state generated by behavioral synthesis. Signal of each cell is same in port signal.

Operation

- Double click Signal Name, C/BDL: Source file is opened, and description of corresponding memory declaration part is highlighted. For memory signal, source

file jump does not take place by double click.

- Double click the state access display part: source file is opened and corresponding access description part is highlighted.

Signal Name	C/bdl	00	01	02	03	04
Memory						
line_array_el(RW1_wrapper)	mem(23.0) line_		w	r		
RW1_wrapper.A			w			
RW1_wrapper.BE						
RW1_wrapper.BUB						
RW1_wrapper.CSB			w			
RW1_wrapper.DI			w			
RW1_wrapper.DO				r		
RW1_wrapper.TBE						
RW1_wrapper.TEST						
RW1_wrapper.WEB			w			
line_array_eu(RW1_wrapper)	mem(23.0) line_		w	r		
line_array_ol(RW1_wrapper)	mem(23.0) line_		w	r		
RW1_wrapper.A			w			

Figure 12-7 Memory tab

12.3.4. Reference and operation method of functional unit tab

Access information related to functional unit can be displayed by enabling "Fu" tab in the upper part of Signal table (Figure 12-8).

Each row

Functional unit used in behavioral synthesis is enumerated.

Each column

- Signal Name : Functional unit name
- C/BDL : Blank column.
- 00, 01, ... : It shows the access condition of functional unit in each state generated by behavioral synthesis. The symbol of each cell is as follows.
 - USE : Functional unit is used
 - use : Functional unit is used in conditional clause

Operation

- Double click Signal Name: Source file is opened, and all the access description used by corresponding functional unit will be highlighted.
- Double click state access display part: source file is opened and corresponding

access description part is highlighted.

Signal Name	C/bdl	00	01	02	03	04
Function Unit						
...add08_09@1			use	USE		
...add08_09_01@1						USE
...add16_16u@1			USE	USE	USE	use
...add16_16u_01@1			USE	USE	use	use
...add16_16u_02@1			USE	USE	use	USE
...add16_16u_03@1			use	USE	use	
...lop08_01u@1			USE			
...lop16_01u@1			USE	USE	USE	
...lop16_01u_01@1			USE	USE	USE	
...sub07_08u@1			use			

Figure 12-8 Functional unit tab

12.3.5. Reference and operation method for Others signal

Access information related to Others signal can be displayed by enabling "Others" tab in the upper part of Signal table (Figure 12-9).

Each row

Others signal is enumerated.

Each column

- Signal Name : Signal name.
- C/BDL : Blank column.
- 00, 01 ... : It shows the access condition of register in each state generated by behavioral synthesis. Symbol of each cell are same as in Port signal.

Operation

- Double click state access display part: source file is opened, and corresponding access description part is highlighted.

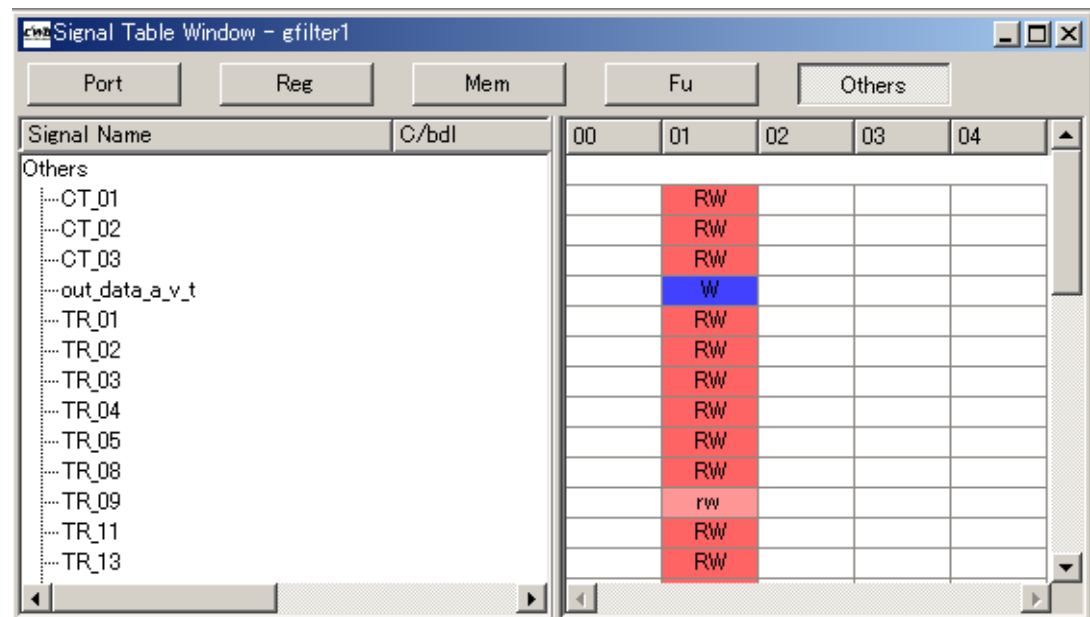


Figure 12-9 Others tabs

12.3.6. Display of Pipeline state

The relevant state for pipeline and folding states is displayed by dividing it into stages and states (Figure 12-10).

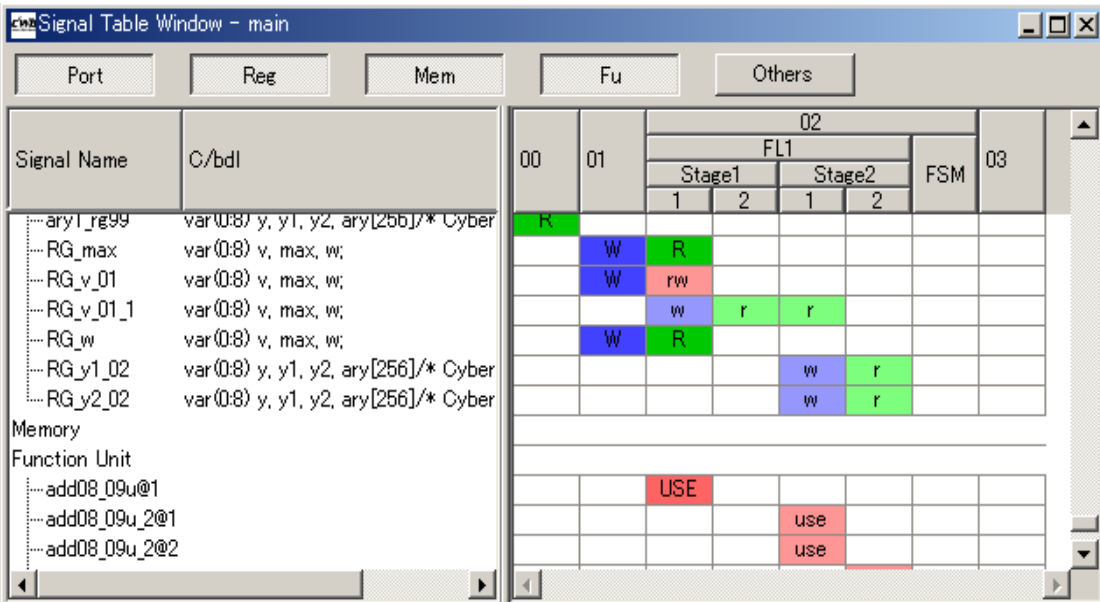


Figure 12-10 Display of Folding state

12.4. Linkage with delay report

By loading the delay report of design compiler, delay report and source code, rtl relation can be investigated.

12.4.1. Basic functionality

Condition

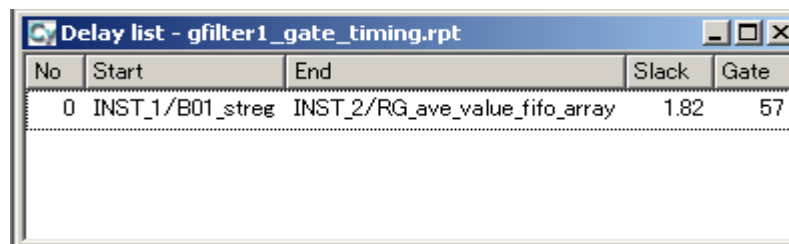
For using this functionality, source file analysis (bdlpars/scpars), behavioral synthesis (bdltran) and RTL generation (veriloggen/vhdlgen) should be completed and delay report file corresponding to that RTL should be generated by Design Compiler.

CWB-gui does not have the functionality to automatically execute Design Compiler in order to generate delay report. Please create delay report by executing the Design Compiler manually.

Operation method

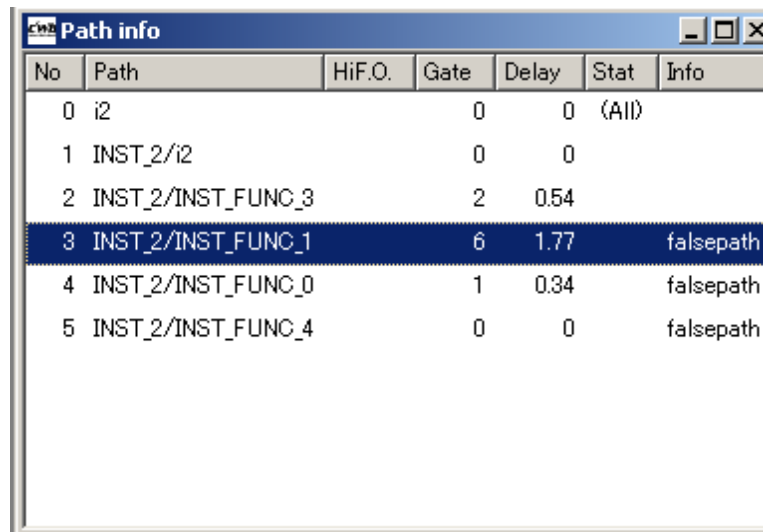
This functionality can be used by going through these steps.

- 1) "Verilog" or "VHDL" is selected in jump destination select combo box of operation tool bar (section 3.2.2). The RTL selected here will show the linkage with delay report. This functionality cannot be used even if "cycle level BDL" is selected.
- 2) "Load delay report" is selected in context menu of work folder of target process, and delay report is loaded.
- 3) If loaded successfully, path list is displayed (Figure 12-11).
- 4) If any path on path list is double clicked, detailed information of corresponding path is displayed. (Figure 12-12).
- 5) By double clicking the instance in detailed information of path, delay report, corresponding source code and RTL will be displayed and highlighted in CWB-gui internal editor.



No	Start	End	Slack	Gate
0	INST_1/B01_streg	INST_2/RG_ave_value_fifo_array	1.82	57

Figure 12-11 Path list



No	Path	HiF.O.	Gate	Delay	Stat	Info
0	i2		0	0	(All)	
1	INST_2/i2		0	0		
2	INST_2/INST_FUNC_3		2	0.54		
3	INST_2/INST_FUNC_1		6	1.77		falsepath
4	INST_2/INST_FUNC_0		1	0.34		falsepath
5	INST_2/INST_FUNC_4		0	0		falsepath

Figure 12-12 Detailed information of path

12.4.2. False path display

In addition to the basic functionalities of section 12.4.1, in the detailed information of path list or path, it can be shown whether the corresponding path is a false path or not.

It can be used when it is required to take measures for issues related to delay and the result of logical synthesis but the user does not want to use false path script and wants to identify the false path in critical path instantly.

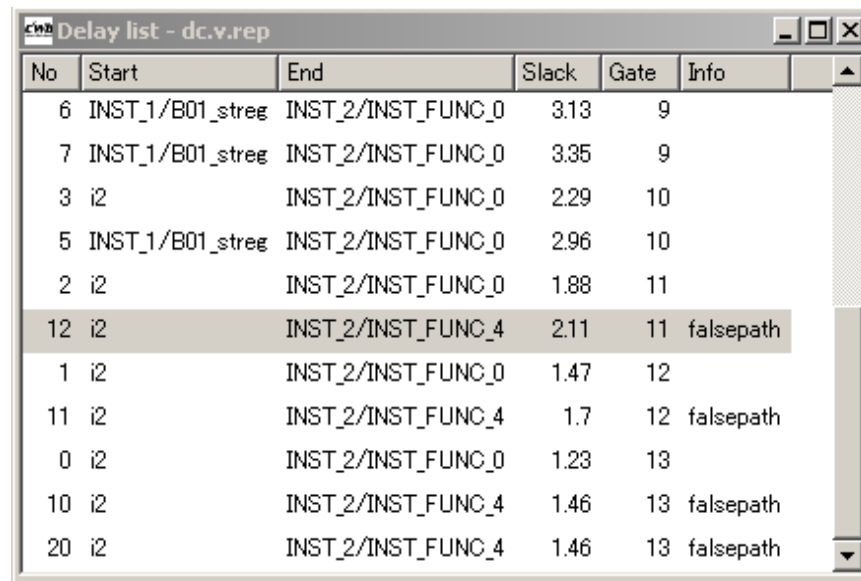
Conditions

For using this functionality, the following steps should be completed.

- Source file analysis (bdlpars/scpars), behavioral synthesis (bdltran), RTL generation (veriloggen/vhdlgen) should be completed and delay report file corresponding to that RTL should be generated by Design Compiler.
- This functionality uses false path information by which behavioral synthesis does file output. In order to generate this file behavioral synthesis should be executed again after specifying "-Zfalse_path_script" option. This option becomes effective by checking the checkbox of "False path script output" in "Synthesis analysis results output" of behavioral synthesis options screen.
- It is necessary to specify "-OD" option during RTL generation. In "general" tab of RTL generation option screen, set the "target" to "Design Compiler".

Operation Method

By ticking the check box "Load falsepath scripts", provided at the bottom most part in the "load delay report" screen, as given in steps of section 12.4.1, and by loading the delay report, false path information will be displayed as shown in Figure 12-13.



No	Start	End	Slack	Gate	Info
6	INST_1/B01_streg	INST_2/INST_FUNC_0	3.13	9	
7	INST_1/B01_streg	INST_2/INST_FUNC_0	3.35	9	
3	i2	INST_2/INST_FUNC_0	2.29	10	
5	INST_1/B01_streg	INST_2/INST_FUNC_0	2.96	10	
2	i2	INST_2/INST_FUNC_0	1.88	11	
12	i2	INST_2/INST_FUNC_4	2.11	11	falsepath
1	i2	INST_2/INST_FUNC_0	1.47	12	
11	i2	INST_2/INST_FUNC_4	1.7	12	falsepath
0	i2	INST_2/INST_FUNC_0	1.23	13	
10	i2	INST_2/INST_FUNC_4	1.46	13	falsepath
20	i2	INST_2/INST_FUNC_4	1.46	13	falsepath

Figure 12-13 Display of false path information

12.5. Linkage with other analysis tools

Analysis tools such as data viewer and BDL viewer can be started from CWB.

12.5.1. Linkage with Data path viewer

Behavioral synthesis result can be displayed as circuit drawing by using pathdraw program.

Conditions

For using this functionality, behavioral synthesis (bdltran) should be normally completed and structural level internal format file _E.IFF should be generated in project management window.

Operation method

Pathdraw application can be started by selecting "Data path display" from context menu of structural level internal format file _E.IFF or process in the project management window (Figure 12-14). You can start it by clicking "Data path" on command palette.

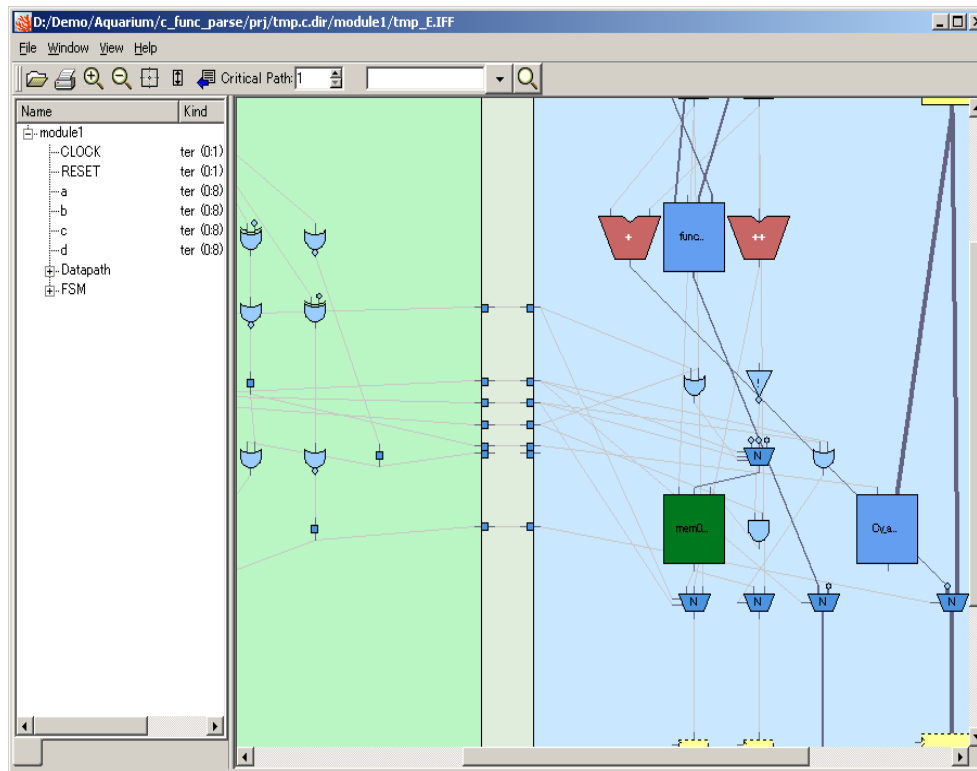


Figure 12-14 Display of circuit diagram display through pathdraw

Linkage with CWB-gui

Linkage between pathdraw and CWB-gui can be displayed.

- When component of circuit drawing or signal in the signal list at left is clicked, source code will be opened in CWB-internal editor and corresponding line will be highlighted.
- If critical path is displayed in pathdraw, source code is opened in CWB-gui internal editor and corresponding row is highlighted.

12.5.2. BDL display by BDL viewer

Contents of internal file (.IFF) of behavioral synthesis can be viewed by using bdlldraw program.

Operation method

bdldraw application is started by double clicking the .IFF file in project management window (Figure 12-15).

When "BDL information" is selected from context menu for description in bdlldraw, the corresponding source code opens and the relevant row is highlighted.

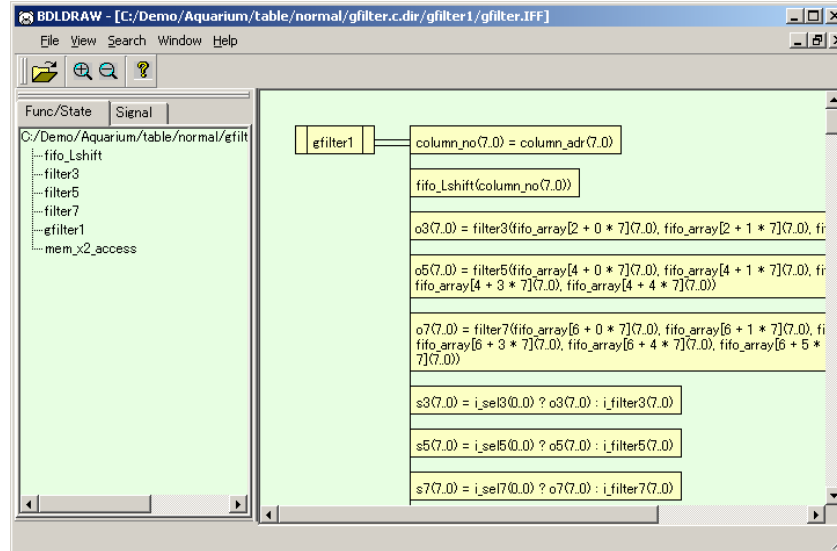


Figure 12-15 Information display through bdlldraw

12.5.3. Circuit quality display

Circuit quality of behaviorally synthesized result can be viewed by HTML viewer, embedded in CWB-gui.

Conditions

For using this functionality, behavioral synthesis (bdltran) should be normally completed and circuit quality file (.QOR.HTML) should be generated in project management window.

Operation method

Circuit quality can be viewed by double clicking the .QOR.HTML files in project management window. (Figure 12-16) . You can start it by clicking "Quality of result" on command palette.

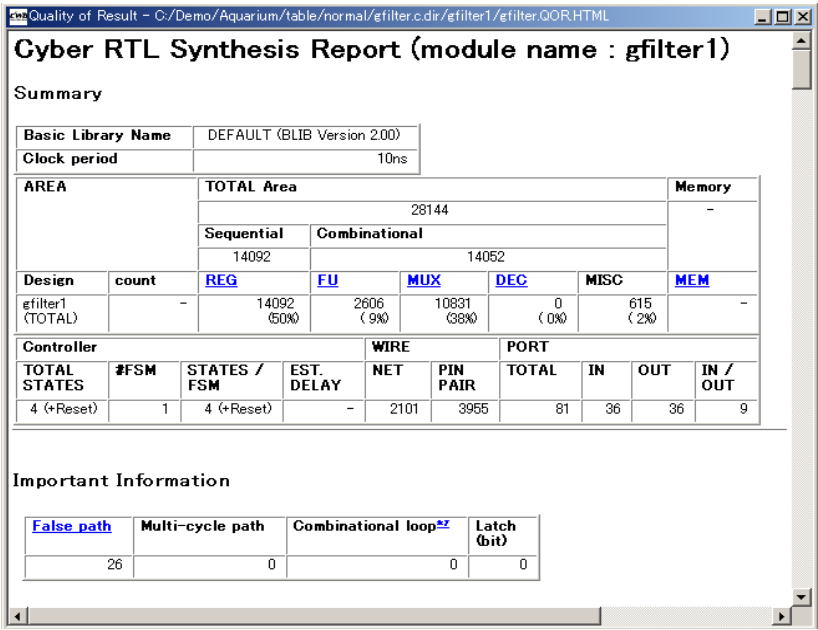


Figure 12-16 Circuit quality display

13. Bus interface automatic generation

In case of ASIC, by using bus IF automatic generation functionality of CWB, bus description can be generated as BDL source files. Two programs cybus and CybusM are used in bus IF automatic generation. For details of such programs, please refer to their respective manuals.

13.1. cybus

cybus is tool to automatically generate HW/SW-IF BDL including address decoder, etc from address map table of register, placed between bus (CPU) and slave, for communication between HW/SW.

cybus option setting

cybus option setting screen can be started by any of the following.

- 1) Select "generate bus interface circuit" from context menu of intermediate project in project management window.
- 2) When intermediate project is selected in project management window, select "generate bus interface circuit" from drop down menu "architecture design".

cybus execution

For executing cybus, start cybus option setting screen and press "generate" button.

Generated file

Source files and include files generated by cybus are stored to source file folder and include file folder of project, on which cybus is to be executed.

13.2. CybusM

CybusM is a tool to automatically generate BDL of bus IF as well as On-chip bus from bus definition file, that describes bus specifications.

Prerequisite of CybusM execution

Clock cycle, standard FLIB and standard BLIB should be specified in the properties of the project. Please refer section 8.1.2 for details.

CybusM option setting

CybusM option setting screen can be started by any of the following ways.

- 1) Select "On-chip bus automatic generation" from context menu of intermediate project in project management window.
- 2) When intermediate project is selected in project management window, select "On-chip bus automatic generation" from drop down menu "architecture design"

CybusM execution

For executing CybusM, start CybusM option setting screen, and press "Generate" button.

Execution control and Generated file

There is bus definition file (hereafter, referred as bdef file) for input of CybusM.

- CybusM is executed in CWB-gui.
 - CybusM generates source file, include file and Makefile.
 - ✧ Naming of Makefile is as following.
 - When bus type is AHB, AHB Multi-Layer: Makefile.ahb
 - When bus type is AXI: Makefile.axi
 - When bus type is APB:
 - When bridge output is not specified (when "No" is specified for "Generate AHB2APB bridge" in properties of bus in bdef editor to be mentioned hereafter): Makefile.apb
 - When bridge output is specified (when "Yes" is specified for "Generate AHB2APB bridge"): Makefile.apb, Makefile.ahb2apb
 - When bus type is AHB2APB: Makefile.ahb2apb
 - When bus type is AXI2AHB: Makefile.axi2ahb
 - When bus type is AHB2AXI: Makefile.ahb2axi
- After the CybusM execution is completed, make execution of Makefile is conducted automatically. Consequently, analysis, synthesis and RTL generation is done for source file generated by CybusM.
 - As per the value of RTL selection (section 3.2.2) in Operations toolbar, it is decided whether Verilog or VHDL will be generated in RTL generation.
- Source file, include file and each file generated through make execution is registered in the project management window.
 - Source files and include files are stored in source file folder and include file folder of the project for CybusM execution.
 - Each file generated by synthesis and RTL generation is stored under work directory of related process.
 - Makefile is deleted after automatic execution of make without getting registered in project management window.

Deleted files

In the project for execution, in case CybusM was executed earlier then all the source files, include files, and synthesis result generated at that time is cleared away.

Display in project management window

In the generated source files and include files there are hidden attributes. In the operations corresponding to the project, only the file in which hidden attributes are provided can be undisplayed. (section 4.3.2). Due to this, the display of automatically generated file by CybusM and manually edited file can be distinguished.

13.3. Usage method of bdef editor

bdef file that becomes input of CybusM can be edited in special editor in CWB-gui.

Starting method of bdef editor

In Options screen of CybusM, "Reference" or "New" in "Bus definition file"

- Press "New" in "Bus definition file"
- When value is specified in "Bus definition file" press "Edit"

It can also be opened by double clicking bdef file in project management window.

Screen structure

Example of screen of bdef editor is shown in Figure 13-1.

- Column: Shows one bus.
- Row: Shows module connected with the bus. Master is shown in light blue color and Slave is displayed in yellow color.
- Cell: Intersection point of bus and module shows the connecting relation between relevant bus and the corresponding module. Blank cell indicates that there is no connection. A numeric figure shows that there is a connecting relation. Connecting Master and Slave against one bus can be independently numbered. This shows connecting order of Master and Slave, and a small number is prioritized. As shown in Figure 13-1, bus of bus11 is connected with master1 and master 2 in the sequence 1 and 2 respectively. For Slave, 1 is assigned to NewSlave. Same number can not be assigned for multiple masters of one bus. It is same for Slave also.

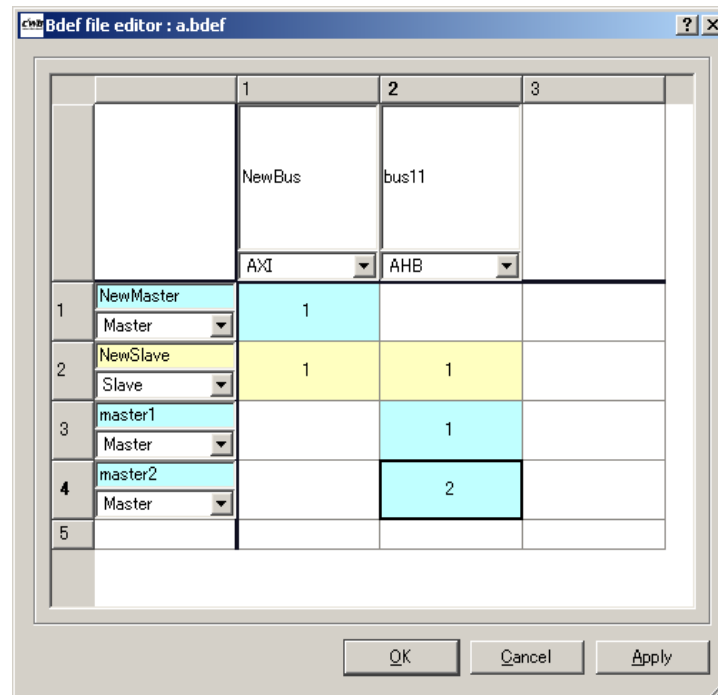


Figure 13-1 bdef editor

Operations

Following Operations can be performed in bdef.

- Operations for bus
 - Change of bus name: When bus name is clicked, cursor is displayed and it becomes editable.
 - Change of bus type: Type of bus can be changed in combo box under bus name. When type of bus is changed, settings for the connection between relevant bus and module of bus is removed. As an exception, in the connection of Slave, the settable address map information is maintained.
 - Insertion of bus: When "Insert" is selected by right click menu of bus name, new bus is added in front of the selected bus.
 - Properties setting of bus: When "Properties" is selected by right click menu of bus name, attributes for bus can be specified.
 - Alignment: Aligning of module is done for the corresponding bus as per the connection state. Aligning is done in the following order. Figure 13-2 shows the example of screen in which aligning is implemented for bus 11 of Figure 13-1.
 - ✧ Master (When multiple masters are connected, master is aligned in the order of small numeric figures)
 - ✧ Slave (When multiple Slaves are connected, Slave is aligned in the order of small numeric figures)
 - ✧ Unconnected module
 - Delete: When "Delete" is selected by right click menu of bus name, selected bus is deleted.

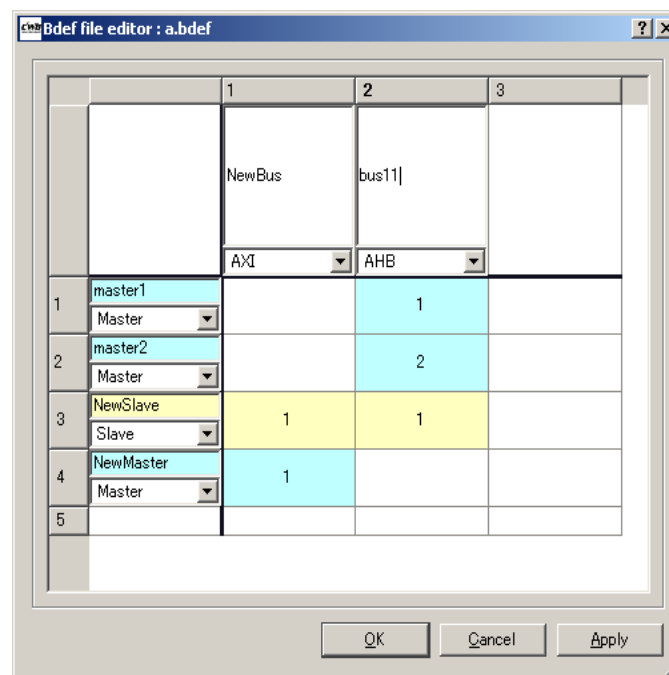


Figure 13-2 Editor screen after array

- Operations for module
 - Change of module name: When module name is clicked, cursor is displayed and it becomes editable.
 - Change of module type: Type of module can be changed from combo box under module name. When type of module is changed, settings related to connection of corresponding module are removed. As an exception, settable address map information in connection of slave is maintained.
 - Insertion of module: When "Insert" is selected from right click menu of module name new module is created in front of selected module.
 - Delete: When "Delete" is selected from right click menu for module name, selected module is deleted.
- Operations for connection
 - Connecting bus and module: For the blank cell, when "Connect" is selected from right click menu, the bus and module corresponding to the selected cell are connected. At this time, the smallest number from the unspecified numbers is automatically assigned to the cell.
 - Properties settings of connection state: When "Properties" is selected by the right click menu for connecting cell, the attributes for the corresponding connection can be specified. Settable attributes are different depending on the type of bus or master.
 - Remove connection of bus and module: When "disconnect" is selected from right click menu for connecting cell, connection can be removed.

14. Top Module Generation

Multiple Internal format files (Output .IFF File of each Source-file Analysis Tool) of behavioral description level and Internal format files (_E.IFF) of structural level are a input to the top module generation tool topmodgen. These modules are kept as lower-level modules one by one, and higher level hierarchy description that connects those modules is generated.

Behavior of CWB-gui differs depending upon the Options given in topmodgen.

Please refer the Reference Manual of Cyber regarding details of topmodgen.

14.1. Option-Settings and execution of topmodgen

topmodgen is used as follows:

Option Settings of topmodgen

Option Settings Screen of topmodgen can be opened by any one of the following methods:

In case of ASIC:

- 1) Select "Generate Top Module" from Context Menu of Intermediate Project in Project Management Window.
- 2) Select "Generate Top Module" from "Architecture Design" Drop down Menu in the state where one Intermediate Project is selected in the Project Management Window.
- 3) Select "Generate Top Module" from "Architecture Design" Drop-down Menu in the state where more than one process of a particular Intermediate Project is selected in the Project Management Window.

In case of FPGA:

- 1) Select "Generate Top Module" from "Architecture Design" drop down menu after selecting more than one processes in the Project Management Window.

topmodgen Execution

For executing topmodgen, start topmodgen Option Settings Screen, and press "Generate" Button.

Specifications of process for topmodgen execution

In case of ASIC, when topmodgen is executed, any of the following processes can be specified,

- 1) Process under the project in which topmodgen option settings screen is opened
- 2) Process under the project corresponding to the submodule, against the project in the project tree in which topmodgen option settings screen is opened

In the project tree shown in the left side in Figure 14-1, when topmodgen option settings screen is opened for project current_prj, module 1 and module 2 are applicable for point 1), and submodule and rt_submodule are applicable for point 2).

When topmodgen option settings screen is opened, all processes of point 1) are

automatically registered in "Module list" table of option screen.

Processes applicable for point 2) are not registered in "Module list" table when topmodgen option settings screen is opened. Register the module by using "Add module" and "Delete module" buttons.

Generation destination of Top module (Target project)

In case of ASIC, any project other than the subproject in the project tree of the project in which option settings screen is opened, can be specified for the target project of Top module.

However, the projects corresponding to the following projects can not be specified

- 1) rtl project
- 2) Specified project for generation destination is not parent project of topmodgen execution target project and moreover in the case when link is generated to the project specified for generation destination project under topmodgen execution target project

In the Figure 14-1 it is shown that in the project tree on the left side, when topmodgen option settings screen is opened in project current_prj, the generated Top module topmodule is built under project top_prj by specifying generation destination of Top module,

After execution of topmodgen, when the project specified as generation destination is not of parent project of topmodgen execution target project, link to topmodgen execution target project is generated under project specified as generation destination.

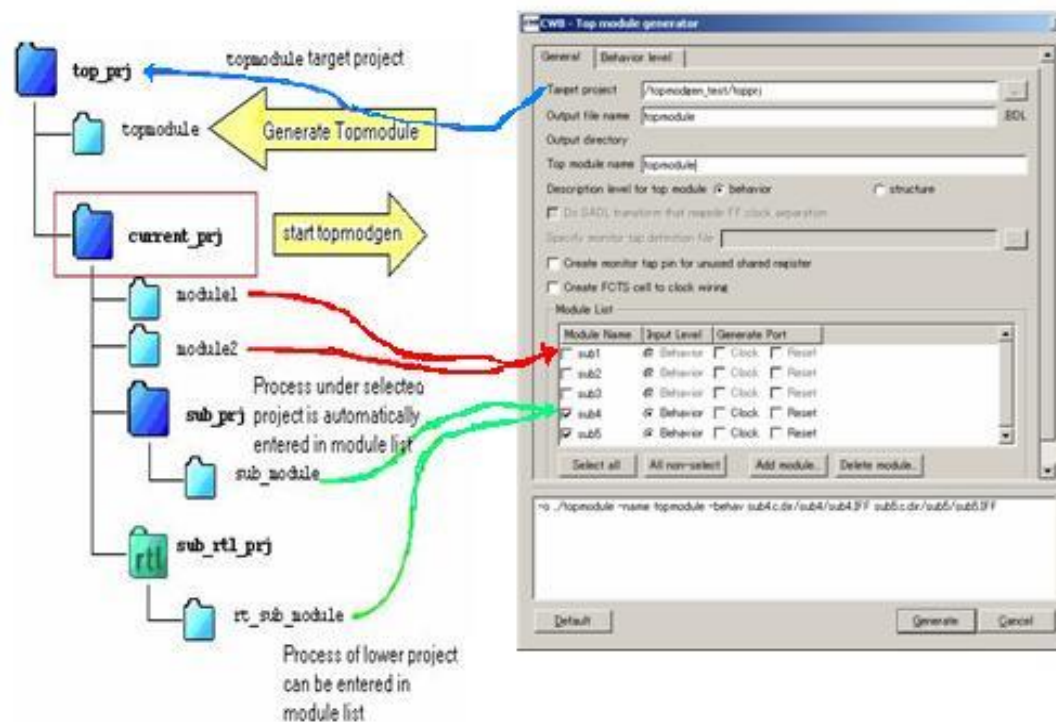


Figure 14-1 topmodule target project and target process

Generated Files

Files generated by topmodgen are different when "Structure" (Section 14.2) is selected and when "Behavior" (Section 14.3) is selected in "Description level for Top module" of Option Settings Screen. Both the cases are explained in this section.

14.2.Structural Level topmodgen

This section explains the behavior when "Structure" is selected in "Description level for Top module" of Option Settings Screen and the files generated at that time.

Behavior of Topmodgen

When "Structure" is selected in "Description level for Top module" of Option Settings Screen, topmodgen generates high-level Structural level Internal Format File.

Here, CWB-gui automatically generates BDL Source File with respect to high-level Structural level Internal Format File, and implements Source File Analysis (bdlpars) and synthesis for that file.

Generated files

Following files are generated:

- 1) High level BDL Description (.BDL) : Registered in the src Folder of the Project specified as generation destination.
- 2) Work-folder of high-level BDL Description : Work-folder regarding 1) is generated under the Project specified as generation destination.
- 3) High level Process Folder: High level Process Folder is generated below 2).
- 4) High level behavioral description level internal format file (.IFF) : Result of executing Source file analysis bdlpars with respect to 1) is generated in 3).
- 5) High-level Internal format file (_E.IFF) : High-level Internal format file that is the output of topmodgen is generated in synthesis result analysis of 3).
- 6) Information file (_E.FCNT) of generated top module: In case function and process that use generated top modules as instances do not exist, this file is generated in constraint file of 3). However, it is not registered as synthesis option. On the contrary, in case function and process that use generated top modules as instances exist, this file is copied under the applicable process and function and is registered as its respective synthesis option. Follow section 9.3 for the policy of copying this file.

14.3.Behavioral level topmodgen

This section explains the behavior, when "Behavior" is selected in "Description level for top module" of option setting screen, and about the files generated at that time.

Behavior of Topmodgen

When "Behavior" is selected in "Description level for top module" of option setting screen, topmodgen generates top module source file (.BDL).

Here CWB-gui automatically implements source file analysis (bdlpars) for generated source files.

Generated files

The following files are generated by automatic execution of topmodgen and subsequent CWB-gui.

- 1) Top hierarchy BDL description (.BDL): Registered in src folder of project specified as generation destination.
- 2) Top hierarchy BDL description work folder: Work folder equivalent to 1) will be generated under project specified as generation destination.
- 3) Top hierarchy process folder: top hierarchy process folder will be generated under 2).
- 4) Top hierarchy behavioral description level internal format file (.IFF): Result of execution of source file bdlpars for 1) is generated under 3).
- 5) Memory library file (MLIB): Sometimes one memory library file is generated. Generated memory library file is registered in "Other files" folder of project specified as generation destination. Memory library file is automatically registered as behavioral synthesis option against the process that generates memory count file.
- 6) Top hierarchy memory count file (MCNT): Sometimes memory count file is generated for top hierarchy process. Generated memory count file is registered in constraint folder of top hierarchy process. This memory count file is automatically registered as behavioral synthesis option.
- 7) Lower hierarchy memory count file (MCNT): Sometimes memory count file is generated for lower hierarchy process that is input file of topmodgen. Generated memory count file is registered in constraint folder of relevant lower hierarchy process. This memory count file is automatically registered as behavioral synthesis option.

15. Operation related to LOG file

In LOG file, all source files, header files, constraint files etc, which forms the analysis source of .IFF file that is input for behavioral synthesis, and options are compiled as one file. LOG files are automatically generated in the compressed form of gzip if behavioral synthesis is executed. For details, please refer to Cyber Reference manual.

15.1.Import of LOG file and its expansion

LOG file is imported in project of CWB-gui and environment of behavioral synthesis can be recreated when LOG file is generated by expanding it.

LOG file import

For expanding the LOG file in CWB-gui, LOG file to be expanded, is imported to "Other files" folder of intermediate project or RTL project. Select "import files" from context menu of "Other files" folder of intermediate project and RTL project and import the target LOG file.

LOG file expansion

For expanding the generated LOG file, select "Restore LOG file" in context menu for imported LOG file.

If selected, log file expansion screen will open (Figure 15-1). All the generated files and their file types will be displayed. In case file and file type handling is not appropriate, file type combo box is operated and it is changed to appropriate file type.

After LOG expansion is done, the following CWB process will be automatically executed.

- In case of intermediate project:
 - source file analysis (bdlpars/scpars) is executed.
 - For the process of LOG file that is to be synthesized, synthesis option described in LOG file is specified.
- In case of RTL project:
 - Source file analysis (rtlpars) is started; file priority order specification screen is opened and comes to a halt.
 - If rtlpars is executed as it is, synthesis option described in LOG file is specified for process of LOG file to be synthesized.

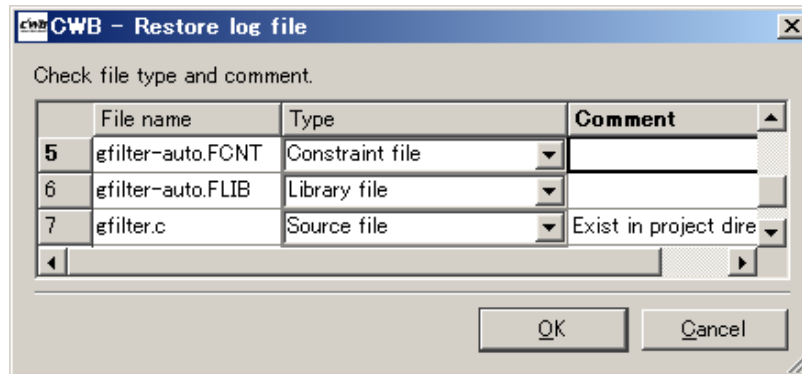


Figure 15-1 Log file expansion screen

Generated files

The following file groups will be generated by LOG file expansion.

- 1) Source file: It is entered in src folder of intermediate project and RTL project to be executed.
- 2) Include file: In case include file exist in intermediate project, it will be entered in include folder of intermediate project to be executed.
- 3) Work folder of source file: Work folder for 1) will be generated under intermediate project to be executed.
- 4) Process folder: In case intermediate project is to be executed, process folder will be generated under 3).
- 5) Behavioral description level internal format file (.IFF): Result of source file analysis for 1) will be generated in 4).
- 6) Behavioral synthesis parameter files (bdltran.prm): Behavioral synthesis parameter files are generated under "Other files" of intermediate process to be executed.
- 7) Each constraint file: There is possibility of constraint file generation through LOG expansion. Constraint files are entered under process.

15.2.LOG file export

If behavioral synthesis is executed, LOG file is generated under work directory of process to be executed.

Generated LOG files can be obtained directly by accessing the directory having files. However, if CWB-gui LOG export functionality is used, LOG files can be extracted more easily.

Methods to start the LOG export

If "export LOG" is selected in context menu of LOG generated by behavioral synthesis in project management window, LOG export can be executed.

16. Others

This section explains about the other additional functionalities.

16.1. Tool options

Various settings are possible in tool options, activated by "options" in drop-down menu "Tools".

16.1.1. General Tab

General tab can perform settings such as system font change, default value of each combo box of Operations toolbar.

- System font
 - Font name: Changes the font of statements displayed in menu etc. of CWB-gui.
- Specification of default value of Toolbar
 - RTL: Specifies the default value of RTL selection combo box of toolbar at the time of starting CWB-gui.
 - Jump: Specifies the default value of Jump destination selection combo box of toolbar at the time of starting CWB-gui.
- Project management window
 - Explanation of file: Selects the state of display and undisplay of "File explanation" column in Project Management window.
 - Status/Timestamp: Selects the display and undisplay of "Status/Timestamp" column in Project Management window.
- Console window
 - Maximum number of verification Consoles: Specifies the maximum number of Console tabs at the time of Properties verification.
 - Maximum number of rows of Console: Specifies the maximum number of rows of messages generated in Console.
- License management
 - License acquisition retry: Specifies whether to wait or not in case License could not be obtained for every program that starts from CWB-gui.

Environment variables settings

Environment Variables settings screen as shown in Figure 16-1 appears when "Environment Variables" button is pressed in lower left part of General tab. Combination of Name and Setting value can be specified in Environment Variables settings screen by using New, Edit and Delete buttons.

Values specified here become valid when OK button is pressed. For the variables deleted in this screen unsetenv will be done. Moreover, setting details will be automatically reflected as environment variables at the time of starting CWB-gui.

Settings of some Environment variables such as CYBERDIR, CYBER_LIB, PATH etc. that influence the execution of CWB system, cannot be implemented in this screen.

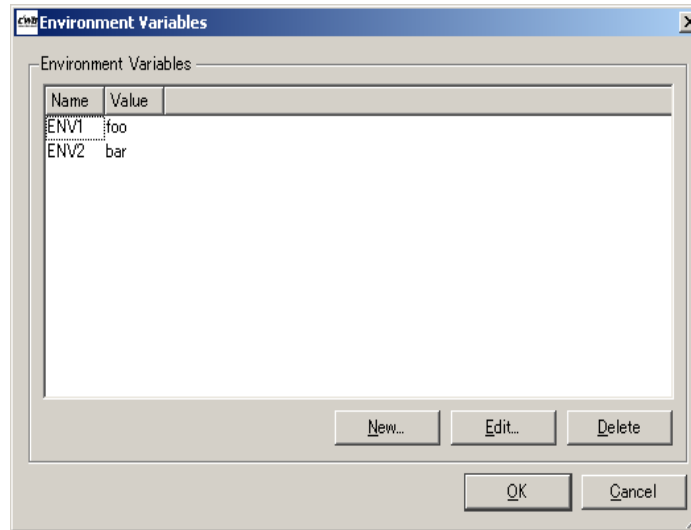


Figure 16-1 Environment variables settings

16.1.2. Keyboard tab

Keyboard tab can perform the key bind settings. When the command that needs to be set is selected and "Edit" button is pressed, then the shortcut key can be set for the specified command. The specified value is reflected in the system by pressing "Assign" button after the setting

16.1.3. Editor tab

Editor tab can perform settings such as font change, change of color used in editor, tab stop change, and external text editor settings.

- Font
 - Font name: Changes the font of statements displayed in internal editor of CWB-gui.
- Color
 - Text: Specifies display color of general characters of internal editor.
 - Comments: Specifies the display color of Comments.
 - Keyword: Specifies display color of reservation language.
 - Row number: Specifies display color of row number.
 - Row number (highlighting color): Specifies the display color of selectable row number.
 - Background color (highlighting color): Specifies the highlighting color at the time rows in some actions are highlighted.

- Tab stop
 - Width: Specifies the width of tab.
- Editor type
 - Internal editor/external editor: When text file is opened by double clicking, one can select whether to use internal editor or external editor of CWB-gui.
- Row number selection
 - Select/do not select: When "select" is specified, the row number of the row where the cursor exists is displayed as underlined bold characters.
- External editor
 - Text file editor: Specifies the binary file location of editor that is used as External text editor.
 - CSV file editor: Specifies the binary file location of CSV file editor that is used at the time of opening CSV file.
 - In case of windows, if check of "Use the default program associated with extension" is ON, programs specified associated with windows will start. For instance, programs with extension '.c' will start if it is C source code. Check must be removed before specifying the program if there are programs other than editor are associated.

16.1.4. Menu tab

Menu tab can specify number of items that can be saved in recently opened workspace and file.

- Menu setting
 - Save count of Used files log: Specifies the count of logs in "Recently opened files" in "File" drop-down menu.
 - Save count of used workspace log: Specifies the count of logs in "Recently opened workspaces" in "File" drop-down menu.

16.1.5. Signal table tab

Signal table tab can change Settings of Signal state handling table (section 12.3).

- Signal table settings
 - Width of state: In each state, initial value of width of row can be changed.

16.1.6. Execution control tab

In execution control tab, settings related to behavioral synthesis execution can be performed.

- Synthesis settings
 - Copy FCNT file generated by Synthesis to new interface: Behavioral synthesis outputs the information of function and process to be synthesized as _E.FCNT.

This FCNT file is used at the time of synthesis of process and functions that use applicable process and function as interfaces (Refer section 9.3). At the time of implementing synthesis, FCNT file of same name exists in the work directory of process and function that will be called, and in case the contents of the file are not same then the behavior selected in this item is followed.

- ✧ Copy: Setting is done in synthesis option by copying by overwriting the _E.FCNT file in work directory to be synthesized that calls the conversion of function into functional unit.
- ✧ Do not copy: Does not conduct the copy process for generated _E.FCNT as well as the setting process for synthesis options.
- ✧ Specifies by dialog box for every synthesis: Whenever _E.FCNT is generated it inquires with a dialog box whether to copy or not.
- Synthesis history settings
 - Save synthesis result (IFF file): Specifies whether to save or not the IFF file at the time of synthesis execution.
 - Save synthesis environment for every synthesis: Whenever synthesis is implemented, it specifies whether to save the synthesis results as history or not.
- Error file display at time of abnormal exit of subsystem
 - Displays error file at the time of abnormal exit of subsystem: When this is enabled, in case each program opened from GUI is abnormally completed, error file is automatically displayed in editor. When it is disabled, error file is not displayed.

16.1.7. Tool path tab

It specifies the path of various external tools used in CWB-gui.

- User definition option file: In the behavioral synthesis option settings screen (section 9), specifications of options not registered in CWB-gui are described. Loaded options with this tab can be specified by clicking "user specification" button in "Macro (Optimization level)" tab of behavioral synthesis option setting screen.

Formatting of user specification option file

```
bdltrn{
    -option name: comment etc
    -option name: #: in case of one argument "default value"
    -option name: # #: In case of two arguments [default value, default
    value]
}
```

Example

```
bdltrn{
    -option1 : test option 1
    -option2 : test option 2
    -option3:# : test option 3(default)[foo]
    -option4:# # : test option 4(default)[foo,bar]
}
```


- PDF file viewer: PDF file in "Cyber reference manual" from "Help" can be opened but, the Pdf viewer used at this time is specified. In case of windows, by default programs with .pdf extension will start.
- Waveform viewer: Selects waveform viewer used in properties verification and specifies the binary for execution.

16.2.Attributes Palette

Detailed synthesis control can be performed by writing attributes in the source files. Attributes palette is a functionality that supports the input of attributes (Figure 16-2). In the Attributes palette, description can be done for the desired attribute in any location by selecting the attribute that needs to be described in the source file within attributes.

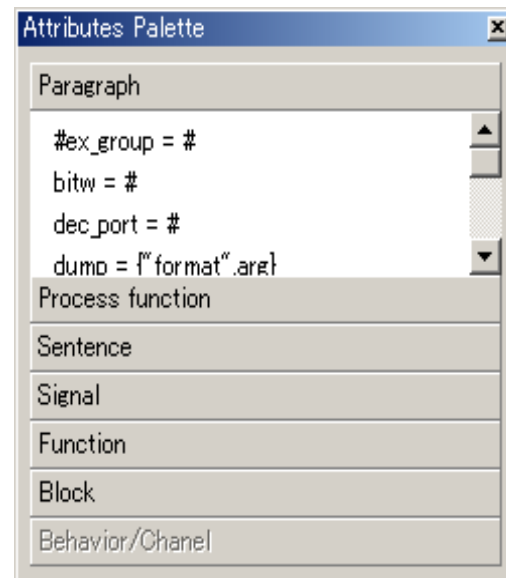


Figure 16-2 Attributes Palette

Starting method

The screen shown in Figure 16-2 opens when you select "Attributes palette" from context menu of opened source file or, you click "Attributes palette" button on the tool bar.

Operation method

When attribute that needs to be entered is selected, the subwindow pops up. In the subwindow, as per the requirement when "Insert" button is pressed after argument settings are implemented, specified attributes are provided in the source file opened in active internal text editor.

Notes

File entered in include or src of project is extracted to internal text editor, and insertion of attributes is effective only when it is active.

16.3.Error filter

Error items displayed in error table (section 11.4) of synthesis support window can be controlled by error filter (Figure 16-3).

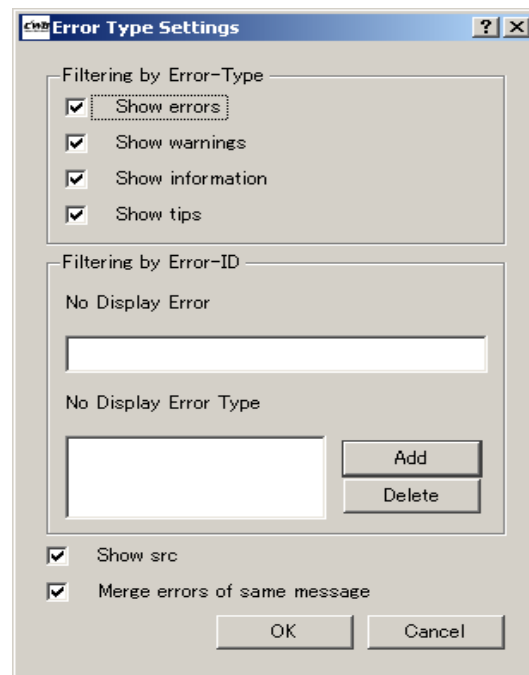


Figure 16-3 Error filter

Start of error filter

Error filter can be started by any of the following ways.

- 1) Select "error filter" from drop-down menu "Synthesis"
- 2) Right click at the label of error table in synthesis support window

Operations

- Classification as per error type
 - Show errors: When it is disabled, the message applicable for error ID that starts with "E_BT" in error table is not displayed.
 - Show warnings: When it is disabled, the message applicable for error ID that starts with "W_BT" in error table is not displayed.
 - Show information: When it is disabled, the message applicable for error ID that

- starts with "I_BT" in error table, is not displayed.
- Show tips: When it is disabled, the message applicable for error ID that starts with "H_BT" in error table, is not displayed.
- Classification as per error ID: Error message corresponding to error ID that contains any one of the rows included in "No display error type" as character string, is not displayed.
 - When characters are entered in "No display error" and "Add" button is pressed, then the characters added to "No display error type" are registered.
 - When "Delete" button is pressed for the selected row in "No display error type", the applicable row is deleted.
- Show src: When it is disabled, error messages below the statements starting with " 【src row】 " are not displayed.
- Compile the similar error messages: When it is enabled, error messages are displayed by allocating one row in the error table for all error messages. If it is enabled and in case there are statements starting with " 【src row】 " and error ID, then for all similar error messages before that, are compiled and they are displayed by assigning one row in the error table. At this time, for this compiled row, the rows corresponding to " 【src row】 " of all messages before compiling, are displayed collectively.

17. Command list

There is a command list in CWB-gui for context menu of Drop down menu, Toolbar and Project Management window. In this section, the numeric figure inside " () " shows the chapter number and section number in this document.

17.1. Drop down menu

17.1.1. File

- **New:** Creates new workspace (4.3.1), project (4.3.2), file (4.3.3), simulation scenario.
- **Open:**
 - Opens the workspace (4.3.1) and expands it in Project Management window.
 - Opens file (4.3.3) in internal editor.
- **Close:** Closes the file opened in internal editor (4.3.3).
- **Save:** Saves the file opened in internal editor (4.3.3).
- **Save All:** Saves all the open workspaces, projects and files in case they are unsaved.
- **Save as:** Saves the text file opened in internal editor with a different name (4.3.3).
- **Close workspace:** Releases the workspace opened in Project Management window (4.3.1).
- **Save selected project:** For workspace opened in Project Management window, the created workspaces and projects can be saved by selecting them one by one.
- **Print:** Prints the text file opened in internal editor.
- **Recently opened workspaces:** Workspace opened from history of Recently opened workspaces can be selected (4.3.1).
- **Recently opened files:** File opened from history of Recently opened files can be selected (4.3.3).
- **Exit:** Exits CWB-gui.

17.1.2. Edit

- **Undo:** Undos only one edited operation in internal editor.
- **Redo:** In case undo is done for an operation in internal editor, then it returns only one operation back to the state before undo is done.
- **Cut:** Removes the selected part from the internal editor and saves it to the clipboard.
- **Copy:** Saves the selected part in the internal editor to clipboard.
- **Paste:** Pastes the contents saved in clipboard to the internal editor.
- **Find:** Searches the specified contents in internal editor, signal table etc.
- **Replace:** Replaces the contents in the internal editor.
- **Select all:** Selects all the content in the internal editor.

17.1.3. Display

- **Project Management window:** View/hide the Project Management window.
- **Browser window:** View/hide the browser window.
- **Console window:** View/hide the console window.
- **Command palette :**View/hide the display of command palette
- **Attributes palette:** Starts the palette for supporting attributes input (16.2).
- **Standard toolbar:** View/hide the standard toolbar.
- **Operations toolbar:** View/hide Operations toolbar.
- **Reload:** Text file opened in internal editor is read again and it is reloaded to internal editor.

17.1.4. Architecture design

- **Bus interface circuit generation:** Bus interface circuit generation program cybus is executed for the project (13.1).
- **On-chip bus automatic generation:** Onchip bus automatic generation program CybusM is executed for the project (13.2).
- **Top module generation:** Top module generation program topmodgen is executed for process and project (14).

17.1.5. Synthesis

- **Analysis: Executes** BDL analysis program (7.1) /SystemC analysis program scpars (7.2) for source files. The program to be executed is decided by the extension of the source file to be used. In case of RTL project, rtl analysis program rtlpars (7.3) can be executed for the src folder.
- **Re-analysis:** When the analysis is successful once, and then if analysis is done again without changing source file, include file and analysis options, normally the analysis is skipped however, this command forcefully executes analysis without skipping.
- **Analyze all source files:** Executes analysis for all source code files in a work space for which analysis is not done.
- **Re-analyze all source files:** Executes re-analysis for all source code files in a work space. It also executes re-analysis for the source files for which analysis is complete.
- **Constraint file generation:**
 - **Functional unit library/Constraint automatic generation:** Automatically generates FLIB/FCNT for process (8.3.1).
 - **Memory library/constraint automatic generation:** Automatically generates MLIB/MCNT for the process (8.5).
- **Synthesis:** Executes behavioral synthesis program bdltran for the process (9).
- **RTL generation:** Executes Verilog generation program veriloggen (10.1) /VHDL generation program vhdngen (10.2) for the process. The program to be executed is decided by the value in the RTL selection combo box of Operations toolbar
- **Generate all RTL:** Executes the processes till RTL generation for all processes for behavioral synthesis registered in the workspace.

- **RTL regeneration:** When analysis is done successfully once and if RTL is generated again without changing RTL generation option, synthesis option, constraint files, analysis option, include file, source file etc., then usually RTL generation is skipped, however, this command forcefully executes RTL generation, synthesis and analysis without skipping.
- **Analysis options:** Specifies options of BDL analysis program bdlpars (7.1)/SystemC analysis program scpars (7.2) for the source files. The option settings screen of the program to be started is decided by the extension of the source file to be used.
- **Synthesis options:** Specifies options of behavioral synthesis program bdltran for the process (9).
- **Synthesis result display:**
 - Synthesis support window: Synthesis support window is started for the process (0).
 - Error table: Displays error list of synthesis (11.4).
 - Graph: Synthesis history is displayed by graph (11.3).
 - Result history: Synthesis history is displayed in table (11.6).
- **RTL generation options:** Specifies options of Verilog generation program veriloggen (10.1)/VHDL generation program vhdngen (10.2) for the process. The program to be started is decided by the value in the RTL selection combo box of Operations toolbar.

17.1.6. Tools

- Import default options: Imports default option settings for bdlpars, scpars, bdltran, veriloggen, vhdngen.
- Default option settings: Conducts default option settings for bdlpars, scpars, bdltran, veriloggen, vhdngen (5).
- Export default options: Exports of default option settings for bdlpars, scpars, bdltran, veriloggen, vhdngen.
- **Options:** Conducts Tools option settings of CWB-gui (16.1).
- **Console:**
 - **Delete:** Clears console.
 - **Save:** Saves display contents of console in file
- **Stop:** Stops the executed program in CWB-gui.

17.1.7. Window

- **Cascade:** Cascades the windows displayed in work screen.
- **Arrange Tiled:** Shows the windows displayed in work screen by arranging them.
- **Close all windows:** Closes all the windows displayed in work screen.
- **Arrange Horizontal:** Shows the windows displayed in work screen by arranging them horizontally.
- **Arrange Vertical:** Shows the windows displayed in work screen by arranging them vertically.
- **Initialization:** Returns the screen size as well as location of toolbar and various

windows such as Project management window to their initial state.

- **Process wise display:** You can change the display of the summarized files of process folder.
- **Display in another window:** Shows windows displayed in work screen by detaching them to another window.
- **Display window in work screen:** Returns the detached window to work screen.

17.1.8. Help

- **Help:** Displays Help of each type of sub program.
- **Attributes:** Displays list of attributes that can be specified in BDL.
- **Restrictions information of system:** Displays restriction information of system in behavioral synthesis.
- **Open manual:** Opens manual of each type of subsystem. The manual placed under \${CYBERDIR}/doc is to be opened. In order to use this functionality, path settings of pdf file viewer in tool options should be implemented (16.1.7).
- **Version information of CWB:** Shows version information of CWB-gui.
- **Version information of each program:** Shows version information of each program in CWB toolkit.

17.2. Toolbar

17.2.1. General Toolbar

- **New:** Creates new workspace (4.3.1), project (4.3.2), file (4.3.3), simulation scenario etc.
- **Open :**
 - Opens the workspace (4.3.1) and extracts it to the Project Management window.
 - Opens file (4.3.3) in internal editor.
- **Save file:** Saves the opened text file in the internal editor (4.3.3).
- **Save all files and projects:** Saves all workspaces, projects and files opened in Project Management window in case they are unsaved.
- **Undo:** Returns only one edited operation to the state before editing in internal editor.
- **Redo:** In case undo is done for an operation in internal editor, then it returns only one operation back to the state before undo is done.
- **Cut:** Removes the selected part from the internal editor and saves it to the clipboard.
- **Copy:** Saves the selected part in the internal editor to clipboard.
- **Paste:** Pastes the contents saved in clipboard to the internal editor.
- **Project Management window:** View/hides Project Management window.
- **Browser window:** View/hides browser window.
- **Console window:** View/hides Console window.
- **Command palette:** View/hide the display of command palette.

- **Attributes palette:** View/hides Attributes palette.
- **Print:** Prints the text file opened in internal editor.

17.2.2. Operations toolbar

- **C compiling of BDL:** Compiles BDL by C compile.
- **Analysis:** Executes BDL analysis program bdlpars (7.1)/SystemC analysis program scpars (7.2) for source files. The program to be executed is decided by the extension of the source file to be used. In case of RTL project, rtl analysis program rtlpars (7.3) can be executed for src folder.
- **Synthesis:** Executes behavioral synthesis program bdltran for process (9).
- **RTL generation:** Executes Verilog generation program veriloggen (10.1)/VHDL generation program vhdngen (10.2) for the process. The program to be executed is decided by the value of RTL selection combo box of Operations toolbar.
- **RTL regeneration:** When analysis is successfully done once, and if RTL is generated again without changing source file, include file, analysis options, constraint files, synthesis options then usually RTL generation is skipped, however, this command forcefully executes analysis, synthesis, RTL generation without skipping.
- **Synthesis options:** Specifies options of behavioral synthesis program bdltran for process (9).
- **Synthesis support window:** Starts synthesis support window for process (0).
- **Save synthesis results:** Manually saves the history of synthesis results for the process. This command is valid only in case it is specified for synthesis environment at the time of synthesis that do not save automatically (11.7.1, 16.1.6) .
- **Model generation:** Executes behavioral level C++ model generation program bmcppgen()/ Cycle level C++ model generation program bmcppgen ()/Cycle level SystemC model generation program cmscgen for simulation scenario. The program to be executed depends on the value of the model language and simulation level of simulation scenario to be executed.
- **Makefile generation:** Executes Makefile generation program mkmfmsim for simulation scenario.
- **Compile:** Compiles model for simulation scenario.
- **Execute:** Executes compiled model for simulation scenario. In case simulation level is RTL and when testbench generation program generates the script for execution, this command can execute RTL simulation.
- **Stop:** Stops the executed program in CWB-gui.
- **RTL selection:** When operations related to RTL generation are implemented, RTL to be generated is selected from Verilog or VHDL (10).
- **Jump selection:** Selects highlighted target linked with source file (12.1).

17.3. Project Management context menu

17.3.1. Workspace

- **New:** Creates new project (4.3.2), file (4.3.3), simulation scenario etc.
- **Import:** Imports file (4.3.3) or project (4.3.1, 4.3.2). For source file, include file, and testbench it can be specified that whether the import will be done by copy or it will be done by reference (refer file to be imported at the time of file reference in CWB-gui) of full path or relative path reference. Type of file that can be imported is different as per the selected icon in "import destination".
 - Workspace: project
 - Intermediate project: All files, Intermediate project, RTL project
 - RTL project: All files
 - Src folder : Source files
 - Include folder : Include files
 - Testbench folder : Testbench
 - Process folder: All files
 - Simulation scenario folder: All files
 - Library folder: Library files
 - Synthesis constraint folder: Constraint files
- **Import hierarchy project:** Project under any directory in file system is imported as project of CWB-gui (4.3.4).
- **Close workspace:** Releases the workspace opened in Project Management window (4.3.1).
- **Save selected project:** For workspace opened in Project Management window, the created workspaces and projects can be saved by selecting them one by one.
- **Export RTL:** Exports RTL generated by RTL generation and script file generated by automatic generation of logical synthesis script (10.4).
- **Clear Trash:** All files deleted in CWB-gui move to the Trash directory under the directory corresponding to the workspace. This command deletes all the files in the Trash directory.
- **Delete synthesis history:** Deletes history of behavioral synthesis (11.7.2) . Locked history becomes out of scope.

17.3.2. Intermediate project – RTL project

- **Basic library automatic generation:** Executes program BLIBgen that automatically generates Basic library (BLIB) (8.2).
- **Top module generation** (only Intermediate project) : Executes top module generation program tomodgen (14).
- **Bus interface circuit generation** (only Intermediate project) : Executes bus interface circuit generation program cybus (13.1).
- **On-chip bus automatic generation** (only Intermediate project) : Executes on-chip bus automatic generation program CybusM (13.2).
- **Select all processes:** Select all processes under the applicable project
- **New:** Creates new project (4.3.2), file (4.3.3), simulation scenario etc.
- **Import:** Imports file (4.3.3) or project (4.3.1, 4.3.2). For source file, include file, and

testbench file it can be specified that whether the import will be done by copy or import will done by reference (refer file to be imported at the time of file reference in CWB-gui) of full path or relative path reference. Type of file that can be imported is different as per the selected icon in "import destination".

- Workspace: Project
- Intermediate projects: All files, intermediate projects and RTL projects
- RTL project: All files
- Src folder: Source file
- Include folder: Include file
- Testbench folder: Testbench
- Process folder: All files
- Simulation scenario folder: All files
- Library folder: Library file
- Synthesis constraints folder: Constraint file
- **Import hierarchy project** (Only intermediate project) : Project under any directory in file system is imported as project of CWB-gui (4.3.4).
- **Save:** Saves the project file of project to be used.
- **Detach from project:** Removes the project from management of workspace. After implementation, the target project is not displayed in Project Management window (4.3.2).
- **Delete: Deletes project.** The deleted project is moved to Trash directory under directory corresponding to the workspace (4.3.2).
- **Move project:** Moves project under workspace or other intermediate project (4.3.2).
- **Link of project:** Places the link of project under other intermediate project or RTL project (4.3.2).
- **Export project:** Copies the project under directory corresponding to specified project to any location (4.3.2).
- **Delete synthesis history:** Deletes history of behavioral synthesis (11.7.2) . Locked history becomes out of scope.
- **Export RTL:** Exports RTL generated by RTL generation and script file generated by automatic generation of logical synthesis script (10.4).
- **Do not show hidden attributes file:** View/hides the source file and include file that have hidden attributes (4.3.2).
- **Properties:** Specifies standard FLIB, standard BLIB and clock cycle provided as synthesis options of process under project (8.1.2).

17.3.3. src, include, testbench

- **New:** Creates new project (4.3.2), file (4.3.3), simulation scenario etc.
- **Import:** Imports file (4.3.3) or project (4.3.1, 4.3.2). For source file, include file, and testbench it can be specified that whether the import will be done by copy or it will

be done by reference (refer file to be imported at the time of file reference in CWB-gui) of full path or relative path reference. Type of file that can be imported is different as per the selected icon in "import destination".

- Workspace: Project
- Intermediate project: All files, Intermediate projectors project
- RTL project: All files
- Src folder: Source files
- Include folder: Include files
- Testbench folder: Testbench
- Process folder : All files
- Simulation scenario folder: All files
- Library folder: Library files
- Synthesis constraint folder: Constraint file
- **Do not show hidden attributes file:** View/hides the source file and include file that have hidden attributes (4.3.2).
- **Analysis** (Only RTL project src) : Executes RTL analysis program rtlpars (7.3) .
- **Re-analysis** (Only RTL project src) : Analysis is done again with same parameters as at the time when RTL analysis program is executed in the last.

17.3.4. Process

- **New:** Creates new project (4.3.2), file (4.3.3), simulation scenario etc.
- **Import:** Imports file (4.3.3) or project (4.3.1, 4.3.2). For source file, include file, and testbench it can be specified that whether the import will be done by copy or it will be done by reference (refer file to be imported at the time of file reference in CWB-gui) of full path or relative path reference. Type of file that can be imported is different as per the selected icon in "import destination".
 - Workspace: project
 - Intermediate project: All files, Intermediate project, RTL project
 - RTL project: All files
 - Src folder : Source files
 - Include folder: include files
 - Testbench folder: testbench
 - Process folder: All files
 - Simulation scenario folder: All files
 - Library folder: Library files
 - Synthesis constraint folder: Constraint files
- **Basic library automatic generation:** Executes program BLIBgen that automatically generates Basic library (BLIB) (8.2).
- **Constraint files generation:**
 - **Functional unit library/Constraint automatic generation:** Automatically generates FLIB/FCNT for process (8.3.1).
 - **Memory library/constraint automatic generation:** Automatically generates MLIB/MCNT for the process (8.5).
- **Synthesis:** Executes behavioral synthesis program bdltran for the process (9).

- **RTL generation:** Executes Verilog generation program veriloggen (10.1) /VHDL generation program vhdngen (10.2). The program to be executed is decided by the value in the RTL selection combo box of Operations toolbar.
- **RTL regeneration:** When analysis is done successfully once, and if RTL is generated again without changing RTL generation option, synthesis option, constraint files, analysis options, include file, source file etc., then usually RTL generation is skipped, however, this command forcefully executes RTL generation, synthesis and analysis without skipping.
- **Stop:** Stops the executed program in CWB-gui.
- **Top module generation** (only Intermediate project) : Executes top module generation program tomodgen (14).
- **Logical synthesis script automatic generation:** Executes logical synthesis script automatic generation program LSscrgen (10.3).
- **Overflow check:** Checks if there is no description that conflicts with overflow and underflow etc in the source file (7.5).
- **Signal state handling table:** After synthesis, displays the table that shows that which signal is accessed by which state (12.3).
- **Data path display:** Refers the circuit generated by synthesis as data path diagram (12.5.1).
- **Circuit properties:** Opens circuit properties file(QoR.HTML).
- **Source file editing:** Opens source file of that process generation.
- **Loading delay report:** Conducts the handling with source file by loading delay report of DesignCompiler (12.4) .
- **Synthesis results display:**
 - Synthesis support window: Start Synthesis support window (0) for process.
 - Error table: Display error list table of synthesis (11.4).
 - Graph: Display synthesis history in graph (11.3) .
 - Results history: Display synthesis history in table (11.6) .
- **Save synthesis results:** Save history of synthesis result manually. This command is valid only when it is specified that do not save synthesis environment automatically at the time of synthesis (11.7.1, 16.1.6).
- **Delete synthesis history:** Deletes history of behavioral synthesis (11.7.2). Locked history is out of scope.
- **Synthesis option:** Specifies the options of behavioral synthesis program bdltran (9).
- **RTL generation option:** Specifies Verilog generation program veriloggen (10.1)/VHDL generation program vhdngen (10.2). The option setting screen of the program to be started is decided by the value of RTL selection combo box of Operations toolbar.
- **Open error file:**
 - **Overflow check:** Opens error file of overflow checker (7.5).
 - **Synthesis:** Opens error file of synthesis bdltran (9.2).
- **Generate simulation scenario:**Generates simulation scenario in order to execute

simulation for source file and process.

- **Export RTL:** Exports RTL generated by RTL generation as well as script file generated by logical synthesis script automatic generation (10.4).
- **Export LOG:** Exports LOG file generated by behavioral synthesis (15.2).

17.3.5. Simulation scenario

- **New:** Creates new project (4.3.2), file (4.3.3), simulation scenario etc.
- **Import:** Imports file (4.3.3) or project (4.3.1, 4.3.2). For source file, include file, and testbench file it can be specified that whether the import will be done by copy or import will done by reference (refer file to be imported at the time of file reference in CWB-gui) of full path or relative path reference. Type of file that can be imported is different as per the selected icon in "import destination".
 - Workspace: project
 - Intermediate project: All files, Intermediate project, RTL project
 - RTL project: All files
 - Src folder : Source files
 - Include folder: include files
 - Testbench folder: testbench
 - Process folder: All files
 - Simulation scenario folder: All files
 - Library folder: Library files
 - Synthesis constraint folder: Constraint files
- **C compile (only source files):** BDL is executed by compiling with C compile.
- **Testbench template generation:** Generates the sample of testbench used in behavioral level C++ simulation. It cannot be started from simulation scenario other than that of behavioral level C++.
- **Model generation:** Executes behavioral level C++ model generation program bmcppgen/Cycle level C++ model generation program cmcppgen /Cycle level SystemC model generation program cmscgen. The program to be executed depends on the value of model language and simulation level of the simulation scenario to be executed.
- **Makefile generation:** Executes Makefile generation program mkmfmsim.
- **Testbench generation:** Executes RTL testbench generation program tbgen for simulation scenario in which simulation level is RTL.
- **Compile:** Compiles model).
- **Execute:** Executes compiled model. In case simulation level is RTL, and when testbench generation program generates script for execution, then RTL simulation can be executed by this command.
- **Model generation option:** Specifies the options of behavioral level C++ model generation program bmcppgen/Cycle level SystemC model generation program cmscgen/ Makefile generation program mkmfmsim. The option settings screen of the program to be started depends on the value of the model language and simulation level of simulation scenario to be executed.
- **Makefile generation option:** Specifies the options of Makefile generation program

mkmfsim for simulation scenario.

- **Testbench generation option:** Specifies the options of RTL testbench generation program tbgen for simulation scenario in which simulation level is RTL.
- **Exection option:** Specifies option settings at the time of simulation.
- **Power estimation:** Power analysis is done by PowerTheater for simulation scenario where the simulation level is RTL.
- **Export simulation environment:** Copies simulation scenario to any location in external part of workspace.
- **Open error file at the time of compiling:** Opens the error file at the time when model is compiled.
- .
- **Delete:** Deletes simulation scenario. The deleted simulation scenario is moved to the Trash directory under the directory corresponding to the workspace.
- **Properties:** Specifies the properties of simulation scenario.

17.3.6. File

- **Open:** Opens file in editor (4.3.3) . It depends on the settings of tool options that whether the file will be opened in internal editor or external editor (16.1.3). The editor will be different as per the type of opened file.
 - Constraint files: Editor of constraint files
 - .IFF file: bdl drawing program bdldraw
 - .QOR.HTML file: HTML browser
 - .dty file: graphviz
 - Other text file: Internal/external text editor
 - Other binary file: Cannot be opened.
- **Open internal text editor:** Opens file in internal text editor. Binary file cannot be opened.
- **Open in external text editor :** Opens the file in external text editor. Binary file cannot be opened. In using the external editor, path of external editor should be specified in tool options (16.1.3).
- **Close:** Closes the file opened file in internal editor (4.3.3).
- **Save:** Saves file (4.3.3).
- **Delete:** Deletes file. The deleted file is moved to Trash directory under directory corresponding to workspace.
- **Detach from project:** Removes the file from management of workspace. After implementation, the target file is no longer displayed in Project Management window (4.3.3).
- **Print:** Prints the text file opened in internal editor.
- **Analysis** (only source files) : Executes BDL analysis program bdlpars (7.1) /SystemC analysis program scpars (7.2) for source files. The program to be executed is decided by the extension of the source file to be used.
- **Re-analysis** (only source files) : When analysis is done successfully once, and if re-analysis is done without changing source file, include file and analysis options etc. then usually analysis is skipped but this command forcefully executes analysis

without skipping.

- **Analysis options (source files only):** Specifies options of BDL analysis program bdlpars (7.1)/SystemC analysis program scpars(7.2). The option setting screen of the program to be started is decided by the extension of source file to be used.
- **State change diagram display** (only source files) : Displays handling information of row and state as well as state change of the specified process in internal editor for the source files (12.2).
- **Open error file:** Opens error file at the time of implementing analysis (7.1, 7.2, 7.3).
- **Generate simulation scenario:** Generates simulation scenario in order to execute simulation for source file and process.
- **Export file:** Exports file.
- **Properties:** Refers information of file. Hidden attributes can be provided for source files and include files (4.3.3).
- **BDL generation** (only .IFF file) : Executes program iff2bdl that generates equivalent BDL description for IFF file.
- **Data path display** (only _E.IFF file) : Refers circuit generated through synthesis as the data path diagram (12.5.1).
- **LOG file extraction** (Only .LOG file of "Other files" folder) : Extracts LOG file and uses the synthesis environment of the time of LOG file generation again (15.1).

18.Index

A	
Attributes palette	127
Automatic generation of Basic library file (BLIB)	63
Automatic generation of Functional Unit Library file (FLIB)/Functional Unit Count file (FCNT)	64
Automatic generation of Memory library/constraint file (MLIB/MCNT)	67
B	
Back up file	39, 43
Bar chart	84
Basic library	60
bdef	113, 114
BDL analysis	52
BDL viewer	109
bdlldraw	109
bdlpars	53
bdltran	71
Behavioral synthesis	71
BLIB	60
BLIBgen	63
Bus definition file	113
Bus interface circuit generation	112
C	
Circuit quality	88, 110
Console window	16
Converting function into a functional unit	54, 71
CSV file editor	125
cybus	112
CybusM	112
D	
Data path viewer	108
defmod	73
Delay report	106
Delete history	92
E	
Editor	124
Environment Variables settings	123
Error filter	128
Error table	88
External text editor	10, 125
F	
False path	107
FCNT	60
File	32
File import	32
FLIB	60
FLIBgen	67, 70
Font	123
Functional unit constraint file	60
Functional unit count file	60
Functional unit library file	60
G	
General tool bar	13
Generate Top Module	117
Generation of logical synthesis script	77
H	
Hidden attributes	32, 34, 113
I	
Include file	23
Intermediate project	18, 29

L

Latencies	87
Latency table	87
License acquisition	123
Link	30, 40
LMT	60
LOG file	121
LSscrgen	77

M

MCNT	60, 120
Memory constraint file	60
Memory count file	60
Memory library file	60
MLIB	60, 120
MLMT	60

O

On-chip bus automatic generation	112
Operations toolbar	14, 75, 76, 94, 95, 96
Overflow check	58

P

Pathdraw	108
<u>Pdf viewer</u>	10, 127
PLMT	60
plot chart	80
Port constraint file	60
Port related file	60
PREL	60
Process	24
Process folder	19
Project export	31
Project import	31
Project import	28
Project management window	15
<u>Project properties</u>	61
Pull down menu	12

Q

QoR	88
-----------	----

R

Restore results	92
RTL analysis	56
RTL project	19, 22, 29
rtlpars	56

S

Save history	91
scpars	55
Shortcut key	124
Signal state handling table	96, 98, 99
Source file	23
Source files analysis	52
Standard BLIB	61, 112
Standard FLIB	61, 65, 112
Synthesis support window	79
SystemC analysis	55

T

Testbench	24
Tool Bar	13
topmodgen	117

U

User definition option file	126
-----------------------------------	-----

V

Verilog generation	75
veriloggen	75
VHDL generation	76
vhdlgen	76
Virtual folder	19

W

Workspace..... 18, 19, 27

Z

-Zflib_fcmt_out..... 64
-Zmlmt_out 67