Model User Manual

Top Module Generation Script of RESL-X

(V1.0)

Summary

This document is used to describe in detail the method to use the Top Module Generation script to generate the top module of RESL-X platform.

Reference:

Table of Contents

Index of Tables

Index of Figures

1. Introduction

- The purpose of this document is to describe the method to use the Top Module Generation script to generate the top module of RESL-X platform.
- Operation condition of Register IF Generator script:

Content	Condition
Linux	Redhat Enterprise 3.2.3
Python	3.1.2 or higher

2. Tool summary

2.1. Overview

- Top Module Generation script is used to create a top module for RESL-X platform. This script uses the configuration file, model info files, and skeleton files to create the top module file and some related files. Top module file contains the declaration of all models, buses and the connections among them. The related files are the handle command file, the Makefile and the main file (storing sc_main function).

2.2. Block Diagram

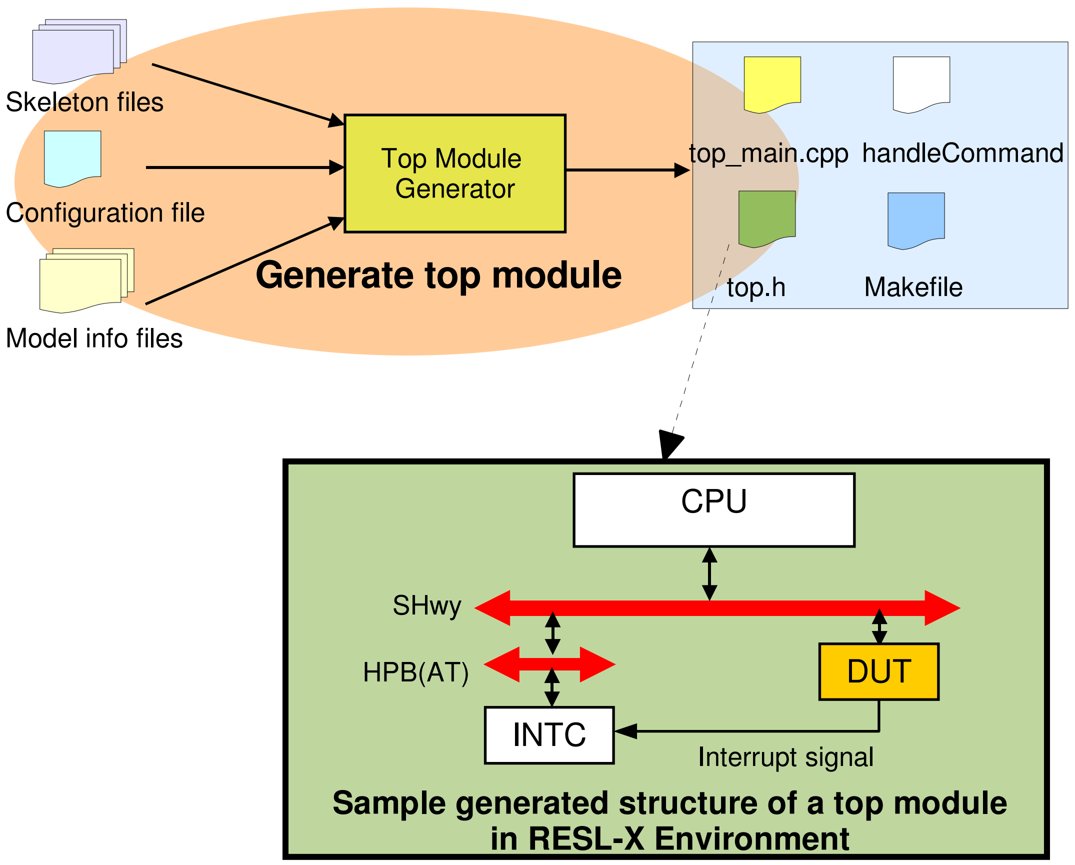


Figure 2.1: Top Module Generation tool

Explanation:

- This script will generate top module, handleCommand, and related files for compilation and simulation.
- Users will modify the configuration file, model info files, and skeleton files then invoke the Top Module Generation script to generate four expected files:
  - top.h: The top module file contains the declarations of buses, models and the connections among them.
  - top_main.cpp: The main file contains the sc_main function that instantiates the top module and clock signals.
  - Makefile: The Makefile is used for compilation.
  - handleCommand: Configuration file is used for handleCommand method.
- Skeleton files include top module skeleton (top.h.skl), main method skeletons (top_main.cpp.skl), and Makefile.skl.

2.3. Supported features

Table 2.1: List of implemented functions of the tool.

No.	Function	Description
1	TLM LT I/F	Using TLM sockets to connect buses and initiator/target modules.
2	Multiple buses	Support generating the environment with multiple buses with different bus widths.
3	Interrupt signal connection	Generate connection of interrupt signals
4	Connect non-connection port and stub	Support generating connection of non-connection ports and stub
5	Clock connection	Support clock declarations and connections to the top model which has clock ports
6	handleCommand generation	Support generating handleCommand file

3. Scripts usage

3.1. File structure

- The Top Module Generation script is a Python script. The input files include three skeleton files, configuration file and model info files. They are shown in Table 3.2. The output files are described in section 3.4.
- The detailed usage of *top_gen.py* script is described in table 3.1.

3.2. Syntax

Table 3.1: Usage of top_gen.py

Type	Content
Script name	*top_gen.py* **(*1)**
Language	Python (version 3.1.2 or higher)
Purpose	This script is used to generate top module, Makefile and handleCommand file
Input file	*top.sci* – Top Module Configuration Information file stored the top module specification.
Output file	*top.h* – top module header file. *top_main.cpp* – top module main file. *HandleCommand* – handleCommand file. *Makefile* – Makefile for environment compilation.
Usage	python *top_gen.py* -[option]
Option	**--version** : show program's version number and exit **-h, --help** : show this help message and exit **-i** *<path>*, **--inputpath=***<path>* : input path for skeleton and module info files. **-o** *<path>*, **--outputpath=***<path>* : path used to store the output files. **-s** *<file>*, **--sci=***<file>* : filename of SCI file.

Notes:

(*1) Script files include 2 files:

•**gen_top_class.py**: Define Python classes which are used in the Generator main function.

•**gen_top.py:** Main function of the Generator.

3.3. Input

– When using this script, users need to modify a configuration file to specify the expected top module. Beside that, users must create model info files to define the information of each model.

Table 3.2: List of needed files for Top Module Generation script.

No.	Filename	Description
1	*top_gen.py*	Main Python script used to generate the expected SIM environment.
2	*top_gen_class.py*	Define python class which is use by in generator main function
3	*top.h.skl*	Skeleton file used for forest top module generation.
4	*top_main.cpp.skl*	Skeleton files used for main function generation.
5	*Makefile.skl*	Skeleton file used for making Makefile.
6	Configuration file (*top.sci*)	Configuration file describes bus, model , clock and connection information
7	Model info files	These files contain information file of each model (include bus model) : model definition, template argument definition and port definition. The name of model info files are declared in *MODULE_LIST* section of the configuration file.

3.3.1. Input files format

– A comment line begins with "#".

– A description line begins with a control keyword. Control keyword started with a number of "%". Keywords start with % are control keywords which are used in configuration file (*top.sci*, model info files)

– Keywords started with %% are notations which are used in skeleton files (*top.h.skl, top_main.cpp.skl, Makefile.skl*). The generated content will replaced each corresponding notation.

– After the control keyword, description lines contain arguments which describe for each control keyword. Determined by information of control keyword, the order of arguments can change flexibly based on the information of control keywords.

– Arguments in a description line are separated by space(s) or tab(s).

3.3.2. Configuration file

– Configuration file is used to describe all the environment. This file includes two main parts:

  • Part 1: Environment descriptions: define common information for all the environment.

  • Part 2: Buses descriptions: to define information for each bus. If multiple buses are used, this part will be repeated many times but information for each bus is different from each other.

– The figure 3.1 describes the structure of this file.

```
################################################################
## Part 1: Environment descriptions
################################################################
%DEFINE
%ENV_INFO
%CLOCK_RATIO
%CHANNEL_SYS
%MODULE_LIST
%VC_SPECIFIC_SYS
%CLOCK_PARAMETER
```

```
################################################################
## Part 2: Bus descriptions
################################################################
%BUS_INFO # for indicate Bus ID in multi buses.
%TOP_MODULE_INI_LIST
%TOP_MODULE_TGT_LIST
%ADDRESS_MAP
%SET_ARB
%SET_INIGRP
%SET_TGTGRP
```

Figure 3.1: The structures of SCI file.

– All information is marked by keywords. The meaning of keywords in this file is described in table 3.3.

Table 3.3: List of keywords in SCI file

No.	Keyword	Meaning	Mandatory
1	%DEFINE	Information of define macro	No
2	%ENV_INFO	Information of coding style (LT or AT) for handleCommand, and information of added external TLM socket to a top module	Yes
3	%CLOCK_RATIO	Clock ratio: name, cycle, duty cycle, start time, positive first or negative first	No
4	%CHANNEL_SYS	Signals information: signal's name, bit width, signal or port in or port out, type	No
5	%MODULE_LIST	List Information of modules in environment : module info file name, class name and instance name.	Yes
6	%VC_SPECIFIC_SYS	Specific system of each module: its names and its signal names.	No
7	%CLOCK_PARAMETER	Clock information for handleCommand (the model which doesn't have clock terminal and has a clock parameter as an object)	No
8	%BUS_INFO	Information of bus: bus instance's name.	Yes
9	%TOP_MODULE_INI_LIST	List information of initiator modules of bus: instance name, socket name.	Yes
10	%TOP_MODULE_TGT_LIST	List information of target modules of bus: instance name, socket name.	Yes
11	%ADDRESS_MAP	Memory range of target model. It includes: start address, end address and module instance's name.	No
12	%SET_ARB	Arbitration information for bus	No
13	%SET_INIGRP	Initiator group information of arbitration	No
14	%SET_TGTGRP	Target group information of arbitration	No

3.3.2.1. Detailed description

3.3.2.1.1. Environment information

- **%DEFINE**: Define the macro. The information in the next line is:

  <Macro> <value>

  - *<Macro>* : Name of macro.
  - *<value>* : Value of macro.

  If there are multiple macros, many lines with the same format will be defined.
- **%ENV_INFO**: Define the information of environment. The information in the next line is:

  <TLM_coding_style> <add_TLM_socket_ini/tgt/both> <ini_buswidth> <tgt_buswidth>

  - *<TLM_coding_style>*: String to choose TLM/LT or TLM/AT coding style.
    - ✔ LT: TLM/LT (Loosely-Timed) Environment

    - ✔ AT: TLM/AT (Approximately-Timed) Environment. This coding style AT is not supported.
  - *<add_TLM_socket_ini/tgt/both>*: String to add external TLM socket to a top module
    - ✔ ADD_INI_SOCKET: add initiator socket.
    - ✔ ADD_TGT_SOCKET: add target socket.
    - ✔ ADD_BOTH_SOCKET: add initiator and target socket.
  - *<ini_buswidth>* : String to define the buswidth of external initiator TLM socket.
  - *<tgt_buswidth>* : String to define the buswidth of external target TLM socket.
- **%CLOCK_RATIO**: Setting attributes of clock like. The information in the next line(s) is:

  <clock_name> <period> <duty_cycle> <start_time> <positive_first>

  - *<clock_name>* : String to define name of a clock. Eg: pclk, iclk, sclk, ...
  - *<period>* : Real number to define period of that clock.
  - *<duty_cycle>* : Real number to define duty cycle of that clock
  - *<start_time>* : Real number to define the start time to activate of that clock
  - *<positive_first>*: String to define that that clock is positive or negative at the first time it is activated. This information has 2 values only:
    - ✔ true : that clock is positive at the first time it is activated
    - ✔ false : that clock is negative at the first time it is activated

  If there are multiple clocks, many lines with the same format will be defined.
- **%CHANNEL_SYS**: Define signals' information of the system. The information in the next line(s) is:

  <signal_name> <bit_width> <signal/port in/port out> <class_name_of_data_type>

  - *<signal_name>*: String to define name of a signal
  - *<bit_width>* : Integer number to define bit width of that signal
  - *<signal/port in/port out>* : String to define type of signal. This information has 4 values only:
    - ✔ in : input port. Eg: clock.
    - ✔ out : output port. Eg: clock.
    - ✔ inout : input/output port. Eg: clock.
    - ✔ signal : signal. Eg: interrupt signal.
    - ✔ buffer : buffer. Eg: interrupt signal.
  - *<class_name_of_data_type>* : String to define class name of data type. The usual values of this information are:
    - ✔ bool
    - ✔ sc_uint
    - ✔ sc_biguint

  If there are multiple signals, many lines with the same format will be defined.
- **%MODULE_LIST** : Define information of modules in environment. The information in the next lines is:

<module_info_file> <class_name> <instance_name>

- *<module_info_file :* String to define name of model info file.
- *<class_name>* : String to define name of model class.
- *<instance_name>* : String to define instance name of a module.

If there are multiple modules, many lines with the same format will be defined.

- **%VC_SPECIFIC_LIST**: Define the system signals that are connected to each module.

<instance_name> <port_name> <signal_name>

- *<instance_name>* : String to define instance name of a module.
- *<port_name>* : String to define name of port of that module.
- *<signal_name>*: String to define name of signal that are connected to port of that module. For example: “reset”

If there are multiple modules or multiple connections, many lines with the same format will be defined.

- **%CLOCK_PARAMETER** : Define clock information for handleCommand. The information in the next lines is:

<instance_name> <parameter_name> <clock_name>

- *<instance_name>* : String to define instance name of a module.
- *<parameter_name>* : String to define name of clock parameter.
- *<clock_name>* : name of clock signal

If there are multiple clock parameters, many lines with the same format will be defined.

3.3.2.1.2. Bus information

- **%BUS_INFO**: Define information of each bus. The information in the next line is:

<instance_name>

- *<instance_name>* : String to define instance name of bus.

- **%TOP_MODULE_INI_LIST**: Define list information of initiator modules of bus. The content of this part is the next three lines:

<instance_name> <socket_name>

- *<instance_name>* : String to define instance name of initiator of bus.
- *<socket_name> :* String to define the name of initiator socket of model.

If there are multiple initiators, many lines with the same format will be defined.

- **%TOP_MODULE_TGT_LIST**: Define list information of target modules of bus. The content of this part is the next three lines:

<instance_name> <socket_name>

- *<instance_name>* : String to define instance name of target of bus.
- *<socket_name> :* String to define the name of target socket of model.

If there are multiple targets, many lines with the same format will be defined.

- **%ADDRESS_MAP**: Define the address map of all target modules.

<start_address> <end_address> <target_instance_name>

- *<start_address>* : Hex number (started by "0x") to define the start address of memory model.
- *<end_address>* : Hex number (started by "0x") to define the end address of memory model.
- *<target_instance_name>*: String to define the instance of target VC

If there are multiple target VCs or address range, many lines with the same format will be defined.

– **%SET_ARB**: Define the information for the arbitration of bus. The information in the next lines is:

<BUS_ID> <Req/Resp> <arb.tree>

- *<BUS_ID>* : Integer number to define the index of bus resource ID.
- *<Req/Resp>* : String to define model is Request or Response bus resource ID.
- *<arb.tree>* : Define arbiter specification.

If there are multiple models, many lines with the same format will be defined.

– **%SET_INIGRP**: Define the initiator group information of arbitration. The information in the next lines is:

<ini_instance> <ini_grp_ID>

- *<instance_name>* : String to define instance name of initiator.
- *<ini_grp_ID>* : Integer number to define ID of initiator group .

If there are multiple initiator groups, many lines with the same format will be defined.

– **%SET_TGTGRP**: Define the target group information of arbitration. The information in the next lines is:

<tgt_instance> <tgt_grp_ID>

- *<instance_name>* : String to define instance name of target.
- *<ini_grp_ID>* : Integer number to define ID of target group .

If there are multiple target groups, many lines with the same format will be defined.

3.3.2.1.3. Example

– In this example, a top module with a structure like that in Figure 3.2 includes:
  - Two buses: SHwy bus (also is the initiator of HPB bus) and HPB bus (also is the target of SHwy bus).
  - One CPU model has the role as both initiator and target of the environment.
  - One DUT (TMU) model is the target module connected to SHwy bus.
  - One INTC model is the target module connected to HPB bus.

– DUT model has a 4-bit interrupt port that is connected to the INTC model. In SHwy bus declaration, there are three sections for bus arbitration (%SET_ARB, %SET_INIGRP, %SET_TGTGRP).

– The SCI file corresponding to this example is shown in Figure 3.3.

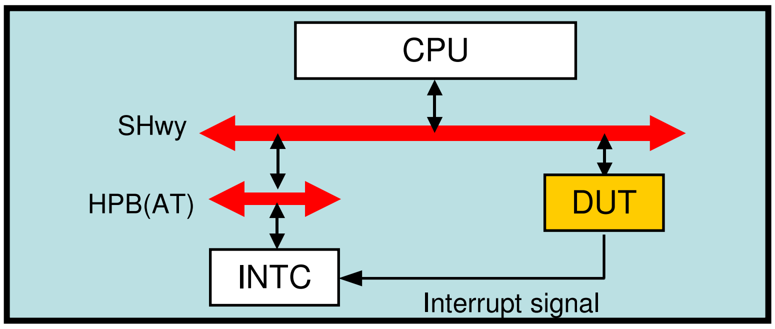


Figure 3.2: Example of an expected environment

```
%DEFINE
BUS_WIDTH_1 64
BUS_WIDTH_2 32
%ENV_INFO
# Coding_style   add_tlm_socket    ini_buswidth    tgt_buswidth
 LT     ADD_TGT_SOCKET                             BUS_WIDTH_2
%CLOCK_RATIO          # clock paramter
#      clock name     period  duty_cycle     start time     positive_first
       sclk           5              0.5            0              true
       pclk           30             0.5            0              true
%CHANNEL_SYS
#      signal_name    bit_width      <signal/port in/port out>     class_name_of_data_type
       sclk           1              in                            bool
       pclk           1              in                            bool
       int_tmu_s             4              signal                 sc_uint
%MODULE_LIST
#       model_info_file  class_name                         instance_name
        shwy_info       Shwy<1,2,BUSWIDTH_1>(“Shwy”)          bus_shwy
        cpu_info        Ccpu(“cpu”,0,0)                       cpu
        intc_info       Cintc(“intc”)                         intc
        hpbc_info       Hpbc<1,BUSWIDTH_1, 32>(“hpbc”)        bus_hpbc
        tmu_info        Ctmu(“tmu”)                           tmu
%VC_SPECIFIC_SYS
#      instance_name          port_name      signal_name
       intc                   int_tmu_n      int_tmu_s
       tmu                    int_tmu_n      int_tmu_s
%CLOCK_PARAMETER
#      instance_name          parameter_name clock_name
       tmu                    tclk0                 pclk
# Shwy Bus definition
%BUS_INFO # for indicate Bus ID in multi buses.
#      instance_name
       bus_shwy
%TOP_MODULE_INI_LIST
#      instance_name  socket_name
       cpu            m_ini_socket
%TOP_MODULE_TGT_LIST
#      instance_name  socket_name
       tmu            m_tgt_socket
       cpu            m_tgt_socket
       bus_hpb        target_socket
%ADDRESS_MAP
#      start_ address   end address    target_instance_name
       0x00000000     0x2FFFFFFF     “cpu”
       0x30000000     0x3FFFFFFF     “bus_hpb”
       0x40000000     0x4000FFFF     “tmu”
%SET_ARB
#      BUS_ID     Req/Resp      arb.tree
       0              Req            (“cpu”/”External_tgt”)
       0              Resp           (“cpu”/”tmu”/”bus_hpb”)
%SET_INIGRP
#      ini_instance   ini_grp_ID
       “cpu”          0
%SET_TGTGRP
#      tgt_instance   tgt_grp_ID
       “cpu”          0
       “tmu”          1
       “bus_hpb“      2
# HPB bus definition
%BUS_INFO # for indicate Bus ID in multi buses.
#      instance_name
       bus_hpbc
%TOP_MODULE_INI_LIST
#      instance_name  socket_name
       bus_shwy       initiator_socket[1]
%TOP_MODULE_TGT_LIST
#      instance_name  socket_name
       intc           m_tgt_socket
%ADDRESS_MAP
#      start_ address   end address    target_instance_name
       0x30000000     0x3FFFFFFF     “intc”
```

Figure 3.3: An example of connecting CPU to TLM environment.

3.3.2.2. Model info files

Model info file contains each model's information: model definition, template definition and port definition. The figure 3.4 describes the structure of this file.

```
################################################################
## Model info file
################################################################
%HEADER_FILES
%IMPLEMENTATION_FILES
%TEMPLATE
%PORT
```

Figure 3.4: Structure of model info file.

All information is marked by keywords. The meaning of keywords in this file is described in table 3.4.

Table 3.4: List of keywords in model info file

No.	Keyword	Meaning	Mandatory
1	%HEADER_FILES	Model header filenames	Exist at least one file name
2	%IMPLEMENTATION_FILES	Model implementation filenames	
3	%TEMPLATE	Template argument definition (two or more specification is possible)	No
4	%PORT	Port definition (two or more specifications are possible)	No

3.3.2.2.1. Detailed description

- **%HEADER_FILES**: Define the header filenames of a model. The information is described as below:

  %HEADER_FILES *<filename 1> <filename 2> ...*

  - *<filename 1>* : String to define the header filename of the model.
- **%IMPLEMENTATION_FILES**: Define the implementation filenames of a model. The information is describe as below:

  %IMPLEMENTATION_FILES *<filename 1> <filename 2> ...*

  - *<filename 1>* : String to define the implementation filename of the model.

  Note: At least one header or implementation file must exist.
- **%TEMPLATE**: Define the template argument of model. The information is described as below:

  %TEMPLATE *<type> <arg>*

  - *<type>* : type of argument of template

- *<arg>* : name of an argument of a template. The last argument in an argument name.

If there are multiple template arguments, many lines with the same format will be defined.

– **%PORT**: Define the port of model. The information is described as below:

%PORT *<port_name> <num_of_bit> <on/out> <type> <fix_value>*

- *<port_name>* : name of port
- *<num_of_bit>* : integer number to define the width of the port.
- *<in/out>* : String to define type of port
  - ✔ in : input port. Eg: clock.
  - ✔ out : output port. Eg: clock.
  - ✔ inout : input/output port. Eg: clock.
  - ✔ signal : signal. Eg: interrupt signal.
  - ✔ buffer : buffer. Eg: interrupt signal.
- *<type>* : String to define class name of data type. The usual values of this information are:
  - ✔ bool
  - ✔ sc_uint
  - ✔ sc_biguint
- *<fix_value>* : integer number to define the fixed value of port (optional). When an attribute can be recognized by the port name, it is not necessary to specify clearly. If fixed value is not defined, Top Module Generation script will base on table 3.5 to decide the fixed value for non-connection ports.

– If there are multiple ports, many lines with the same format will be defined.

Table 3.5: Fixed value based on port name (fixed value is not defined)

Port name	Fixed value
xxx_n	1
xxx_p	0
In the cases of other than the above	0

3.3.2.3. Skeleton files

– The list of notations in top.h.skl, top_main.skl, and Makefile.skl is shown in Table 3.6. The content of top.h.skl is shown in Figure 3.6. Figure 3.5 shows the content of the top_main.cpp.skl. Two other figures (3.7 and 3.8) show the content of Makefile.skl.

– Top.h.skl is used to generate the top.h file. In top.h.skl, beside the notations which are defined in the below Table 3.6, there are some notation blocks %%GEN_XXX_START - %%GEN_XXX_END. Users can insert source code into these blocks in top.h file . The content between these blocks will remains when re-generate the top.h file.

– Top_main.cpp.skl is used to generate the top_main.cpp file.

– Makefile.skl is used to generate the Makefile.

Table 3.6: List of notations in skeleton files

Filename	Notation	Description
top.h.skl (Content is show in figure 3.5)	%%DEFINE	Place to put the macro definitions
	%%ADD_TLM_SOCKET	Place to put adding part of external TLM socket
	%%INCLUDE_MODULE_HEADERS	Place to put module importations
	%%DECLARE_PORTS_AND_SIGNALS	Place to put ports and signals declarations
	%%DECLARE_MODULE_POINTERS	Place to put module declarations
	%%INIT_PORTS_AND_SIGNALS	Place to put ports and signals initializations
	%%MODULE_INSTANTIATION	Place to put module allocations.
	%%CONNECT_MODULES	Place to put module signal connections and socket connections
	%%COMMAND_HANDLER	Place to put command handler
	%%DELETE_MODULE_POINTERS	Place to put module deallocations.
top_main.cpp.skl (Content is show in figure 3.6)	%%SYSTEM_SIGNAL_DECL	Place to put clock declarations
	%%CONNECT_SYSTEM_SIGNAL	Place to put top module and signals connections
Makefile.skl (Content is show in figure 3.7,3.8)	%%IMPLEMENTATION_FILES	Place to put list of implementation files
	%%HEADER_FILES	Place to put list of header files

```
#include "top.h"
char *trace_file_namep;

int sc_main(int argc, char *argv[])
{
    sc_report_handler::set_actions("/IEEE_Std_1666/deprecated", SC_DO_NOTHING);

    Creslx *reslx = new Creslx ("reslx");
    if (reslx != NULL) {
        // System Signals Generation
        %%SYSTEM_SIGNAL_DECL

        // Modules Connection
        %%CONNECT_SYSTEM_SIGNAL

        // Start SystemC
        sc_start ();
        //sc_stop ();
        delete reslx;
        return 0;
    }
    else {
        printf ("\n\n[sc_main] Error: Cannot allocate memory for reslx instance!!!\n\n");
        return 0;
    }
}
```

Figure 3.5: Content of top_main.cpp.skl file

```
#ifndef __TOP_H__
#define __TOP_H__

#include <systemc.h>
#include "commandHandler.h"
extern char* trace_file_namep;

%%DEFINE
%%INCLUDE_MODULE_HEADERS

// =====  %%GEN_SPECIFIC_INCLUDE_HEADERS_POST_START  ===== >>>
// any additional code here
// =====  %%GEN_SPECIFIC_INCLUDE_HEADERS_POST_END  ===== >>>

class Creslx: public sc_module
%%ADD_TLM_SOCKET
{
public:
    %%DECLARE_PORTS_AND_SIGNALS
    // pointer for module instance
    %%DECLARE_MODULE_POINTERS
    // =====  %%GEN_SPECIFIC_MODULE_POINTERS_POST_START  ===== >>>
    // any additional code here
    // =====  %%GEN_SPECIFIC_MODULE_POINTERS_POST_END  ===== <<<

    vpcl::commandHandler *cmd_handler;

    SC_HAS_PROCESS (Creslx);
    Creslx (sc_module_name name) :
    sc_module (name)
        %%ADD_DECLARE_TLM_SOCKET
        %%INIT_PORTS_AND_SIGNALS
    {
        // module instantiation
        %%MODULE_INSTANTIATION
        // =====  %%GEN_SPECIFIC_MODULE_INSTANTIATION_POST_START  ===== >>>
        // any additional code here
        // =====  %%GEN_SPECIFIC_MODULE_INSTANTIATION_POST_END  ===== <<<
        //connect modules
        %%CONNECT_MODULES
        // =====  %%GEN_SPECIFIC_CONNECT_MODULES_POST_START  ===== >>>
        // any additional code here
        // =====  %%GEN_SPECIFIC_CONNECT_MODULES_POST_END  ===== <<<
        // handleCommand
        cmd_handler = new vpcl::commandHandler("handleCommand");
        cmd_handler->register_command_processor(cpu->name(),cpu, &Ccpu::handleCommand);
        %%COMMAND_HANDLER
        cmd_handler->handleCommand();
        // =====  %%GEN_SPECIFIC_COMMAND_HANDLER_POST_START  ===== >>>
        // any additional code here
        // =====  %%GEN_SPECIFIC_CONNECT_HANDLER_POST_END  ===== <<<
        // trace all signal
        #ifdef __WAVE_DUMP_DEBUG__
        if (trace_file_namep != NULL) {
            sc_trace_file *tf = sc_create_vcd_trace_file(trace_file_namep);
            // =====  %%GEN_SPECIFIC_TRACE_POST_START  ===== >>>
            // any additional code here
            // =====  %%GEN_SPECIFIC_TRACE_POST_END  ===== <<<
        }
        #endif
    }
    ~Creslx() {
        delete cmd_handler;
        %%DELETE_MODULE_POINTERS
        // =====  %%GEN_SPECIFIC_DELETE_MODULE_POINTERS_POST_START  ===== >>>
        // any additional code here
        // =====  %%GEN_SPECIFIC_DELETE_MODULE_POINTERS_POST_END  ===== <<<
    }
};

#endif // __TOP_H__
```

Figure 3.6: Content of top.h.skl file

```
# ---------------------------------------
# indispensable macro
# ---------------------------------------
# your source files
       SRCS = \
         top_main.cpp \
        %%IMPLEMENTATION_FILES

       HEDS = top.h %%HEADER_FILES
    EXE = tlmreslx.exe

# ---------------------------------------
# basic optional macro
# ---------------------------------------
# C++ compiler
CC = g++
#TARGET_ARCH = gccsparcOS5-g2_95_2
#TARGET_ARCH = linux-g2_96
#TARGET_ARCH = linux-g3_2_3

# compile & link option
INCDIR = -I./include -I/shsv/sld/Common/Lib/99_Others/include
LIBDIR = -L./libs/$(TARGET_ARCH) -L/usr/local/lib -L/usr/lib -L/shsv/sld/Common/Lib/99_Others
#LIBDIR = -L../
#LIB    = -lcedar2_forest

#ifndef UNUSE_BFD
##LIB+= -lbfd -liberty ##-lintl
#endif /* USE_BFD */
#ifdef USE_READLINE

#LIB+= -lcurses
LIB+= -lreadline -lcurses
#endif //USE_READLINE
DEBUG  = -g #-DDEBUG_TIMING #-DDEBUG_REQ #-DDEBUG_RES #-DDEBUG_ARBIT
DEFS   = -DSC_INCLUDE_DYNAMIC_PROCESSES
OPT    = #-funroll-loops
OPT   += -O0 -Wall #-O3
OPT   += -m32
EXTRA_DEF = -DUSE_READLINE #-DNO_CHECK_API #-DMERGE_ICLK #-DPRIORITY_OMIT
CFLAG  = $(DEFS) $(DEBUG) $(OPT) $(EXTRA_DEF) $(DEFADDR) -fexceptions -D__WAVE_DUMP_DEBUG__
-fprofile-arcs -ftest-coverage #-D__SHWY_TEST__ #-D__IF_DEBUG__

# generated execute file's name
```

Figure 3.7: Content of Makefile.skl file(part 1)

```
# ---------------------------------------
# advanced optional macro
# ---------------------------------------
#include file directory
SEARCH_DIR =

# generated object file's directory
OBJ_DIR = ./obj

# compile SystemC
#SC_TYPE = SystemC-2.0.1
#SC_TYPE = SystemC-2.1.v1
#SC_TYPE = SystemC-2.0.1gcc
#SC_TYPE = SystemC-2.0.1-kai
#SC_TYPE = SystemC-2.2_070319
SC_TYPE = systemc-2.2.0

# add rule to create EXE
ADD_EXE = $(HEDS)

# use SCV
#SCV_USE = yes # with SCV
 SCV_USE = no  # without SCV

# debug SystemC code in DDD
#SYSTEMC_DEBUG = _debug  # yes
SYSTEMC_DEBUG =         # no
#SYSTEMC_DEBUG = _ee3.0


# install directory
SYSTEMC = /shsv/sld/Common/Lib/01_SystemC/$(SC_TYPE)
#SYSTEMC = /home/u/tuantu/Desktop/test/$(SC_TYPE)
#SYSTEMC = /ssv/prj113/prj/SHX/shx_work/users/nishihr/$(SC_TYPE)
#SCVDIR  = /common/appl/Renesas/SystemC/SCV-1.0
TLM = /shsv/sld/Common/Lib/04_TLM/TLM2.0-2008-06-09

# ---------------------------------------
# include rule file
# ---------------------------------------
# original Makefile.defs is below.
#include /common/appl/Renesas/SystemC/examples/Makefile.defs
include ./Makefile.defs

# ---------------------------------------
# execute rule
# ---------------------------------------
# primary rule
all: pre_check $(EXE)
```

Figure 3.8: Content of Makefile.skl file(part 2)

3.4. Output

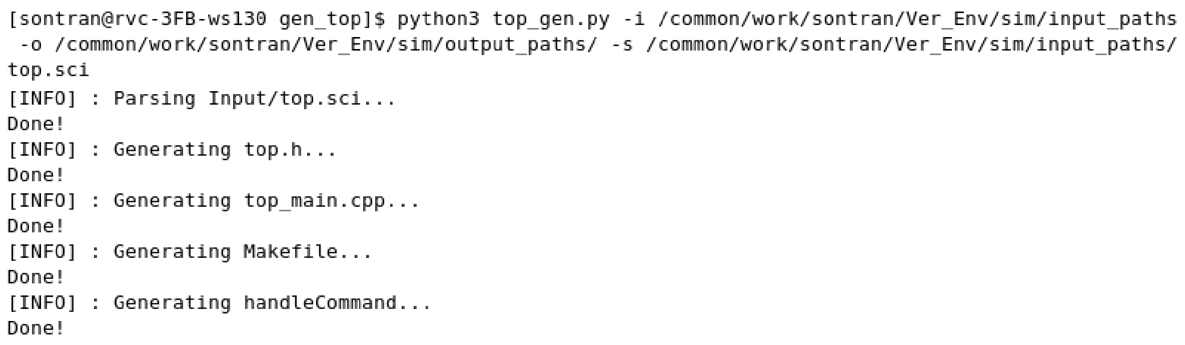
– After running the script, the top module files (*top.h, top_main.cpp*) and handleCommand file are created. Another output file is the Makefile which is used for environment compilation.

3.5. Example

– Using the configuration file *top.sci* (figure 3.3), *Makefile.skl* (figure 3.7,3.8), *top.skl* (figure 3.6), *top_main.cpp.skl* (figure 3.5) and needed model info files, Top Module Generation Script will generate the top.h, Makefile and handleCommand file.

```
% python3 top_gen.py -i /home/u/user/Ver_Env/sim/input_files/
-o /home/u/user/Ver_Env/input_files/sim/output_files/
-s /home/u/user/Ver_Env/sim/input_files/top.sci
```

- **-i** /home/u/user/Ver_Env/input_files/ : Specify the input path where stores all the needed input files.
- **-o** /home/u/user/Ver_Env/input_files/output_files/ : Specify the output path where output files will be stored
- **-s** /home/u/user/Ver_Env/input_files/top.sci : Specify the configuration files



```
[sontran@rvc-3FB-ws130 gen_top]$ python3 top_gen.py -i /common/work/sontran/Ver_Env/sim/input_paths
 -o /common/work/sontran/Ver_Env/sim/output_paths/ -s /common/work/sontran/Ver_Env/sim/input_paths/
top.sci
[INFO] : Parsing Input/top.sci...
Done!
[INFO] : Generating top.h...
Done!
[INFO] : Generating top_main.cpp...
Done!
[INFO] : Generating Makefile...
Done!
[INFO] : Generating handleCommand...
Done!
```

Figure 3.9: Example of running Top Module Generation Script

– After running script successfully, users have the below generated output files:

```
#ifndef __TOP_H__
#define __TOP_H__

#include <systemc.h>
#include "commandHandler.h"

extern char* trace_file_namep;

#define BUSWIDTH_1 64
#define BUSWIDTH_2 32

#include "tmu.h"
#include "intc.h"
#include "Shwy.h"
...

// =====  %%GEN_SPECIFIC_INCLUDE_HEADERS_POST_START  ===== >>>
// any additional code here
// =====  %%GEN_SPECIFIC_INCLUDE_HEADERS_POST_END  ===== >>>

class Creslx: public sc_module
{
public:
    sc_in<bool>iclk;
    sc_in<bool>bclk;
    sc_in<bool>pclk;
    sc_inout<bool>     reset;
    sc_in<bool>sclk;
    sc_signal<sc_uint<6> >    int_vin_n;
    ...

    // pointer for module instance
    Ctmu *tmu;
    Cintc *intc;
    Shwy<1,3,BUSWIDTH_1> *bus_shwy;
    ...

    // =====  %%GEN_SPECIFIC_MODULE_POINTERS_POST_START  ===== >>>
    // any additional code here
    // =====  %%GEN_SPECIFIC_MODULE_POINTERS_POST_END  ===== <<<

    vpcl::commandHandler *cmd_handler;
    SC_HAS_PROCESS (Creslx);
    Creslx (sc_module_name name) :
    sc_module (name)
        ,reset("reset")
        ,sclk("sclk")
        ...

    {
        // module instantiation
        sbsc = new Csbsc("sbsc");
        tmu = new Ctmu("tmu");
        intc = new Cintc("intc");
        bus_shwy = new Shwy<1,3,BUSWIDTH_1>("bus_shwy");
        ...

        // =====  %%GEN_SPECIFIC_MODULE_INSTANTIATION_POST_START  ===== >>>
        // any additional code here
        // =====  %%GEN_SPECIFIC_MODULE_INSTANTIATION_POST_END  ===== <<<


        //connect modules
        cpu->clk_in(iclk);
        cpu->rst_in(reset);
        pedmy->int_vin_n(int_vin_n);
        ...
```

Figure 3.10: Example of top.h(part 1)

```
        // =====  %%GEN_SPECIFIC_CONNECT_MODULES_POST_START  ===== >>>
        // any additional code here
        adopter->set_cpu_pointer(cpu);
        // =====  %%GEN_SPECIFIC_CONNECT_MODULES_POST_END  ===== <<<

        // handleCommand
        cmd_handler = new vpcl::commandHandler("handleCommand");
        cmd_handler->register_command_processor(cpu->name(),cpu, &Ccpu::handleCommand);
        cmd_handler->register_command_processor(tmu->name(),tmu,&Ctmu::handleCommand);
        cmd_handler->register_command_processor(intc->name(),intc,&Cintc::handleCommand);
        ...
        cmd_handler->handleCommand();

        // =====  %%GEN_SPECIFIC_COMMAND_HANDLER_POST_START  ===== >>>
        // any additional code here
        std::vector<std::string> cmd_line;
        cmd_line.clear();
        cmd_line.push_back("set_mem");
        cmd_line.push_back("0x00000000");
        cmd_line.push_back("1G");
        sbsc->handleCommand(cmd_line);
        cmd_line.clear();
        // =====  %%GEN_SPECIFIC_CONNECT_HANDLER_POST_END  ===== <<<

        // trace all signal
        #ifdef __WAVE_DUMP_DEBUG__
        if (trace_file_namep != NULL) {
            sc_trace_file *tf = sc_create_vcd_trace_file(trace_file_namep);
            // =====  %%GEN_SPECIFIC_TRACE_POST_START  ===== >>>
            // any additional code here
            // =====  %%GEN_SPECIFIC_TRACE_POST_END  ===== <<<
        }
        #endif
    }
    ~Creslx() {
        delete cmd_handler;
        delete tmu;
        delete intc;
        delete bus_shwy;
        ...

        // =====  %%GEN_SPECIFIC_DELETE_MODULE_POINTERS_POST_START  ===== >>>
        // any additional code here
        cpu = NULL;
        bus_shwy = NULL;
        // =====  %%GEN_SPECIFIC_DELETE_MODULE_POINTERS_POST_END  ===== <<<
    }
};
#endif // __TOP_H__
```

Figure 3.11: Example of top.h(part 2)

```
#include "top.h"

char *trace_file_namep;
void *p_cpup[N_CPU];
Creslx *reslx;

int sc_main(int argc, char *argv[])
{
    sc_report_handler::set_actions("/IEEE_Std_1666/deprecated", SC_DO_NOTHING);

    sc_set_time_resolution (1, SC_NS);
    //get trace file name and ssc file name
    trace_file_namep = NULL;
    char* ssc_file_namep = NULL;
    for (int i = 1; i < argc; i++) {
        if (strcmp (argv[i], "-vcd") == 0) {
            if (i+1 < argc) {
                trace_file_namep = argv [i+1];
                i++;
            }
        } else if (strcmp (argv[i], "-src") == 0) {
            if (i+1 < argc) {
                ssc_file_namep = argv [i+1];
                i++;
            }
        }
    }

    reslx = new Creslx ("reslx");
    if (reslx != NULL) {
        // System Signals Generation
        sc_clock pclk("pclk",30,0.5,0,true);
        sc_clock sclk("sclk",5,0.5,0,true);
        sc_signal <bool>      reset;
        ...

        // Modules Connection
        reslx->pclk(pclk);
        reslx->sclk(sclk);
        reslx->reset(reset);
        ...

        vpcl::commandHandler *cmd_handler_ssc;
        //set handler
        cmd_handler_ssc = new vpcl::commandHandler(ssc_file_namep);
        cmd_handler_ssc->register_command_processor(reslx->cpu->name(),reslx->cpu,
&Ccpu::handleCommand);
        cmd_handler_ssc->register_command_processor(reslx->tmu->name(),reslx->tmu,
&Ctmu::handleCommand);
        reslx->cpu->setCommandHandler(cmd_handler_ssc);
//        cmd_handler_ssc->set_msg_lvl(0);
        cmd_handler_ssc->handleCommand();

        // Start SystemC
        sc_start ();
        //sc_stop ();
        delete reslx;
        return 0;
    }
    else {
        printf ("\n\n[sc_main] Error: Cannot allocate memory for reslx instance!!!\n\n");
        return 0;
    }
}
```

Figure 3.12: Example of top_main.cpp

```
# --------------------------------------
# indispensable macro
# --------------------------------------
# your source files
       SRCS = \
        top_main.cpp \
        re_register.cpp \
        memory_if.cpp \
        tmu.cpp \
        Cgeneral_timer.cpp \
        intc.cpp \
        adopter.cpp \
        cpu.cpp \
        cpu_command.cpp \
        cpu_utility.cpp \
       ...

       HEDS = top.h re_register.h tmu.h intc.h intc_reg.h Shwy.h cpu.h ...
    EXE = tlmreslx.exe

# --------------------------------------
# basic optional macro
# --------------------------------------
# C++ compiler
CC = g++
...
```

Figure 3.13: Example of Makefile

4. Reference

4.1. Message list

– Table 4.1 lists all messages which can be displayed during the operation of Top Module Generation Script of RESL-X.

Table 4.1 List of messages

No.	Severity	Message	Description
1	Info and Error	*[INFO] : Current Python version: %d. [ERROR]: Python 3.1.2 or higher is recommended for running this script.*	Using wrong version of Python
2	Warning	*[top_gen.py] WARNING : Old file TOPFILE does not exist.*	The old TOPFILE file does not exist so new TOPFILE will be created
3	Error	*[top_gen_class.py] ERROR: Cannot open FILE_NAME file.*	File has name FILE_NAME can not be opened
4	Error	*[top_gen.py] [ERROR] : Too many arguments.\nTry 'top_gen.py -h' fo r more information.*	There are too many arguments of an option
5	Error	*[top_gen_class.py] [MODEL_INFO_FILE] ERROR: Model info file is empty.*	MODELINFOFILE has no content
6	Error	*[top_gen_class.py] ERROR: Cannot open MODELINFOFILE file.*	MODELINFOFILE can not be opened
7	Error	*[top_gen_class.py] [%s - line %d] ERROR: Lack of KEYWORD information.*	KEYWORD's information was lacked
8	Error	*[top_gen_class.py] [%s - line %d] ERROR: Lack of argurment.*	The content of KEYWORD was lacked of argument
9	Error	*[top_gen_class.py] [%s - line %d] ERROR: Too many arguments.*	The content of KEYWORD has too many arguments
10	Error	*[top_gen_class.py] [%s - line %d] ERROR: Setting value must be greater than 0.*	The setting value of an argument must be greater than 0 (Ex: clock cycle)
11	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid value setting.*	The setting value of an argument is invalid

No.	Severity	Message	Description
12	Error	*[top_gen_class.py] [%s - line %d] ERROR: Macro was not defined.*	The macro was not defined
13	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid defined value. Macro should be CLASSTYPE.*	The type of defined macro is invalid
14	Error	*[top_gen_class.py] [%s - line %d] ERROR: Instance name must be put in quote.*	The instance name must be put in quote in %ADDRESS_MAP, %SET_ARB, %SET_INIGRP and %SET_TGTGRP
15	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid instance name.*	The instance name is invalid
16	Error	*[top_gen_class.py] [%s - line %d] ERROR: Too many quotation marks.*	There are too many quotation marks in instance name
17	Error	*[top_gen_class.py] [%s - line %d] ERROR: Keyword KEYWORD is duplicated.*	The KEYWORD is duplicated
18	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid KEYWORD content.*	The content of KEYWORD is invalid
19	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid adding external TLM socket.*	Adding wrong external TLM socket
20	Error	*[top_gen_class.py] [%s - line %d] ERROR: Lack of bus width argument.*	The bus width argument was lacked
21	Error	*[top_gen_class.py] [%s - line %d] ERROR: Too many bus width arguments.*	There are too many bus width arguments
22	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid TLM coding style (LT/AT).*	The coding style is invalid
23	Error	*[top_gen_class.py] [%s - line %d] ERROR: Duplicated coding style declaration.*	The declaration of coding style is invalid
24	Error	*[top_gen_class.py] [%s - line %d] ERROR: Duplicated name of clock signal.*	The name of clock signal is duplicated
25	Error	*[top_gen_class.py] [%s - line %d] ERROR: Clock duty cycle must less than 1.*	The clock duty cycle must be less than 1
26	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid Positive first setting.*	The setting of Positive first is invalid
27	Error	*[top_gen_class.py] [%s - line %d] ERROR: Clock name does not exist.*	The name of clock does not exist
28	Error	*[top_gen_class.py] [%s - line %d] ERROR: Instance name does not exist.*	The instance name does not exist
29	Error	*[top_gen_class.py] [%s - line %d] ERROR: Duplicated name of signal.*	The name of signal is duplicated
30	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid signal type.*	The type of declaration signal is invalid
31	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid class type of signal.*	The class type of declaration signal is invalid
32	Error	*[top_gen_class.py] [%s - line %d] ERROR: Class type of connected port does not match.*	The class type of connected port does not match
33	Error	*[top_gen_class.py] [%s - line %d] ERROR: Bitwidth of connected port does not match.*	The bitwidth of connected port does not match
34	Error	*[top_gen_class.py] [%s - line %d] ERROR: Port name does not exist.*	The port name does not exist
35	Error	*[top_gen_class.py] [%s - line %d] ERROR: Signal name does not exist.*	The signal name does not exist
36	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid Template declaration in class name.*	The Template declaration in class name is invalid
37	Error	*[top_gen_class.py] [%s - line %d] ERROR: Error while processing MODEL_INFO_FILE file.*	Error was occurred while processing MODEL_INFO_FILE
38	Error	*[top_gen_class.py] [%s - line %d] ERROR: Duplicate instance name.*	The instance name is duplicated
39	Error	*[top_gen_class.py] [%s - line %d] ERROR: Bus ID should be an integer.*	The Bus ID must be an integer

No.	Severity	Message	Description
40	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid Req/Resp setting.*	The setting of Req/Resp is invalid
41	Error	*[top_gen_class.py] [%s - line %d] ERROR: Adding external target TLM socket was not defined.*	Adding external target TLM socket was not defined before using
42	Error	*[top_gen_class.py] [%s - line %d] ERROR: Instance name of initiator does not exist in %TOP_MODULE_INI_LIST.*	The declared instance name does not exist in %TOP_MODULE_INI_LIST
43	Error	*[top_gen_class.py] [%s - line %d] ERROR: Adding external initiator TLM socket was not defined.*	Adding external initiator TLM socket was not defined before using
44	Error	*[top_gen_class.py] [%s - line %d] ERROR: Instance name of target does not exist in %TOP_MODULE_TGT_LIST.*	The declared instance name does not exist in %TOP_MODULE_TGT_LIST
45	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid Arbitration tree setting.*	The setting of arbitration tree is invalid
46	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid syntax of Arbitration tree setting.*	The syntax of setting arbitration tree is invalid
47	Error	*[top_gen_class.py] [%s - line %d] ERROR: Lack of %BUS_INFO information.*	The information of %BUS_INFO keyword is lacked
48	Error	*[top_gen_class.py] [%s - line %d] ERROR: Bus instance is not declared in %MODULE_LIST.*	The declared instance name does not exist in %MODULE_LIST
49	Error	*[top_gen_class.py] [%s - line %d] ERROR: Duplicated bus instance name.*	The bus instance name is duplicated
50	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid content of %BUS_INFO.*	The content of %BUS_INFO keyword is invalid
51	Error	*[top_gen_class.py] [%s - line %d] ERROR: There is no keyword cover this line.*	There is no keyword cover this line
52	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid class name.*	The class name must be started by an alphabet or underscore
53	Errror	*[top_gen_class.py] [%s - line %d] ERROR: Invalid instance name.*	The instance name must be started by an alphabet or underscore
54	Error	*[top_gen_class.py] [%s - line %d] ERROR: Bitwidth of bool signal must be equal 1*	The bit width of a bool signal must be equal 1
55	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid clock signal name*	The clock signal name must be started by an alphabet or underscore
56	Error	*[top_gen_class.py] [%s - line %d] ERROR: Invalid signal name*	The signal name must be started by an alphabet or underscore
57	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Duplicated port name.*	The port name in MODEL_INFO_FILE is duplicated
58	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Bitwidth must be greater than 0.*	The bitwidth of port in MODEL_INFO_FILE must be greater than 0
59	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Invalid template name.*	The Template name in MODEL_INFO_FILE is invalid
60	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Bitwidth should be an integer.*	The bitwidth of port in MODEL_INFO_FILE must be an integer
61	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Invalid signal/port type.*	The type of signal/port in MODEL_INFO_FILE is invalid
62	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Invalid class type of signal/port.*	The class type of signal/port in MODEL_INFO_FILE is invalid
63	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Lack of KEYWORD content.*	The content of KEYWORD in MODEL_INFO_FILE is lacked
64	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Too many arguments of KEYWORD content.*	There are too many argument of content of KEYWORD in MODEL_INFO_FILE
65	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Lack of header and implementation file name.*	There is no declaration of header file or implementation file in BUS_MODEL_INFO_FILE

No.	Severity	Message	Description
66	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Invalid class name.*	The class name must be started by an alphabet or underscore
67	Errror	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Invalid instance name.*	The instance name must be started by an alphabet or underscore
68	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Bitwidth of bool signal/port must be equal 1*	The bit width of a bool signal/port must be equal 1
69	Error	*[top_gen_class.py] [MODEL_INFO_FILE - line %d] ERROR: Invalid signal/port name*	The signal/port name must be started by an alphabet or underscore

Revision History

Rev.	Modified Contents	Approval	Reviewed by	Created by
1.0	New creation		Chau Nguyen 07/06/2010	Son Tran 07/06/2010