EDA user’s manual

Tool name (version):

High-Level design supporting tool
ssgen user’s manual
(v1.5)

Abbreviation: ssgen

Outline

This manual is a summary of the execution procedure to apply ssgen to the high-level design efficiently.

Related material

Attention

Contents

1. Introduction

1.1 About this manual

This manual has aimed to apply high-level design supporting tool ssgen (Synthesizable SystemC code Generator) to the high-level design efficiently.

1.2 What is ssgen?

Ssgen is a tool that generates complicated SystemC description parts such as module definition, port connections, and external memory accesses, automatically. The designer can concentrate on implementing the main function by generating the SystemC framework with this tool.

2. Environmental setting

Execution file ssgen.pl is necessary for the design by using ssgen. Please contact SIDA if you need these files. These files are installed in the following path. Please get them from the following path.

[REL] /common/appl/Renesas/SystemC/utility/ssgen

Ssgen is a perl script. The below table shows the system environment of ssgen.

OS	Version of Perl
Linux (RHEL3, RHEL4, RHEL5, SLES10)	perl v5.8.0
WindowsXP	Active Perl v5.14

And, the below table shows the basic version of EDA tools linking to ssgen.

EDA tool	Version
OSCI SystemC	2.2
VCS-MX of Synopsys	2011.12-sp1-1
IES of Cadence	12.10s004
CtoS of Cadence	12.20-s201
1Team:System of Atrenta	1.16.7
SLEC of Calypto	7.1-prod-5
SSChecker of SIDA	2.2

It is possible to change the environment and the version arbitrarily by specifying the input to ssgen.

3. Functional overview

3.1 SystemC module generation mode

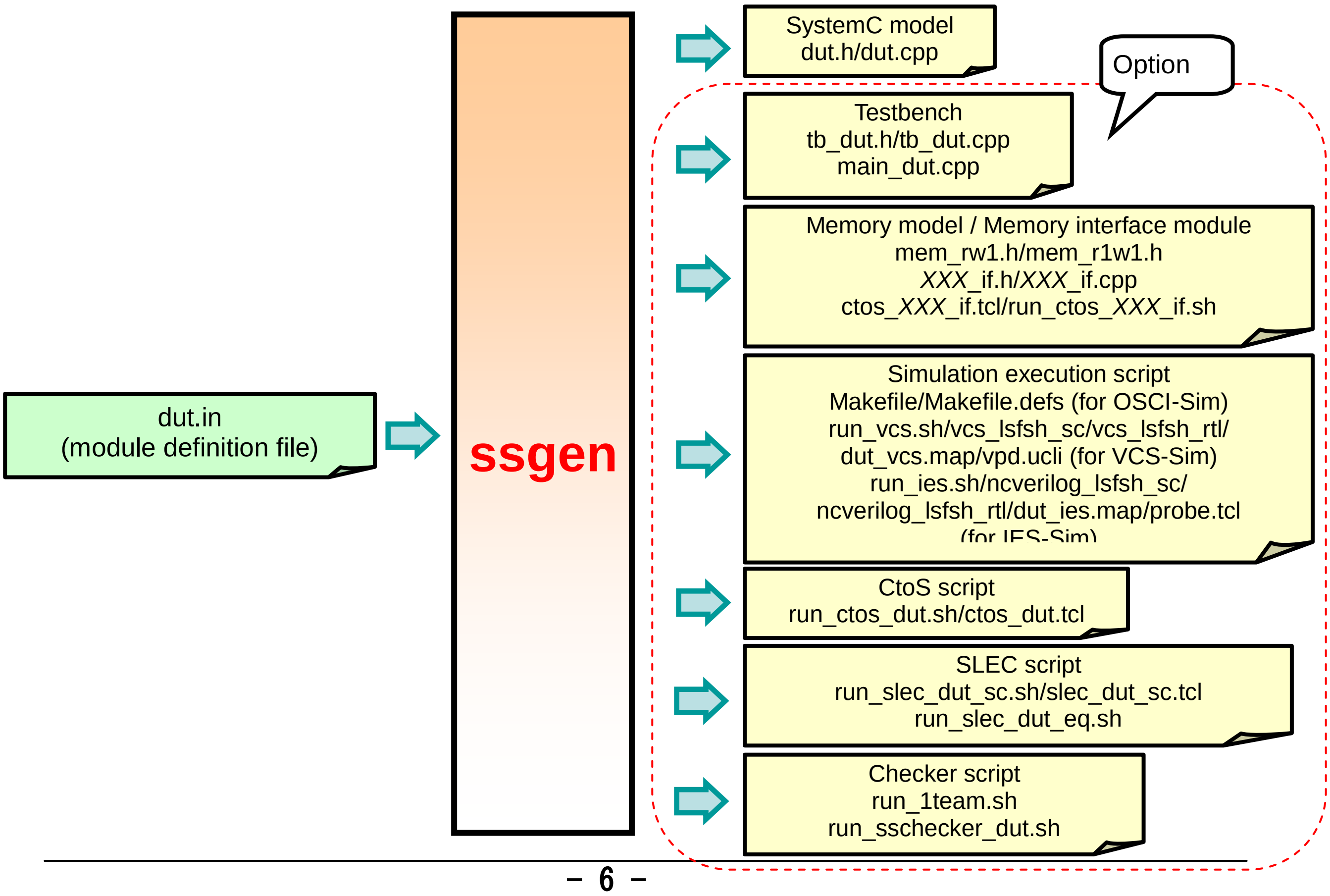
Ssgen has the module generation mode and the hierarchy generation mode. The module generation mode generates the SystemC framework from the module definition file which is an input file of this mode. The hierarchy generation mode generates the hierarchical module instantiating internal modules from the hierarchy definition file which is an input file of this mode.

Ssgen switches the module generation mode and the hierarchy generation mode by the format of input file.

Moreover, it is also possible that both modes generate the model of module, the testbench, the memory model, simulation execution script (for OSCI-Sim, VCS-MX of Synopsys, and for IES of Cadence), the CtoS script (CtoS: high-level synthesis tool of Cadence) and etc.

3.1.1 Module generation mode

The figure below shows the I/O of the module generation mode. This mode generates the model of the module (SystemC model), the model of the testbench, the memory model, the simulation execution script, the CtoS script, the SLEC script, the checker script and the memory interface module (SystemC model and the CtoS script) from a module definition file. About, condition of generation of a memory interface module, refer to the command reference of {u|s}mem.


SystemC model
dut.h/dut.cpp
Option
Testbench
tb_dut.h/tb_dut.cpp
main_dut.cpp
Memory model / Memory interface module
mem_rw1.h/mem_r1w1.h
XXX_if.h/XXX_if.cpp
ctos_XXX_if.tcl/run_ctos_XXX_if.sh
dut.in
(module definition file)
ssgen
Simulation execution script
Makefile/Makefile.defs (for OSCI-Sim)
run_vcs.sh/vcs_lsfsh_sc/vcs_lsfsh_rtl/
dut_vcs.map/vpd.ucli (for VCS-Sim)
run_ies.sh/ncverilog_lsfsh_sc/
ncverilog_lsfsh_rtl/dut_ies.map/probe.tcl
(for IES-Sim)
CtoS script
run_ctos_dut.sh/ctos_dut.tcl
SLEC script
run_slec_dut_sc.sh/slec_dut_sc.tcl
run_slec_dut_eq.sh
Checker script
run_1team.sh
run_sschecker_dut.sh

When ssgen generates the model of the module, the model of the testbench, and the simulation execution script and the CtoS script, if the same name file exists, ssgen takes the following measures for prevention from overwrite.

(1) the source files of module and testbench

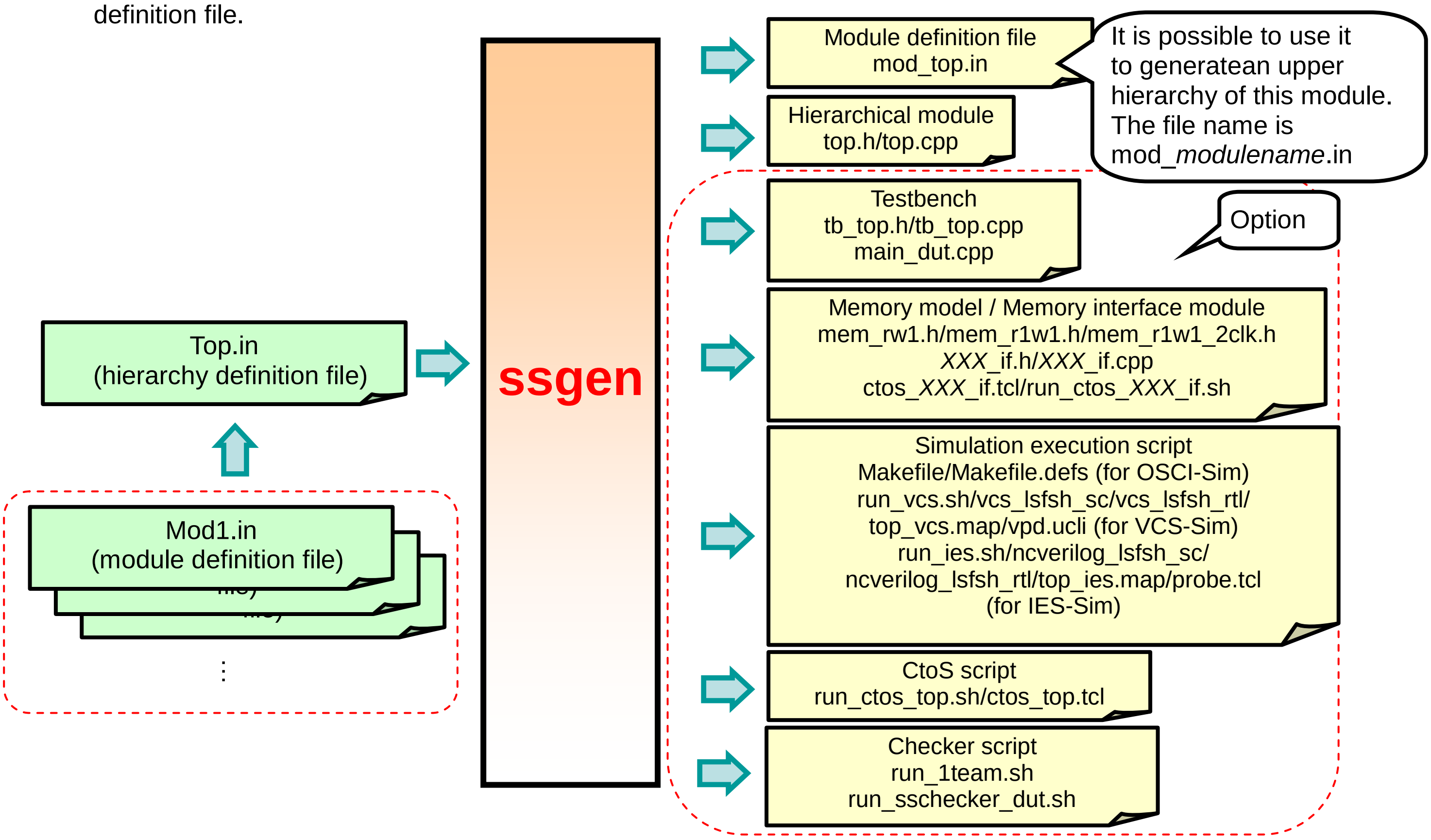
ssgen generates a file by the name which added "_tmp" to the last of the file name, without renaming the existing file. For example, when dut.cpp exists at the time of dut.cpp generation, it generates by a name called dut.cpp_tmp. It is overwritten when dut.cpp_tmp already exists.

(2) the other files

ssgen generates these files after renaming the existing files. For example, when ssgen generates dut.h and dut.h already has existed, ssgen generates dut.h after renaming the existing dut.h to dut.h.1. When dut.h and dut.h.1 already have existed, ssgen generates dut.h after renaming the existing dut.h to dut.h.2. Because the generated code doesn't depend on the content of the module definition file, rename operations are not worked about mem_rw1.h, mem_r1w1.h, vpd.ucli or probe.tcl.

3.1.2 Hierarchy generation mode

The figure below shows the I/O of the hierarchy generation mode. This mode generates the model of a hierarchical module that bundles one or more internal modules, the model of the testbench, the memory model, the simulation execution script, CtoS script for hierarchy RTL generation, the checker script and the memory interface module (SystemC model and the CtoS script) from a hierarchy definition file.


Top.in
(hierarchy definition file)
Mod1.in
(module definition file)
⋮
ssgen
Module definition file
mod_top.in
It is possible to use it to generatean upper hierarchy of this module. The file name is mod_*modulename*.in
Hierarchical module
top.h/top.cpp
Testbench
tb_top.h/tb_top.cpp
main_dut.cpp
Option
Memory model / Memory interface module
mem_rw1.h/mem_r1w1.h/mem_r1w1_2clk.h
*XXX*_if.h/*XXX*_if.cpp
ctos_*XXX*_if.tcl/run_ctos_*XXX*_if.sh
Simulation execution script
Makefile/Makefile.defs (for OSCI-Sim)
run_vcs.sh/vcs_lsfsh_sc/vcs_lsfsh_rtl/
top_vcs.map/vpd.ucli (for VCS-Sim)
run_ies.sh/ncverilog_lsfsh_sc/
ncverilog_lsfsh_rtl/top_ies.map/probe.tcl
(for IES-Sim)
CtoS script
run_ctos_top.sh/ctos_top.tcl
Checker script
run_1team.sh
run_sschecker_dut.sh

When ssgen generates the model of the hierarchical module, the model of the test bench, and the simulation execution script and the CtoS script, if the same name file exists, ssgen generates these files after renaming the existing files in the same way as the module generation mode.

In the hierarchy generation mode, ssgen connects the signal between internal modules as follows.

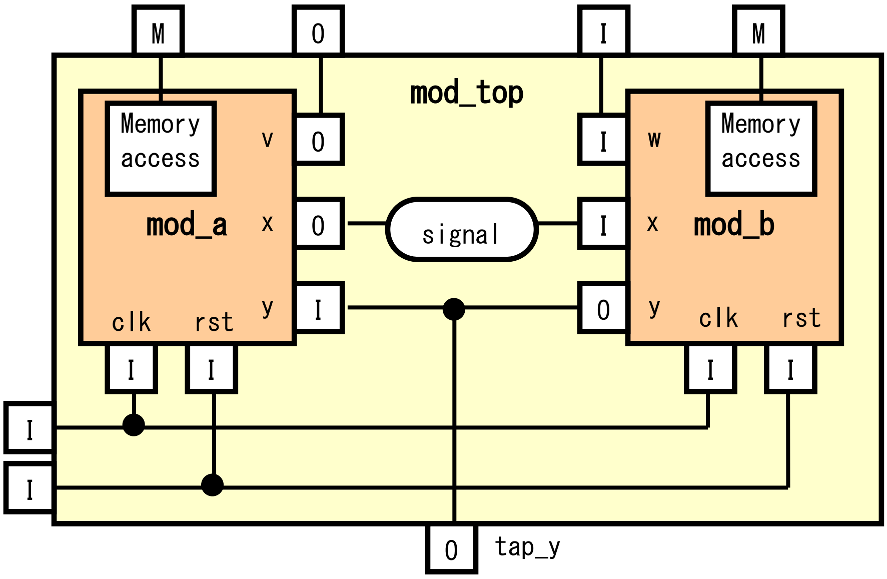
・If internal modules have an same name port and the port is the pair of input and output between internal modules, or if the connection of input and output pair is specified by BIND (the below-mentioned bind command), ssgen connects the signal(sc_signal) between internal modules (signal x in the figure below).

・The ports except the above is tapped out as a port of a hierarchical module (v and w in the figure below). If internal modules have same name port, ssgen merges it and connect it to internal modules (clk and rst in the figure below).

・Ssgen connects an output port of an internal module to input ports of two or more modules by fan-out.

・When TAP (the following tap commands) is specified with the hierarchy definition file for connecting signal between internal modules, the signal is tapped out as an output port of a hierarchical module (y and tap_y in the figure below). It is also possible to specify TAP to a port and to pull out as a port of a hierarchy module by an alias.

・When an internal module has the memory definition, ssgen generates the memory access ports in a hierarchical module.



3.2 Memory model

In ssgen, when the SystemC module has the memory access, the memory model can be generated. Moreover, the function description for the SystemC module/testbench to access the memory can be generated.

3.2.1 Simulation environment

In the simulation environment of SystemC generated with ssgen, it can be operated by switching the following two memory access description.

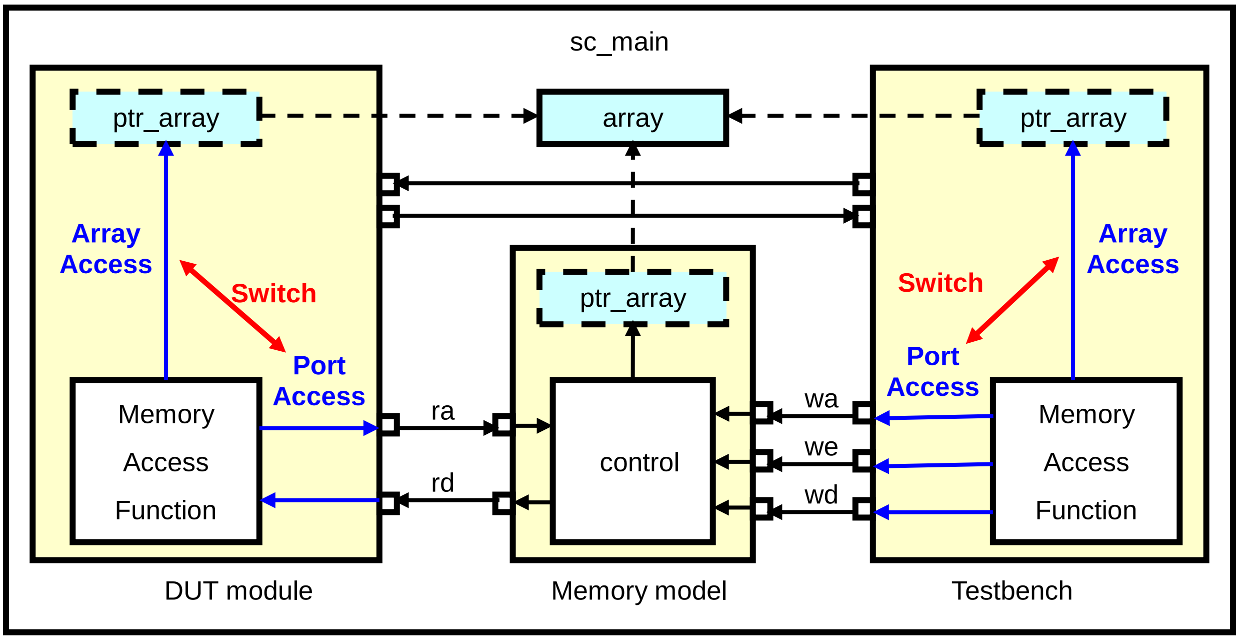
・Memory port access description connected to a memory model (**Default from v1.5**)

・Memory array access description without memory ports (**Not supported from v1.5**)

The memory port access and the memory array access are switched by the macro specification (_ MEM_MODEL macro) when compiling. The memory port access description becomes effective when the _MEM_MODEL macro is specified, and when _MEM_MODEL macro is not specified, the memory array access description becomes effective.

To make high-level synthesis, it is necessary to use the memory port access description.

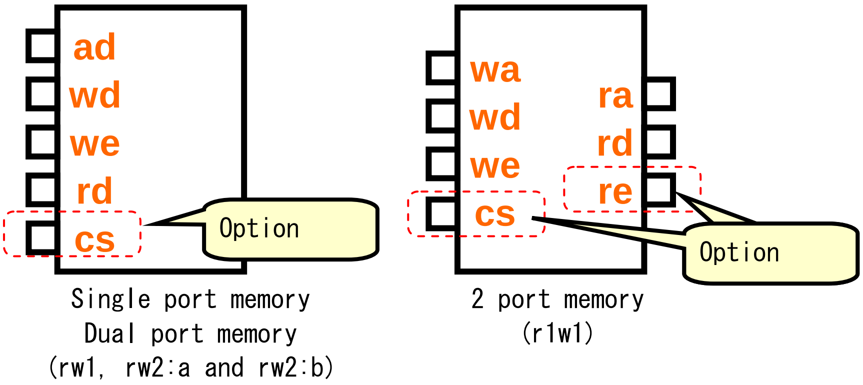
The following figure shows a simulation environment including 2-port memory model.



We recommend that you select memory port access description in SystemC simulation (specifying _MEM_MODEL macro). Because the simulation speed of memory array access description rarely different from that of memory port access description.

3.2.2 Kind of memory model

Ssgen corresponds to two memory models (single port memory and 2 port memory). You can select whether you use or not chip-select port (cs) of single port memory, whether you use or not write-enable (we)/chip-select (cs)/read-enable port (re) of 2 port memory, and also whether you use or not chip-select port (cs) of dual port memory in the module definition file.

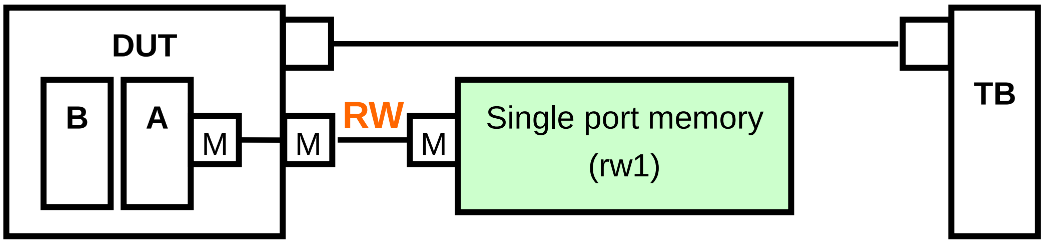


All memory models which are generated by ssgen have a function storing previous value of read data port. A value of read data port has never been changed until the next read access. So, if you don't expect to store previous value of read data, please change memory model directly.

3.2.3 Pattern of generation model including memory model

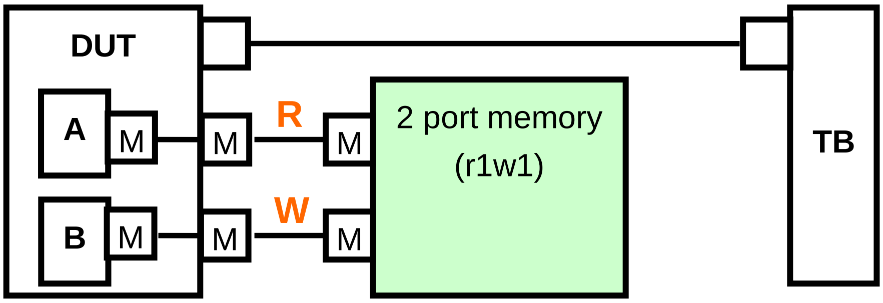
Ssgen can generate the design pattern including memory model as follows.

(1) When the DUT module accesses the single port memory



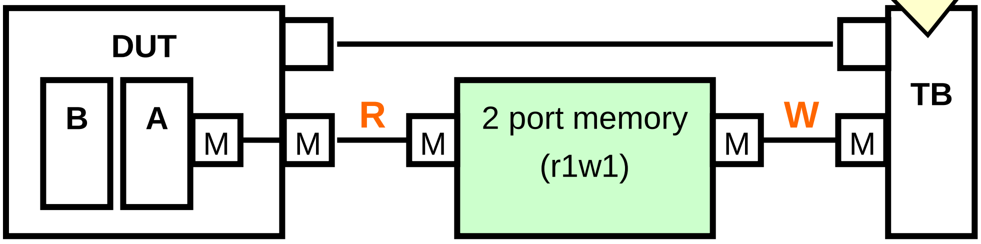
(2) When the DUT module Read/Write accesses 2 port memory

In the example of the following figure, supports the both cases when the clock of module A and the clock of module B are the same, and when the clock of module A and the clock of module B differ.



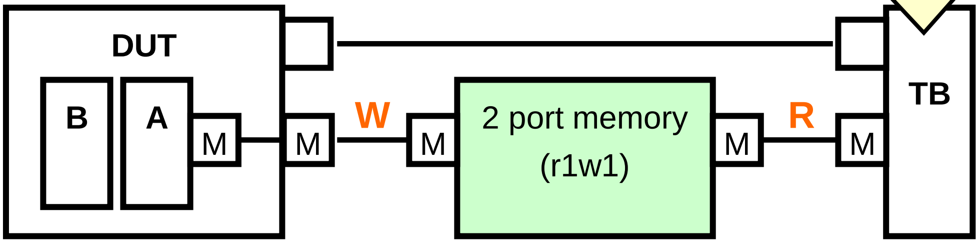
(3) When the DUT module Read accesses 2 port memory

Generate ports and functions about “memory write” in the testbench

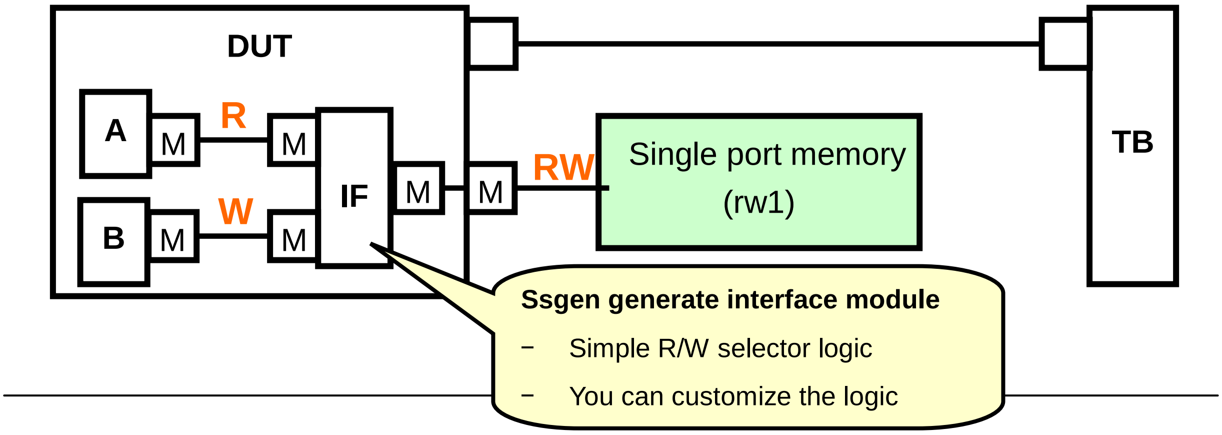


(4) When the DUT module Write accesses 2 port memory

Generate ports and functions about “memory read” in the testbench



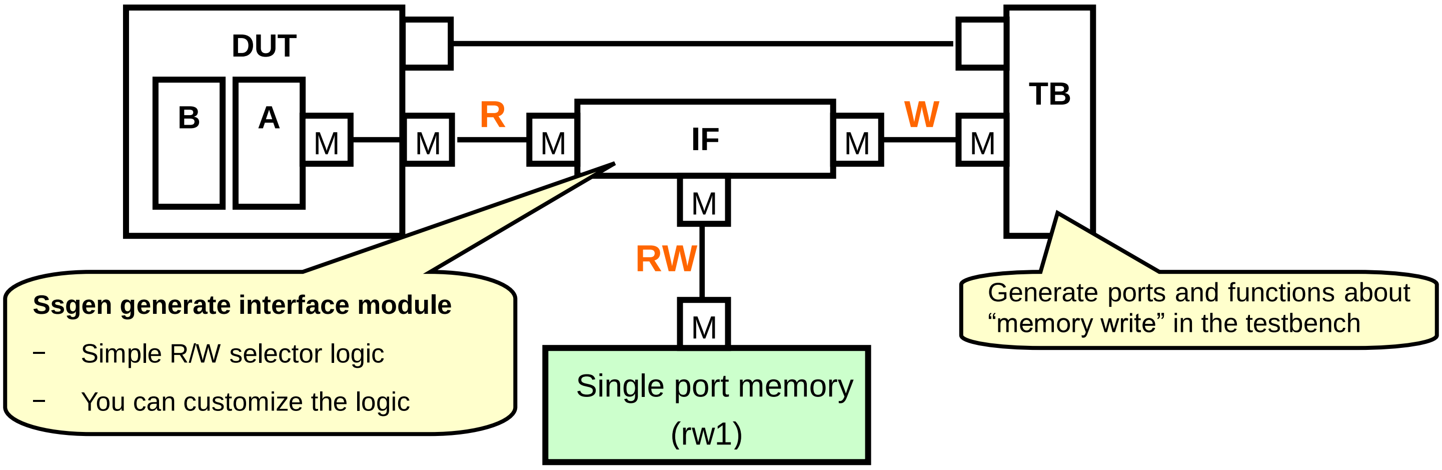
(5) When the DUT module Read/Write accesses single port memory



Ssgen generate interface module

- Simple R/W selector logic
- You can customize the logic

(6) When the DUT module Read accesses single port memory

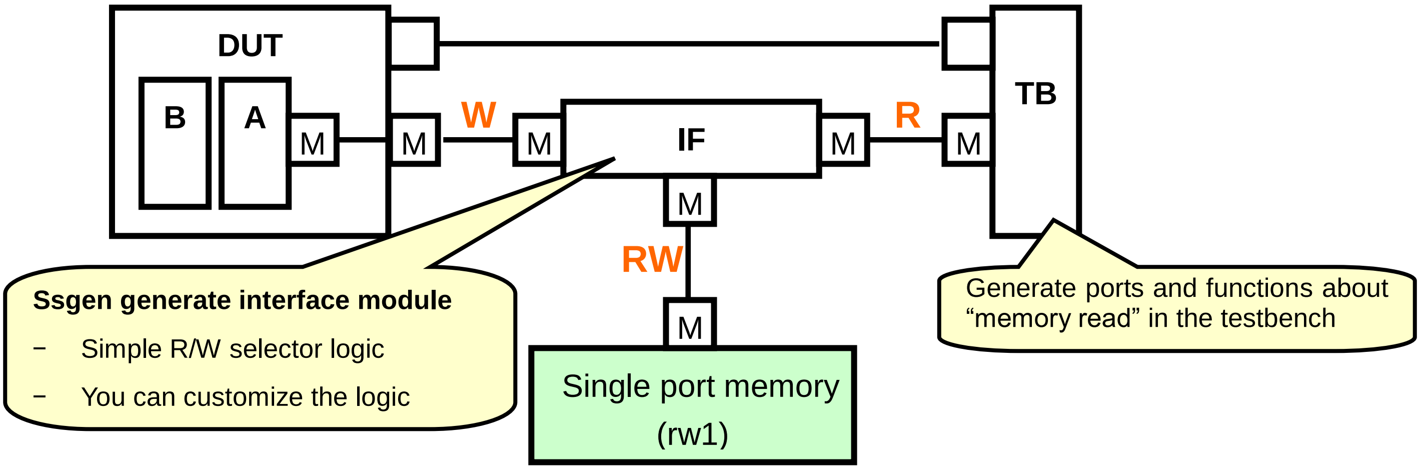


Ssgen generate interface module

- Simple R/W selector logic
- You can customize the logic

Generate ports and functions about "memory write" in the testbench

(7) When the DUT module Write accesses single port memory



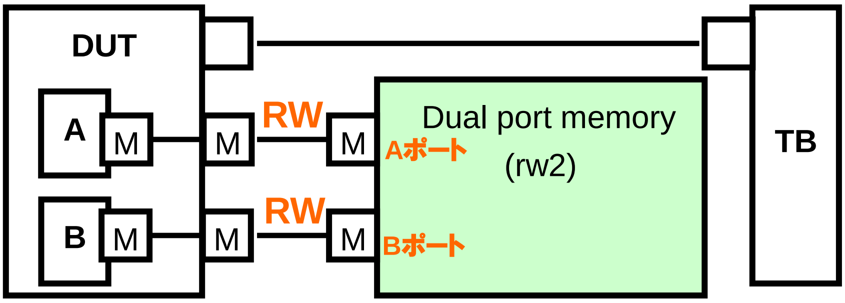
Ssgen generate interface module

- Simple R/W selector logic
- You can customize the logic

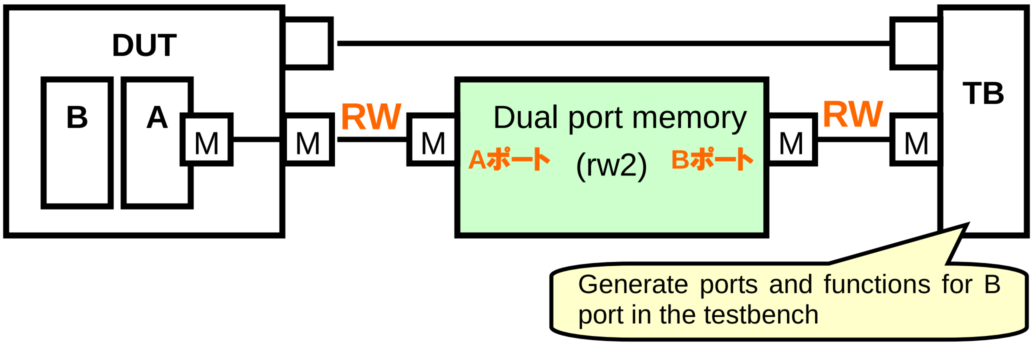
Generate ports and functions about "memory read" in the testbench

(8) When the DUT modules Read/Write access dual port memory

In the example of the following figure, supports the both cases when the clock of module A and the clock of module B are the same, and when the clock of module A and the clock of module B differ.



(9) When a DUT module Read/Write accesses dual port memory



Generate ports and functions for B port in the testbench

Ssgen does not support design models including memory model as follows.

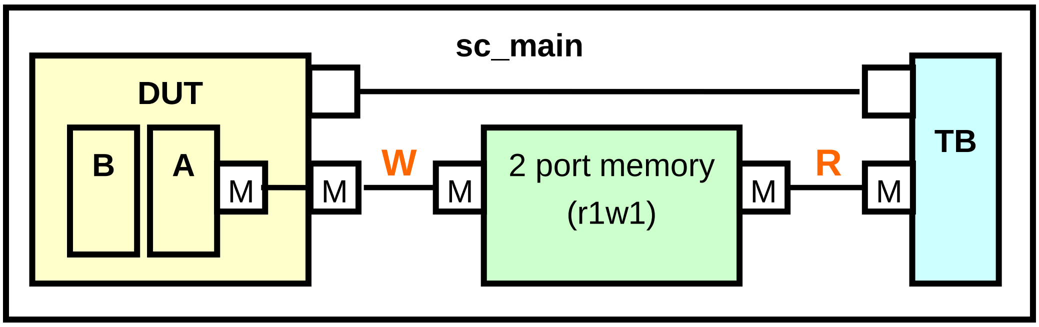
・Two or more modules access one single port memory Read or 2 port memory Read.

・Two or more modules access one single port memory Write or 2 port memory Write.

・One module which has logic has both A and B port for a dual port memory

4. Design procedure using ssgen

This chapter shows the design procedure using ssgen. Here, the design procedure of the following design pattern is described. In this example, OSCI-Sim is used for the SystemC simulation.
Please refer to chapter 5 for details of the command line option of ssgen.



(1) Preparation for module definition file and hierarchy definition file

You should prepare module definition file A.in of module A, module definition file B.in of module B, and hierarchy definition file DUT.in.

(2) Generation model of module A (module generation mode)

When ssgen generates the model of module A, generates also the CtoS script of module A (-ctos).

```
 %s> ssgen.pl -in A.in -ctos
        Output file: A.h/A.cpp (model of module A)
                     run_ctos_A.sh/ctos_A.tcl (CtoS script of module A)
                     tb_A.h/tb_A.cpp (testbench for module A but not be used for this example)
```

(3) Generation model of module B (module generation mode)

When ssgen generates the model of module B, generates also the CtoS script of module B (-ctos).

```
 %s> ssgen.pl -in B.in -ctos
        Output file: B.h/B.cpp (model of module B)
                     run_ctos_B.sh/ctos_B.tcl (CtoS script of module B)
                     tb_B.h/tb_B.cpp (testbench for module B but not be used for this example)
```

(4) Generation hierarchical module DUT (hierarchy generation mode)

When ssgen generates hierarchical module DUT, generates also the CtoS script of module DUT (-ctos). In order to simulate it in the DUT hierarchy, generates the simulation script for OSCI-Sim (-osci). Moreover, generates 2 port memory model because module A of DUT's sub module have the write access to two port memory (-mem).

```
%s> ssgen.pl -in DUT.in -ctos --osci -mem
            Output file: DUT.h/DUT.cpp (SystemC description of module DUT)
                         run_ctos_DUT.sh/ctos_DUT.tcl (CtoS script of module DUT)
                         tb_DUT.h/tb_DUT.cpp/main_DUT.cpp (test bench for module DUT)
                         Makefile/Makefile.defs (OSCI-Sim script)
                         mem_1r1w.h (2 port memory model)
                         mod_DUT.in (module definition file for DUT hierarchy but not be used for
                                     this example).
```

(5) Implement module function

Please implement module function on A.h/A.cpp and B.h/B.cpp. Moreover, implement testbench function on tb_DUT.h/tb_DUT.cpp/ main_DUT.cpp.

(6) Execution OSCI-Sim

Please verify the implemented function with OSCI-Sim.

```
%s> make
%s> run.exe
```

When you want to dump signal value to VCD file, use –vcd option (run.exe –vcd)

When you want to specify the hierarchy level of dump, use the number (run.exe –vcd 1)

(7) Generation the RTL description by CtoS.

Please generate the RTL description by CtoS after you have finished verification of the module function with OSCI-Sim. RTL description (A.v,B.v, DUT.v) of each module is generated by using the CtoS script generated at step (2), (3) and (4).

```
%s> run_ctos_A.sh
%s> run_ctos_B.sh
%s> run_ctos_DUT.sh
```

5. Command line option

The below table shows the list of the command line options that can be specified when execute ssgen.pl.

Option	Content of processing
-in *filename*	Specify the module definition file or the hierarchy definition file. (mandatory option)
-out *outdir*	Specify the directory path where Ssgen generates output files in. If not specify this option, Ssgen generates output files in current directory. (It is impossible to specify un-existed path)
-mem	Generate memory model (mem_rw1.h, mem_r1w1.h, mem_r1w1_2clk.h).
-ctos	Generate CtoS script for synthesis.
-osci	Generate Makefile for the OSCI compile.
-vcs [*memsize osname*]	Generate the script file for the VCS-MX simulation. If necessary, specify the memory size and OS name for bs command. (Default memory size is 500, and OS name is RHEL5.)
-ies [*memsize osname*]	Generate the script file for the IES simulation. If necessary, specify the memory size and OS name for bs command. (Default is memory size is 500, and OS name is RHEL5.)
-checker	Generate the script files for 1Team:System and SSChecker.
-slec	Generate the script file for SLEC (Ignore this option when input file is the hierarchy definition file)
-only_script	Generate only script file. Specify with needed script options. (some of -ctos/-osci/-vcs/-ies)
-notb	Stop generating testbench files.
-include *filename*	Specify the other files for include (SSGEN treats this option same as `include command which is described later). Multi specified OK.
-subdir	Generates files in sub-directories src/ SystemC module files tb/ Testbench files(when not specifying -notb) ctos/ CtoS script(when specifying -ctos) 1team/ 1Team:System script(when specifying -checker) sschecker/ SSChecker script(when specifying -checker) gcc/ OSCI simulation script(when specifying -osci) vcs/ VCS-MX script(when specifying -vcs) ies/ IES script(when specifying -ies) slec/ SLEC script(when specifying -slec)

6. Command

This chapter explains the command specified in the module definition file and the hierarchy definition file of the input file of ssgen. The commands used in the module generation mode and the commands used in the hierarchy generation mode are prepared separately.

6.1 Module generation command

The below table shows the list of the command used in the module generation mode. Please specify a necessary command with the module definition file, and input it to ssgen.

Command name	Explanation
changelog	Generate description of update history (summary log for changing).
style_module	Specify the style of module and constructor definition. sc:"SC_MODULE" and "SC_CTOR" c++: using class definition If not specify this command, the default style is "sc".
space_indent	Specify the number of indent spaces of generation description. If not specify this command, the default number is 4.
env_systemc/env_ssgen/env_vcs/env_ies/env_ctos/env_vcs_gcc/env_1team/env_sschecker/env_slec	Specify the environment setting path of SystemC/ssgen/ VCS-MX/IES/CtoS/ 1Team:System/SSChecker/SLEC ・If not specify these commands, the default paths are as the EWS environment of REL / Musashi.
mem_suffix	Specify the suffix configuration file for specifying suffix of memory port.
style_alloc	Specify the style of module instantiation in sc_main. static: static instantiation dynamic: dynamic instantiation If not specify this command, the default style is "static".
vcd_trace	Control generation of sc_trace descriptions for ports and signals on: generate sc_trace descriptions off: not generate sc_trace descriptions If not specify this command, sc_trace descriptions are generated in default configuration.
`include	Specify the other module definition file for including.
`define	Specify the define macro that becomes effective only in the input file.
`ifdef/`else/ `endif	Specify `ifdef syntax that becomes effective only in input file.
#include	Generate the include description of the header file.
#define	Generate the define macro description.
#ifdef/#endif	Generate the #ifdef /#endif description.
module	Specify module name.
clock	Generate clock port.
areset	Generate asynchronous reset port.
sreset	Generate synchronous reset port.
soft_reset	Generate internal signal for soft reset.
uinN/sinN	Generate input port (sc_in). ・uinN generates the port with the sc_uint type, sinN generates with sc_int type. ・The bit width is specified by N(If N is 'b', bool type is adopted).
uoutN/soutN	Generate output port (sc_out). ・The specification of sign type and bit width is the same as uinN/sinN command. ・It is possible to set initial value by '=' (When not set, default value is 0).

Command name	Explanation
uregN/sregN	Generate internal signal (sc_signal). ･The specification of sign type and bit width and initial value is the same as uoutN/soutN command.
uvarN/svarN	Generate member variable with SystemC data type. ･The specification of sign type and bit width and initial value is the same as uoutN/soutN command. ･It is possible to set const type by “const” identifier.
char/uchar/short/ ushort/int/uint	Generate C data type member variable. ･char/short/int command generates char/short/int type variable. ･The command started by “u" generates it with the unsigned type. ･The specification of initial value and const type is the same as uvarN/svarN command.
uevN/sevN	Generate extend SystemC data type member variable. ･The specification of sign type and bit width and initial value is the same as uoutN/soutN command.
umem/smem	Generate memory access description for both port access and array access. ･umem uses sc_uint type for data, and smem uses sc_int types.
cthread	Generate SC_CTHREAD.
method	Generate SC_METHOD. ･It is possible to specify the sensitivity list.
func	Generate the member function.
!-- --!	Free area ･The part enclosed with this command is output to the output header file as it is.
//	Comment ･The line started by “//" is treated as comment and is skipped. ･The comment described in the each command by "//" is output to the output file as comment.

6.1.1 Reference of module generation command

Details of each command are described as follows.

changelog

Describe update history (summary log for changing). The described history is output to the head of the header file.

Format:

changelog - *history*

Example:

changelog - 2011/1/1 1 Renesas new

style_module

Specify the style of module and constructor definition (SC_MODULE/SC_CTOR definition or general class definition). When specify "sc", SC_MODULE/SC_CTOR definition are used, when specify “c++”, the general class definition is used. It is also possible to use "SC" and "C++" instead of “sc” and “c++”.

When this command is unspecified, “sc”(the SC_MODULE/SC_CTOR definition) is adopted.

Format:

style_module {sc|c++}

Example:

style_module sc

space_indent

Specify the number of indent spaces of generation description.

When this command is unspecified, the default number is 4.

Format:

space_indent *SpaceNum*

Example:

space_indent 2

env_systemc

Specify the path of SystemC library. The path specified as this command is set in Makefile.defs generated at the time of the command line option "-osci" specification. It is not checked whether the path specified as this command exists or not.

”/common/appl/Renesas/SystemC/SystemC-2.2” is adopted at the time of un-specifying of this command.

Format:

env_systemc *SystemCLibraryPath*

Example:

env_systemc /RVC/SystemC/SystemC-2.2

env_ssgen

Specify the path of ssgen library file. The path specified as this command is set in compile option “-I” in script files. It is not checked whether the path specified as this command exists or not.

”/common/appl/Renesas/SystemC/utility/ssgen” is adopted at the time of un-specifying of this command.

Format:

env_ssgen *SsgenLibraryFilePath*

Example:

env_ssgen /RVC/SystemC/ssgen

env_vcs

Specify the path of the environment configuration file of VCS-MX. The path specified as this command is set in run_vcs.sh generated at the time of the command line option "-vcs" specification. It is not checked whether the path specified as this command exists or not.

”/common/appl/dotfiles/vcs_mx.CSHRC_2011.12-sp1-1” is adopted at the time of un-specifying of this command.

Format:

env_vcs *VCSEnvFile*

Example:

env_vcs /RVC/dotfiles/vcs_mx.CSHRC_2011.12-sp1-1

env_ies

Specify the path of the environment configuration file of IES. The path specified as this command is set in run_ies.sh generated at the time of the command line option "-ies" specification. It is not checked whether the path specified as this command exists or not.

"/common/appl/dotfiles/cadence.CSHRC_ius12.10s004" is adopted at the time of un-specifying of this command.

Format:

env_ies *IESEnvFile*

Example:

env_ies /RVC/dotfiles/cadence.CSHRC_ius12.10s004

env_ctos

Specify the path of the environment configuration file of CtoS. The path specified as this command is set in run_ctos_*modulename*.sh generated at the time of the command line option "-ctos" specification. It is not checked whether the path specified as this command exists or not.

"/common/appl/dotfiles/cadence.CSHRC_ctos_v12.20-s201" is adopted at the time of un-specifying of this command.

Format:

env_ctos *CtoSEnvFile*

Example:

env_ctos /RVC/dotfiles/cadence.CSHRC_ctos_v12.10-s201

env_vcs_gcc

Specify the path of the environment configuration file of GCC used by VCS-MX. The path specified as this command is set in vcs_lsfsh_sc and vcs_lsfsh_rtl generated at the time of the command line option "-vcs" specification. It is not checked whether the path specified as this command exists or not.

"/common/appl/Synopsys/vg_gnu_package/2011.12/linux/source_me_gcc4_32.csh" is adopted at the time of un-specifying of this command.

Format:

env_vcs_gcc *GCCEnvFile*

Example:

env_vcs_gcc /RVC/VG_GNU_PACKAGE/linux/source_me_gcc3_32.csh

env_1team

Specify the path of the environment configuration file of 1Team:System. The path specified as this command is set in run_1team.sh generated at the time of the command line option "-checker" specification. It is not checked whether the path specified as this command exists or not.

”/common/appl/dotfiles/1TeamSystem.CSHRC_1.16.7” is adopted at the time of un-specifying of this command.

Format:

env_1team *1TeamSystemEnvFile*

Example:

env_ctos /RVC/dotfiles/1TeamSystem.CSHRC_1.16.7

env_sschecker

Specify the path of the script file of SSChecker. The path specified as this command is set in run_sschecker_*modulename*.sh generated at the time of the command line option "-checker" specification. It is not checked whether the path specified as this command exists or not.

”/common/appl/Renesas/SystemC/utility/SSChecker” is adopted at the time of un-specifying of this command.

Format:

env_sschecker *SSCheckerFilePath*

Example:

env_sschecker /RVC/SystemC/sschecker

env_slec

Specify the path of the environment configuration file of SLEC. The path specified as this command is set in run_slec_*modulename*_sc.sh and run_slec_*modulename*_eq.sh generated at the time of the command line option "-slec" specification. It is not checked whether the path specified as this command exists or not.

”/common/appl/dotfiles/slec.CSHRC_7.1-prod-5” is adopted at the time of un-specifying of this command.

Format:

env_slec *SLECEnvFile*

Example:

env_slec /RVC/dotfiles/slec.CSHRC_7.1-prod-5

mem_suffix

Specify the suffix configuration file for specifying suffix of memory port. Suffix settings which are specified in the suffix configuration file are applied to memory ports generated by all {u|s}mem commands. "-suffix" option of {u|s}mem command (described later) has priority over this command. Format of suffix configuration file and port kinds are as follows. The port which does not have specification within suffix configuration file adopts default suffix.

Memory type	Port kinds	Port identifier	Default suffix
Single port memory (rw1 or rw1:r or rw1:w)	Address	rw1_ad	ad1
	Write data	rw1_wd	wd1
	Write enable	rw1_we	we1
	Read data	rw1_rd	rd1
	Chip select	rw1_cs	cs1
	Write address	rw1_wa	wa1
	Read address	rw1_ra	ra1
	Read enable	rw1_re	re1
2port memory (r1w1:r or r1w1:w)	Write address	r1w1_wa	wa1
	Write data	r1w1_wd	wd1
	Write enable	r1w1_we	we1
	Read address	r1w1_ra	ra1
	Read data	r1w1_rd	rd1
	Read enable	r1w1_re	re1
	Chip select	r1w1_cs	cs1
Dual port memory (rw2:a or rw2:b)	Address (A port)	rw2a_ad	ad1
	Write data (A port)	rw2a_wd	wd1
	Write enable (A port)	rw2a_we	we1
	Read data (A port)	rw2a_rd	rd1
	Chip select (A port)	rw2a_cs	cs1
	Address (B port)	rw2b_ad	ad2
	Write data (B port)	rw2b_wd	wd2
	Write enable (B port)	rw2b_we	we2
	Read data (B port)	rw2b_rd	rd2
	Chip select (B port)	rw2b_cs	cs2

Format:

```
 Port identifier=Suffix
```

Example:

```
r1w1_wa=waddr
r1w1_wd=wdata
r1w1_we=wreq
r1w1_cs=wcs
```

Format:

mem_suffix *suffixfile*

Example:

mem_suffix suffix.txt

style_alloc

Specify the style of module (DUT and testbench) instantiation in sc_main. When specify "static", static instantiation is used, when specify "dynamic", dynamic instantiation is used. If stack overflow is caused by module having large logic, we recommend using this command with “dynamic”.

When this command is unspecified, “static” is adopted.

Format:

style_alloc {static|dynamic}

Example:

style_alloc static

vcd_trace

Control generation of sc_trace descriptions for ports and signals. When specify “on”, sc_trace descriptions are generated, when specify “off”, sc_trace descriptions are not generated. If there is much number of signals in design, we recommend that you specify this command with “off” in module whose signals are unnecessary to be dumped in VCD trace file, because VCD trace works wrongly if the number of signals is more than 17576 in a design.

When this command is unspecified, “on” is adopted.

Format:

vcd_trace {on|off}

Example:

vcd_trace off

`include

Include the other module definition file. Do not specify module command and cthread command in an include file which is specified by this command.

Format:

`include *filename*

Example:

`include common.in

`include ../inc/common.in

`include /ssgen/inc/common.in

Notes:

・Max nest level of include file is only one.

`define

Specify the define macro name that becomes effective only in the input file. The defined macro name can be used for `ifdef command. Also, the defined macro name can be used for some command parameters (refer to 7.).

Format:

`define *macroname*

Example:

```
`define MODE1
```

Notes:

･Please do not define "TESTBENCH" macro because it is a reserved word.

`ifdef/`else/`endif

Specify `ifdef syntax that becomes effective only in the input file. Only an effective range is taken as an input command in `ifdef-`else-`endif. Please set macro name to `ifdef command.

The command enclosed with the “TESTBENCH” macro is treated as a command for the testbench. The command that becomes effective in the “TESTBENCH” macro is only uregN, sregN, uvarN, svarN, char, uchar, short, ushort, int, uint, func, and a free area (!-- --!).

When you use a macro except “TESTBENCH”, you can specify areset, sreset, soft_reset, uinN, sinN, uoutN, soutN, uregN, sregN, uvarN, svarN, char, uchar, short, ushort, int, uint, umem, smem, method, func, and a free area(!-- --!) in the macro.

When you want to set a macro name effective, please define the macro name by the `define command before use it by `ifdef command. (but “TESTBENCH” must not be defined because of the reserved word).

Format:

`ifdef *macroname*

Example:

```
`ifdef MODE1
ureg8 tmp // When `define MODE1 is defined, it uses it.
`else
sreg8 tmp // When macros other than MODE1 are defined by the `define command, it uses it.
`endif
`ifdef TESTBENCH
 ureg8 tb_tmp // Command for test bench
 `endif
```

Notes:

･It is prohibited to make the nesting description of `ifdef and #ifdef.

#include

Generate the include description of the header file.

Format:

#include “*filename*”

Example:

```
#include “common.h"
```

#include “../inc/common.h"

#include “/ssgen/inc/common.h"

#include common.h // It encloses with """ in the output file like #include “common.h”.

Notes:

・Ssgen does not check whether the specified header file exists or not.

#define

Generate the define macro description.

Format:

#define *macroname*

Example:

#define MODE1

#define MAX 512

#define ADD(a, b) ((a)+(b))

Notes:

・To define multi line macro causes an error.

#ifdef/#endif

Generate #ifdef/#endif description. You can specify only "_DEBUG*" macro or “_SLEC_BBOX” macro for #ifdef command. To specify other macro name for #ifdef causes an error.

The description part where _DEBUG* becomes effective is generated as a description for debugging. The commands that can be used in the _DEBUG* macro are only uregN, sregN, uvarN, svarN, char, uchar, short, ushort, int, uint, func, and a free area (!-- --!).

The description part where _SLEC_BBOX becomes effective is generated as a description for SystemC-RTL equivalence check using function blackboxing method. The commands that can be used in the _SLEC_BBOX macro are only uinN, sinN, uoutN, and soutN.

Format:

#ifdef _DEBUG_SIM

#endif

Example:

#ifdef _DEBUG_SIM

sreg8 tmp // output as a description for the debugging.

#endif

Notes:

・It is prohibited to make the nesting description of #ifdef and `ifdef.

module

Specify the module name. This command must be always defined only once.

Format:

module *modulename*

Example:

module test

clock

Generate a clock port with pos edge. This command must be always defined only once.

Format:

clock *clockname*

Example:

clock clk

areset

Generate an asynchronous reset port. The active edge is specified by pos or neg. Please do not define this command 2 or more times.

Ssgen always generates an asynchronous reset as a top priority reset regardless of the order of specification of resets (areset, sreset and soft_reset) in the module definition file.

Format:

areset *resetname* {pos|neg}

Example:

areset rst_n neg // Asynchronous reset of negative edge

sreset

Generate a synchronous reset port. The active edge is specified by pos or neg. Please specify this command when you do not generate an asynchronous reset by areset command.

Format:

sreset *resetname* {pos|neg}

Example:

sreset rst pos // Synchronous reset of positive edge

soft_reset

Generate an internal signal for soft reset. The active edge is specified by pos or neg.

If Specify "-header" option, the body of SC_METHOD function generating soft reset, and member function for soft reset trigger is output to header file instead of source file.

Format:

soft_reset *resetname* {pos|neg} [-header]

Example:

soft_reset rst pos // Inner Reset of positive edge

uinN/sinN

Generate an input port (sc_in). uinN generates the port with the sc_uint type, sinN generates with sc_int type. The bit width is specified by N. When the value of 65 or more is set, it becomes sc_bigint/sc_biguint type. if "b" is specified for N, it becomes bool type.

The array port can be generated by specifying the port name by the array form. Ssgen supports one-dimensional array and two-dimensional array.

Ssgen does not generate the description for VCD dump when you specify 1000 or more value to bit width, because SystemC library does not support VCP dump of signal which has 1000 or more bit width. uoutN, soutN, uregN and sregN are also same.

Format:

{u|s}inN *inputname*

Example:

```
uin8 inp1 // Input port of sc_uint<8 > type
sin8 inp2 // Input port of sc_int<8 > type
uinb inp3 // Input port of bool type
uin65 inp4 // Input port of sc_biguint<65 > type
uin16 inp5[16][16] // Port of input of two dimensional array of sc_uint<16 > type
```

uoutN/soutN

Generate an output port (sc_out). The specification of sign type(u or s) and bit width and array form is the same as uinN/sinN command.

It is possible to set an initial value by “-init” option. When an initial value is not specified, it is initialized by default "0". It is also possible to specify "no initialization" by specifying "n" or “N”. In the case of array port, you can specify the initialization by each element or all elements together.

By using "-th" option, the thread (name specified by the cthread command) which initializes output ports can be specified. When there is no specification of this option, they are initialized by the first defined thread.

By using “-range_check” option, the dummy coverage codes, which are used for checking the range of value of the variable, are generated. These codes are generated in “my_wait” function. Ssgen decides maximum value and minimum value by the bit width of the variable, but you can specify maximum value and minimum value by “-max”/”-min” options. When you specify “-max”/”-min” options, ssgen also generates assertion codes (SSGEN_ASSERT) which checks that the value of variable is over the maximum value / under the minimum value. “-range_check” option is available for 1-64 bit non-array variable.

Format:

{u|s} outN *outputname* [-init *initialvalue*] [-th *threadname*]
[-range_check [-max *maximum*] [-min *minimum*]]

Example:

uout8 outp1 // initial value is 0.

uout8 outp2 -init 1 // initial value is 1.

uout8 outp3 -init n // no initialization

uout8 outp4[4] // all elements are initialized by 0.

uout8 outp5[4] –init 1 // all elements are initialized to 1.

uout8 outp6[4] -init {1, 2, 3, 4} // initial value for the each element is set.

uout8 outp7[4] -init {1, 2, 3} // outp7[3] is initialized to 3.

uout8 outp8 –th thread_sub // initialized by 0 inside thread_sub

uout8 outp9 –range_check // generate dummy coverage codes for
checking the range of value (0-255)

uout8 outp10 –range_check –max 128 –min 16 // generate dummy coverage codes for
checking the range of value (16-128)

Notes:

・Ssgen does not check an illegal initial value (the value is specified with more than the bit width, or illegal format for array initialization, etc.). For an example, when specify "uout8 outp7[4] = {1, 2, 3}", outp7[3] is initialized by "3"(The value specified at the end is used for over range element).

・Specify "n" to "-init" option when an output port is assigned value in method. If not specify "n", initialization of the output port is generated in reset block of thread, multi-drive error is occurred in simulation and high-level synthesis.

・Ssgen does not check an illegal maximum value("-max" option) / minimum value("-min" option).

uregN/sregN

Generate an internal signal (sc_signal). The specification of sign type(u or s) and bit width and array form and initial value and thread and range check is the same as uoutN/soutN command.

Format:

{u|s}regN *signalname* [-init *initialvalue*] [-th *threadname*]
[-range_check [-max *maximum*] [-min *minimum*]]

Example:

ureg8 sig1 // initial value is 0.

ureg8 sig2[2][2] -init {{1,2}, {3,4}} // An initial value for the each element is set.

ureg8 sig3[2] –init 1 –th thread_sub // all elements are initialized by 1 inside thread_sub

Notes:

・Ssgen does not check an illegal initial value (the value is specified with more than the bit width, or illegal format for array initialization, etc.).

・Specify "n" to "-init" option when a signal is assigned value in method. If not specify "n", initialization of the signal is generated in reset block of thread, multi-drive error is occurred in simulation and high-level synthesis.

uvarN/svarN

Generate a member variable with SystemC data type. The specification of sign type(u or s) and bit width and array form and initial value and thread and range check is the same as uoutN/soutN command.

By using "-var2reg", descriptions to assist the script "var2reg.pl" are generated. About details of "var2reg.pl" and this option, please contact FEDT.

By using "-debug_trace", SystemC data type variable (var) and sc_signal variable (var_dbg) for VCD trace are generated. Because SystemC data type variable cannot be dumped in VCD trace file, if you want to confirm the value of the variable which is defined by this option in VCD trace file. The sc_signal variable (var_dbg) which is generated by this option is valid only when _DEBUG_SIM macro is specified in compiling. If you want to change name of macro, please specify "-debug_macro" option too. And add "_DEBUG" to macro name as the prefix.

It is possible to set const type by "const" identifier and that makes the member variable of the static const type. You can generate three-dimension array only when you specify const. Initialization description of static const type member variable is certainly generated in source file. If you set "const", please do not specify "n" (no initialization). Moreover, at the time of const specification, specifications of "-th", "-range_check", "-var2reg" and "debug_trace", are ignored.

Format:

[const] {u|s} varN *variablename* [-int *initialvalue*] [-th *threadname*]
[-range_check [-max *maximum*] [-min *minimum*]]
[-var2reg] [-debug_trace [-debug_macro *macroname*]]

Example:

uvar8 tmp1 // initial value is 0.

const uvar8 tmp2 -int 1 // An initial value is 1 declared in the static const type.

uvar8 mVar –debug_trace // sc_uint<8> mVar and sc_signal <sc_uint<8>> mVar_dbg are generated

Notes:

･Ssgen does not check an illegal initial value (the value is specified with more than the bit width, or illegal format for array initialization, etc.).

･Specify "n" to "-init" option when a member variable is assigned value in method. If not specify "n", initialization of the member variable is generated in reset block of thread, racing is occurred.

char/uchar/short/ushort/int/uint

Generate a member variable with C data type. char/short/int command generates char/short/int type variable, and the command started by "u" generates it with the unsigned type. The specification of array form and initial value and "const" and thread and range check and var2reg and debug_trace is the same as uvarN/svarN command.

Format:

[const] {char/uchar/short/ushort/int/uint} *variablename* [-init *initialvalue*] [-th *threadname*]
[-range_check [-max *maximum*] [-min *minimum*]]
[-var2reg] [-debug_trace [-debug_macro *macroname*]]

Example:

char tmp1 // initial value is 0.

const ushort tmp2 // It declares as static const type.

Notes:

･Ssgen does not check an illegal initial value (the value is specified with more than the bit width, or illegal format for array initialization, etc.).

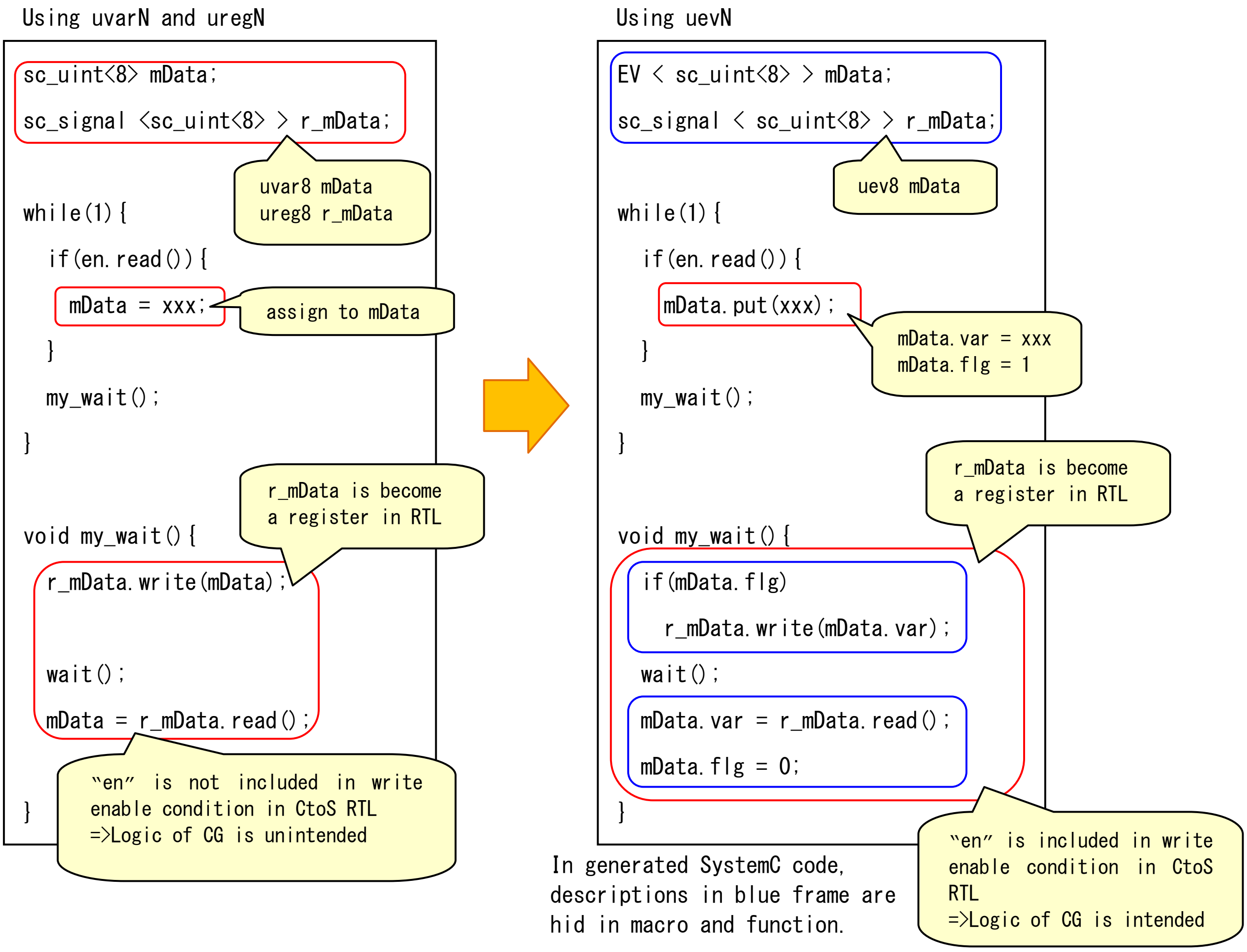
･Specify “n” to “-init” option when a member variable is assigned value in method. If not specify “n”, initialization of the member variable is generated in reset block of thread, racing is occurred.

uevN/sevN

Generate a member variable with extend SystemC data type. The specification of sign type(u or s) and bit width and array form and initial value and thread and range check is the same as uoutN/soutN command.

By this command, a variable (var) whose type is EV and sc_signal variable (r_var) are generated. EV is struct type and defined in ssgenlib.h. You should describe “var.get()” when you want to refer the value of EV variable, also you should describe “var.put(xxx)” when you want to assign a value to EV variable. You cannot access to EV variable by using “=”.

If you want to keep register configuration between SystemC and CtoS RTL, we recommend that you use this command. You can keep register configuration between SystemC and CtoS RTL by using uvarN/svarN and uregN/sregN, but this method is unfitted low-power design.



Using uvarN and uregN

```
sc_uint<8> mData;
sc_signal <sc_uint<8> > r_mData;

while(1){
  if(en.read()){
    mData = xxx;
  }
  my_wait();
}

void my_wait(){
  r_mData.write(mData);

  wait();
  mData = r_mData.read();
}
```

Using uevN

```
EV < sc_uint<8> > mData;
sc_signal < sc_uint<8> > r_mData;

while(1){
  if(en.read()){
    mData.put(xxx);
  }
  my_wait();
}

void my_wait(){
  if(mData.flg)
    r_mData.write(mData.var);
  wait();
  mData.var = r_mData.read();
  mData.flg = 0;
}
```

uvar8 mData
ureg8 r_mData
assign to mData
r_mData is become a register in RTL
"en" is not included in write enable condition in CtoS RTL
=>Logic of CG is unintended
uev8 mData
mData.var = xxx
mData.flg = 1
r_mData is become a register in RTL
"en" is included in write enable condition in CtoS RTL
=>Logic of CG is intended

In generated SystemC code, descriptions in blue frame are hid in macro and function.

Format:

{u|s}evN *variablename* [-init *initialvalue*] [-th *threadname*]

[-range_check [-max *maximum*] [-min *minimum*]]

Example:

uev8 tmp1 // EV variable tmp1 and sc_signal variable r_tmp1 are generated

Notes:

・Ssgen does not check an illegal initial value (the value is specified with more than the bit width, or illegal format for array initialization, etc.).

・You should not assign a value to the variables (EV variable and sc_signal variable) in SC_METHOD.

umem/smem

Generate a memory access description for both port access and array access. umem uses sc_uint/sc_biguint for data type, and smem uses sc_int/sc_bigint for data type.

Please be sure to specify data width (width), data size (size), name, memory type (rw1, r1w1:r, r1w1:w, rw1:r, rw1:w, rw2:a, rw2:b), and latency by always this order.

Please set the value of 2 or more to the data width and the size.

When you chose “rw1” as memory type, memory access description for single port memory is generated, when you chose “r1w1:r”, memory read access description for 2 port memory is generated, when you chose “r1w1:w”, memory write access description for 2 port memory is generated, when you chose “rw1:r”, memory read access description for single port memory is generated, when you chose “rw1:w”, memory write access description for single port memory, when you chose “rw2:a”, memory access description of A port for dual port memory is generated and when you chose “rw2:b”, memory access description of B port for dual port memory is generated.

Please set either of 1 and 2, 3 or 4 to latency value. The latency set up here is latency of the memory itself, and is not access latency. Therefore, in the case of FF output (memory access is implemented in SC_CTHREAD), It takes "latency cycle + 1" for the memory read access from DUT/testbench.

If specify "-init" option, the initial value of memory array can be set up. When constant value is specified, all elements of memory array are initialized with the value (note that the initial value is not checked even if incorrect). When rand is specified, each element of memory array is initialized with the random value which uses rand() function. When this option is not specified, all elements of memory array are initialized by 0.

If specify “-ponly” option, description of only port access to memory model is generated.

If specify "-header" option, the body of memory access functions is generated in header file instead of source file.

If specify “-we” option, it is possible to set active level (high/low) for write enable signal (we). If not specify, the active level is “high” as default. However, it is impossible to set this option to memory type r1w1:r and rw1:r.

If specify “-cs” option, it is possible to generate chip select signal (cs) and to set active level (high/low) for it. However, it is impossible to set this option to memory type r1w1:r and rw1:r and rw1:w.

If specify “-re” option, it is possible to generate read enable signal (re) and to set active level (high/low) for it. However, it is possible to set this option to memory type r1w1:r or rw1:r only.

If you specify low active to “-we”, “-cs” and “-re”, ssgen adds “_n” to enable port name as the suffix.

It is possible to add prefix and to specify suffix to memory port. Please use "-prefix" option for common prefix of input port and output port, and use "-iprefix" option for prefix of input port (read data port), and use "-oprefix" option for prefix of output port (except read data port). It is impossible to use "-iprefix/-oprefix" option with "-prefix" option. Please use "-suffix" option specified suffix configuration file for specifying suffix. This option has priority over mem_suffix command. Format of suffix configuration file and port kinds are same as mem_suffix command.

By using "-th" option, the thread (name specified by the cthread command) which initializes memory ports can be specified. When there is no specification of this option, they are initialized by the first defined thread.

When you chose “rw1:r” or “rw1:w”, ssgen also generates memory interface module. This module is an arbitration module which has a simple R/W select logic. The name of this module is *prefix_memoryname*_if (Ex. When umem 8 256 ram1 rw1:r 1 –prefix=m_, “m_ram1_if”). If specifying “-re” option to “rw1:r”, chip select signal is generated in the interface module. Also, the active level of chip select is set by “-re” option.

If you specify “-nowd” option, assignment (.write) of write data signal (wd) is not generated while declaration of write data signal is generated. This option is used when register of write data signal is shared between multiple memory accesses. If you specify “-noad” option, assignment (.write) of address signal (ad) is not generated while declaration of address signal is generated. This option is used when register of address signal is shared between multiple memory accesses. If you specify “-nowd” and “-noad”, declaration of write data signal and address signal are only generated.

Format:

{u|s}mem *width size memoryname* {rw1|r1w1:r|r1w1:w|rw1:r|rw1:w|rw2:a|rw2:b} *latency*

[-ponly] [-header]

[-init={*initval*|rand}] [-we={high|low}] [-cs [={high|low}]] [-re[={high|low}]]

[-prefix=*prefixname*] [-iprefix=*input-prefixname*] [-oprefix=*output-prefixname*] [–suffix=*suffixfile*]

[-th *threadname*] [-nowd] [-noad]

Example:

```
umem 8 256 ram1 rw1 1 // data width 8, size 256, and latency 1,
                         single port memory access
umem 8 256 ram2 rw1 1 –ponly –init=rand // single port memory access
                         only port access description, initialize by random value
umem 8 256 ram3 rw1 1 –cs=high // Single port memory access with chip select signal
                                (active level of chip select is HIGH)
umem 32 256 ram4 r1w1:w 3 –we=low // write access to two port memory
                                     active level of write enable is LOW (no chip select)
umem 32 256 ram5 r1w1:w 3 –cs=low –we=high // write access to two port memory
                                     having chip select signal (active level is LOW)
                                     active level of write enable is HIGH
umem 16 128 ram6 r1w1:r 2 –re=low // Read access to two port memory
                                    having read enable signal (active level is LOW)
umem 16 128 ram7 r1w1:r 2 -prefix=m_   // read access to two port memory
                                        "m _" is added to the prefix of the memory port
umem 8 256 ram8 rw1 1 –iprefix=i_ -oprefix=o_   // “i_” is added to the prefix of the input port
                                                   “o_” is added to the prefix of the output port
umem 8 256 ram9 rw1 1 –suffix=suf.txt   // suffix configuration file of memory port
umem 8 256 ram10 rw1 1 –th=thread_sub   // initialized in thread_sub thread
umem 8 256 ram11 rw1:r 1 –re // Read access to single port memory
umem 8 256 ram12 rw1:w 1 –we=low // Write access to single port memory
```

cthread

Generate a SC_CTHREAD. This command must be defined one or more times. If name of thread is specified without options, the necessary clock and reset information for defined SC_CTHREAD is based on all of the clock command, areset command, sreset command, and soft_reset command.

It is possible to specify clock and reset by “-clk” option and “-rst” option respectively. (But now multi-clock module is not supported, so “-clk” option doesn’t affect result.) It is possible to specify list of multiple resets to “-rst” option. If specify “n” to “-rst” option, it is possible to generate SC_CTHREAD with no reset. Basically priority of resets is decided by order of specification to “-rst” option, but asynchronous reset is treated as top priority reset if there is asynchronous reset. Please do not specify only soft reset to “-rst” option or do not specify same soft reset to multiple cthread commands.

If specify “-clk_edge” option, it is possible to specify the edge (pos|neg) of synchronous clock of the thread. When there is no specification of this option, the edge of synchronous clock is “pos”.

If specify “-pipe” option, it is possible to generate thread for pipeline synthesis.

- Macro “CtoS_MAIN_LOOP” is added to head of infinite loop. If specify “-pipe_macro” option, it is possible to change the macro name.
- Negate memory enable operation is generated at head of infinite loop instead of generated in my_wait function.
- “pipeline_loop” command is generated in CtoS script by comment. Here, CtoS can pipeline a loop between 2 and 3 stage in default. If specify “-pipe_max” option, it is possible to change the max stage number of this pipeline constraint.

If specify "-reset_header" option, the body of reset function is generated in header file instead of source file.

If specify “-wait_header” option, the body of my_wait function is generated in header file instead of source file.

Format:

cthread *threadname* [-clk *clockname*] [-clk_edge {pos|neg}]
[-rst *rstname1* [*rstname2* …]]
[-pipe [-pipe_macro *macroname*] [-pipe_max *MaxStageNumber*]]
[-reset_header] [-wait_header]

Example:

```
cthread main_cth // having all reset signals defined by areset/sreset/soft_reset commands
cthread sub_cth –rst rst2 rst3_n –pipe    // having rst2 and rst3_n as reset signals
                                             generated for pipeline synthesis
cthread sync_cth –rst n    // having no reset signal
```

method

Generate a SC_METHOD. The sensitivity list should be enumerated after method name definition. Please specify one or more port or signal that has been specified above this command to sensitivity list.

Also, you can specify sc_signal variable generated by uevN/sevN to sensitivity list. At that time, please specify the EV variable name (var) instead of sc_signal variable name (r_var).

Format:

method *methodname sensitivity1* [*sensitivity2* …]

Example:

```
method main_meth1 inp1
method main_meth2 inp2 sig1
```

func

Generate a member function. It is possible to use uvarN/svarN and uchar/ushort/uint command for the definition of return type and argument type. The type conversion is same as these command specification. And It is also possible to specify const for return type and argument type.

Format:

func *returntype functionname*([*argumenttype1 argumentname1*, …])

Example:

```
func void m_func1(svar8 arg1, char arg2) // Void type member function with two arguments
func uvar8 m_func2() // Member function of sc_uint<8 > type (The argument is none).
func svar65 m_func4(const uvar8 arg1) // Member function of sc_bigint type
                                        (Argument arg1 is const type).
```

Notes:

･Ssgen does not check the format of the function argument. The command specification such as not describing comma(,) between arguments, using invalid type for arguments are not detected as errors.

･Even if two functions have same name and same argument, ssgen does not detect an error of duplicate definitions.

Free area (!-- --!)

The part enclosed with this command (!-- --!) is output directly at the above constructer of the output header file as it is (even if the description causes compile error). Please use this command when you want to use the other descriptions that are not supported by ssgen commands (structure etc.).

Do not describe “!--“ and “--!” in one line. Please describe separately.

Example:

```
!--
// structure
struct st{
  int a;
  int b;
  st(): a(0), b(0) {}
};
--!
```

Comment (//)

You can describe comment by “//”. The line started by "//" is treated as comment, and is skipped. The comment described in the each command by "//" is output to the output file as comment.

Example:

```
// this is comment
sreset rst pos // sync reset
```

Notes:

・Please do not describe the comment by multi-byte codes(Japanese etc.).

6.1.2 Format of module definition file

Please specify the above-mentioned commands for the module definition file by the following specified order and specified numbers. However, you can arrange the order arbitrary in the range (1) or (2). If you violate this rule, ssgen will detect an error.

```
     changelog log_info    // unspecified OK, multi specified OK.
     style_module {sc|c++}    // unspecified OK, or should be specified only 1 time.
     space_indent spacenum    // unspecified OK, or should be specified only 1 time.
     env_name file    // name is systemc, ssgen, vcs, ies, ctos, vcs_gcc, 1team, sschecker, slec.
                      // unspecified OK, or should be specified as each only 1 time.
(1)  mem_suffix suffixfile // unspecified OK, or should be specified only 1 time.
     style_alloc {static|dynamic} // unspecified OK, or should be specified only 1 time.
     vcd_trace {on|off} // unspecified OK, or should be specified only 1 time.
     `include filename
     `define macro
                                unspecified OK, multi specified OK.
     #include filename
     #define macro
     module name // should be specified only 1 time.
     clock name // should be specified only 1 time (it's OK to specified in the file of `include command)
     areset name {pos|neg} // unspecified OK, or should be specified only 1 time.   should be specified 1
     sreset name {pos|neg} // unspecified OK, multi specified OK.                   or more time in total
     soft_reset name {pos|neg} [-header] // unspecified OK, multi specified OK.
     [const] type name [options] // type is {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN,
                                 // [u]char, [u]short, [u]int, {u|s}evN, multi specified OK.
(2)  {u|s}mem width size name {rw1|r1w1:r|r1w1:w} latency [options]    // unspecified OK, multi specified OK.
     cthread name [options] // should be specified 1 or more times.
     method name sensitivity1 [sensitivity2 …] // unspecified OK, multi specified OK.
        // (specify one or more port or signal that has been specified above this command to sensitivity list.)
     func returntype name([argument_description]) // unspecified OK, multi specified OK.
     !--
     Free Area // unspecified OK, multi specified OK.
     --!
```

You can use **`ifdef/`else/`endif** command for areset, sreset, soft_reset, *type*, {u|s}mem, method, func, free area (!-- --!) if necessary.
You can use **#ifdef/#endif** command for {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char, [u]short, [u]int, func, free area(!-- --!) if necessary.

6.2 Hierarchy generation command

The table below shows the list of the command used in the hierarchy generation mode. Please specify a necessary command with the hierarchy definition file, and input it to ssgen.

Command name	Explanation
changelog	Generate description of update history (summary log for changing). ・This command is same as one of module generation command.
style_module	Specify the style of module and constructor definition. ・This command is same as one of module generation command.
space_indent	Specify the number of indent spaces of generation description. ・This command is same as one of module generation command.
env_systemc/env_ssgen/env_vcs/env_ies/env_ctos/env_vcs_gcc/env_1team/env_sschecker/env_slec	Specify the environment setting path of SystemC/ssgen/ VCS-MX/IES/CtoS/ 1Team:System/SSChecker/SLEC ・This command is same as one of module generation command.
mem_suffix	Specify the suffix configuration file for specifying suffix of memory port. ・This command is same as one of module generation command.
style_alloc	Specify the style of module instantiation in sc_main. ・This command is same as one of module generation command.
vcd_trace	Control generation of sc_trace descriptions for ports and signals ・This command is same as one of module generation command.
`include	Specify the other module definition file for including. ・This command is same as one of module generation command.
`define	Specify the define macro that becomes effective only in the input file. ・This command is same as one of module generation command.
top	Specify hierarchical module name.
sub	Specify an internal module.
tap	Generate tap output port from internal signal between internal modules.
bind	Connect output port to input port between internal modules by optional name.
#ifdef/#endif	Generate the #ifdef /#endif description. ・This command is same as one of module generation command. ・But the macro name which can be used is only "_DEBUG*".
`ifdef/`endif	Specify `ifdef syntax that becomes effective only in input file. ・This command is same as one of module generation command. ・But the macro name which can be used is only "TESTBENCH".
uregN/sregN	Generate internal signal (sc_signal) ・This command is same as one of module generation command. ・But this command can be used in only "TESTBENCH" macro by `ifdef.
uvarN/svarN	Generate member variable with SystemC data type. ・This command is same as one of module generation command. ・But this command can be used in only "TESTBENCH" macro by `ifdef.
char/uchar/short/ ushort/int/uint	Generate C data type member variable. ・This command is same as one of module generation command. ・But this command can be used in only "TESTBENCH" macro by `ifdef.
func	Generate the member function. ・This command is same as one of module generation command. ・But this command can be used in only "TESTBENCH" macro by `ifdef.

Command name	Explanation
!-- --!	Free area ・This command is same as one of module generation command. ・But this command can be used in only “TESTBENCH” macro by `ifdef.
//	Comment ・This command is same as one of module generation command.

6.2.1 Reference of hierarchy generation command

Details of each command are described as follows. However, please refer to “ **6.1.1 Reference of module generation command**” for the commands that are described as “This command is same as one of module generation command.” in the above table.

top

Specify the name of hierarchical module which bundles internal modules. This command must be always defined only once. Ssgen judges module generation mode or hierarchy generation mode from the existence of this command.

Moreover, please specify this command ahead of tap command and sub command.

Format:

top *modulename*

Example:

top top_test

sub

Specify an internal module that becomes an instance of hierarchy module by using module definition file.

When specifying “-rtl” option, ssgen also generates instantiation descriptions for connecting with a Verilog module. In particular, ssgen generates the descriptions of array port connecting per element. If specifying “_MODE_RTL_*instancename*” macro in compiling, these descriptions are effective. Also, you can set this macro name by “-macro” option.

When specifying “-port_pfx” option, ssgen add the prefix to the name of all ports of the internal module.

If specifying this command in #ifdef area and the following conditions are satisfied, it is possible to generate debug module instantiation.

1) A module which is specified in sub command in #ifdef area has no output ports (no {u|s}out and {u|s}mem command).
2) A module definition file which is specified in sub command in #ifdef area is not specified outside #ifdef area.
3) All input ports of debug module must be bound to other port which generated by sub commands outside #ifdef area.

Format:

sub *moduledefinitionfile instancename* [*path of internal module SystemC header*]
[-rtl [–macro *macroname*]]
[-port_pfx *prefix*]

Example:

sub test1.in test1_ins // instance “module test1” as test1_ins.

sub test2.in test2_ins ../test2 // instance “module test2” as test2_ins.
The header file is in "../test2".

sub test3.in test3_ins /ssgen/test3 // instance “module test3” as test3_ins.
The header file is in "/ssgen/test3".

Notes:

・Even if file path specified in the third argument doesn't exist, ssgen does not detect an error.

tap

Generate tap output port from internal signal between internal modules. Also, it is possible to generate external port with different name from port of internal module. When you specify the array signal, please specify only signal name without array form. If specify “*instance*.*signalname*”, only a signal of specified instance is targeted. If specify “signalname”, “*signalname*” of all instances are targeted. When there is no specification of “*portname*”, “*signalname*” is adopted as name of generated output port.

Also, it is possible to generate external memory with different name from memory of internal module. In this case, you must specify the second argument as external memory name.

It is impossible to specify this command in #ifdef area.

Format:

tap [*instancename*.]*signalname* [*portname*]

tap [*instancename*.]*memoryname memoryname*

Example:

tap sig1 tap_out1 // Generate output port tap_out1 from internal signal sig1 that is the pair of the input
and the output between internal modules

tap sig2 tap_out2 // Generate output port tap_out2 (array) from internal signal sig2 (array)
The definition of the array form is unnecessary.

tap sig3 // Generate output port sig3 as same name from internal signal sig3
When same name, you can omit to specify output port name.

tap in1 inA // Generate external port inA from input port in1 of internal module.

tap mod_A.out1 outA // Generate external port outA from output port out1 of instance mod_A.

tap mod_A.ram ramA // Generate external memory ramA from memory ram1 of instance mod_A.

bind

Connect output port to input port between internal modules. When you specify the array signal, please specify only signal name without array form. If not specify any this commands, connect only pair of input and output having the same name.

Output port of start point is specified to the first argument, input port of end point is specified to the second argument, and then format of specification is “instancename.portname”. Name of signal between start point and end point is specified to the third argument, but it is optional. When there is no specification of “*signalname*” in the third argument, the name of start point is adopted as name of generated signal.

Please do not specify different signal name having the same start point. It is possible to specify a signal generated by bind command to tap command.

Also, it is possible to generate floating ports by specifying a constant value (0 or positive value) to the first argument or second argument. In this case, you must specify the third argument as floating port name.

It is possible to specify this command in #ifdef area.

Format:

bind *startinstancename.outportname endinstancename.inportname* [*signalname*]

bind *constant endinstancename*.*inportname signalname*

bind *startinstancename*.*outportname constant signalname*

Example:

```
bind mod_A.out1 mod_B.in1 // Connect out1 of mod_A to in1 of mod_B
                             Then signal name is out1
bind mod_A.out2 mod_B.in2 // Connect out2(array) of mod_A to in2(array) of mod_B
                             The definition of the array form is unnecessary.
bind mod_A.out3 mod_B.in3 sig // Connect out3 of mod_A to in3 of mod_B
                                 Then signal name is sig.
bind 0 mod_B.in4 const_0 // Connect floating port const_0 to in4 of mod_B
                         // const_0 keeps on outputting the constant value 0
bind mod_A.out4 0 float_o4 // Connect out4 of mod_A to floating port float_o4
```

Note:

・Please do not specify different signal name having the same start point.

・Please do not specify same end point in multiple bind commands.

6.2.2 Format of hierarchy definition file

Please specify the above-mentioned commands for the hierarchy definition file by the following specified order and specified numbers. However, you can arrange the order arbitrary in the range (1) or (2) or (3). If you violate this rule, ssgen will detect an error.

```
    changelog log_info // unspecified OK, multi specified OK.
    style_module {sc|c++} // unspecified OK, or should be specified only 1 time.
    space_indent spacenum    // unspecified OK, or should be specified only 1 time.
    env_name file    // name is systemc, ssgen, vcs, ies, ctos, vcs_gcc, 1team, sschecker, slec.
(1)                 // unspecified OK, or should be specified as each only 1 time.
    mem_suffix suffixfile // unspecified OK, or should be specified only 1 time.
    style_alloc {static|alloc} // unspecified OK, or should be specified only 1 time.
    vcd_trace {on|off} // unspecified OK, or should be specified only 1 time.
    `include filename // unspecified OK, multi specified OK.
    `define macro // unspecified OK, multi specified OK.
    top name // should be specified only 1 time.
    sub filename instancename [filepath] // should be specified once or more time.
(2) tap [arguments] // unspecified OK, multi specified OK.
    bind [arguments] // unspecified OK, multi specified OK.
    `ifdef TESTBENCH    // unspecified OK.
    [const] type name [options] // type is {u|s}regN, {u|s}varN,
                                     // [u]char, [u]short, [u]in. multi specified OK.
                                     // can be used in only `ifdef TESTBENCH.
    func returntype name([argument_description]) // unspecified OK, multi specified OK.
(3)
                                                  // can be used in only `ifdef TESTBENCH.
    !--
    Free Area // unspecified OK, can be used in only `ifdef TESTBENCH.
    --!
    `endif    // unspecified OK, or should be specified certainly when specify `ifdef.
```

You can use **#ifdef/#endif** command for sub and bind if necessary.

7.Specifying macro to command parameters

You can specify macro to the following command parameters. Then, the macro is defined by **`define** command. You can not specify macro which is defined by **#define** command.

- Bit width N of **{u|s}inN**, **{u|s}outN**, **{u|s}regN, {u|s}varN** and **{u|s}evN**
- Element count of array variable specified by **{u|s}inN**, **{u|s}outN**, **{u|s}regN**, **{u|s}varN**, **[u]char**, **[u]short**, **[u]int** and **{u|s}evN**
- Initial value specified by **{u|s}outN**, **{u|s}regN**, **{u|s}varN**, **[u]char**, **[u]short**, **[u]int** and **{u|s}evN** (“-init” option)
- Maximum value/minimum value of range check specified by **{u|s}outN**, **{u|s}regN**, **{u|s}varN**, **[u]char**, **[u]short**, **[u]int** and **{u|s}evN** (“-max”/”-min” option)
- Width, size and latency of **{u|s}mem**
- Bit width N of **{u|s}varN** specified in return type and arugument type of **func**

ssgen supports specifying the following type of macro to command parameters

- Constant value

  `define BW0 8

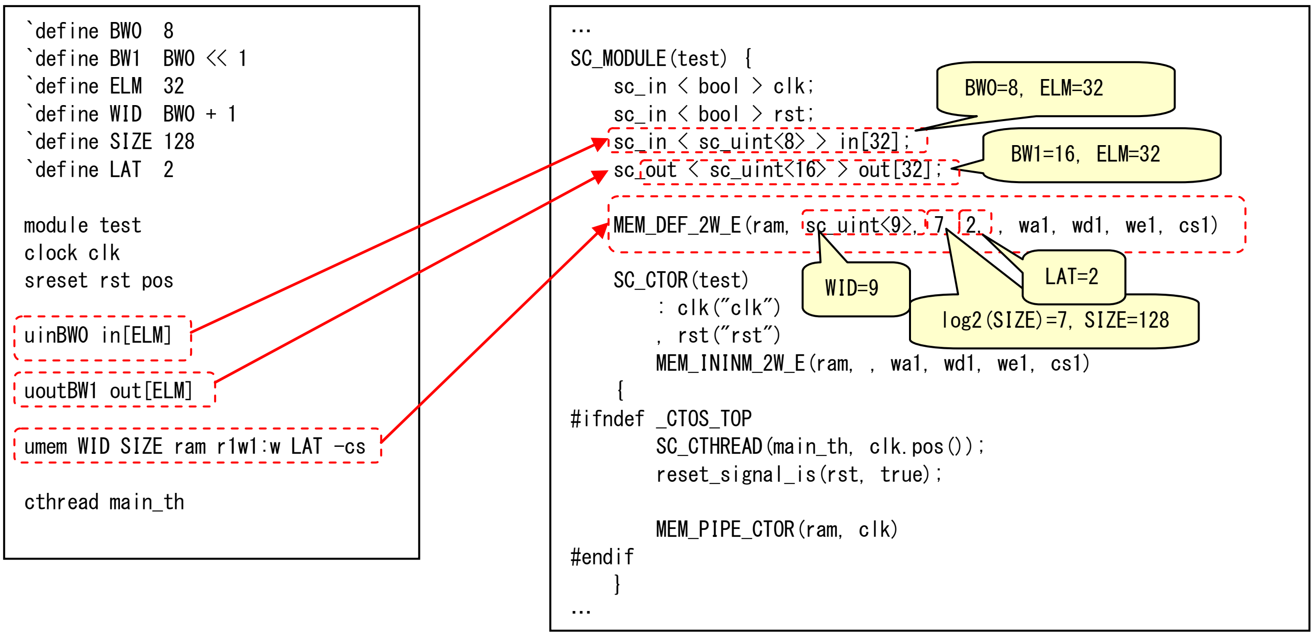
- Arithmetic operations (“+”, “-”, “*”, “/”) and shift operations (“<<”, “>>”)

  `define BW1 BW0 + 2

Here shows an example.

test.in

test.h



```
`define BW0  8
`define BW1  BW0 << 1
`define ELM  32
`define WID  BW0 + 1
`define SIZE 128
`define LAT  2

module test
clock clk
sreset rst pos

uinBW0 in[ELM]

uoutBW1 out[ELM]

umem WID SIZE ram r1w1:w LAT -cs

cthread main_th
```

```
…
SC_MODULE(test) {
    sc_in < bool > clk;
    sc_in < bool > rst;
    sc_in < sc_uint<8> > in[32];
    sc_out < sc_uint<16> > out[32];

    MEM_DEF_2W_E(ram, sc_uint<9>, 7, 2, wa1, wd1, we1, cs1)

    SC_CTOR(test)
        : clk("clk")
        , rst("rst")
        MEM_ININM_2W_E(ram, , wa1, wd1, we1, cs1)
    {
#ifndef _CTOS_TOP
        SC_CTHREAD(main_th, clk.pos());
        reset_signal_is(rst, true);

        MEM_PIPE_CTOR(ram, clk)
#endif
    }
…
```

8. Example of output file of ssgen

This chapter shows the example of the output files generated by module generation mode and hierarchy generation mode. The below table shows preprocessor macros which are used in output files of ssgen.

Macro	Usage
_DEBUG*	Use in SystemC simulation. The codes enclosed in this macro become effective when you specify this macro in SystemC compiling.
_MEM_MODEL	Use for switching the memory array access for the high-speed simulation and the memory port access for the high-level synthesis. The memory port access description for the high-level synthesis becomes effective when this macro is specified in SystemC compiling.
_OSCI	Use in SystemC simulation by OSCI SystemC. This macro is defined in Makefile generated by ssgen.
_MODE_RTL	Use in SystemC-RTL Co-simulation.
_CTOS_TOP	Use in high-level synthesis. Disable process registration of lower module in hierarchy module synthesis. (for reduction time and memory in synthesis)
__CTOS__	Use in high-level synthesis. This macro is defined by CtoS automatically.
CALYPTO_SYSC	Use in SystemC-RTL equivalence check (SLEC of Calypto). This macro is defined by SLEC automatically.
_SLEC_BBOX	Use in SystemC-RTL equivalence check with function blackboxing.
_COVERAGE	Use in SystemC code coverage.
SSGEN_ASSERT	Use in assertion check.

8.1 Example of output file of module generation mode

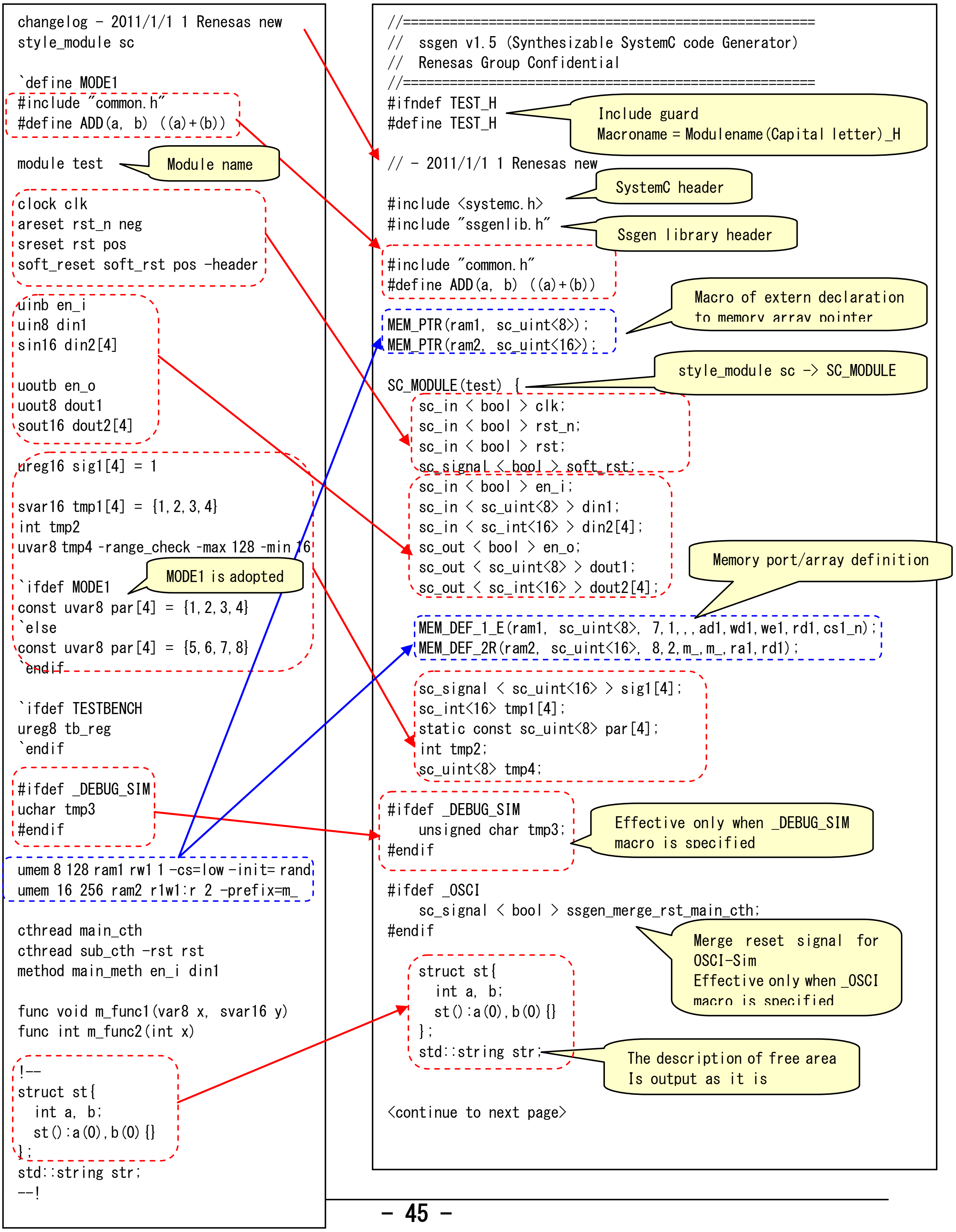
Here shows the SystemC description, the testbench, the memory model, and the CtoS script of module test generated from module definition file test.in. The execution of ssgen is

```
%s> ssgen.pl -in test.in -mem -osci -ctos
```

Moreover shows the list of the macro function for memory access that are generated when umem/smem command is specified.

8.1.1 SystemC description

Here shows SystemC description(test.h/test.cpp) generated from module definition file test.in.



test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init= rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst
method main_meth en_i din1

func void m_func1(var8 x, svar16 y)
func int m_func2(int x)

!--
struct st{
  int a, b;
  st():a(0),b(0) {}
};
std::string str;
--!
```

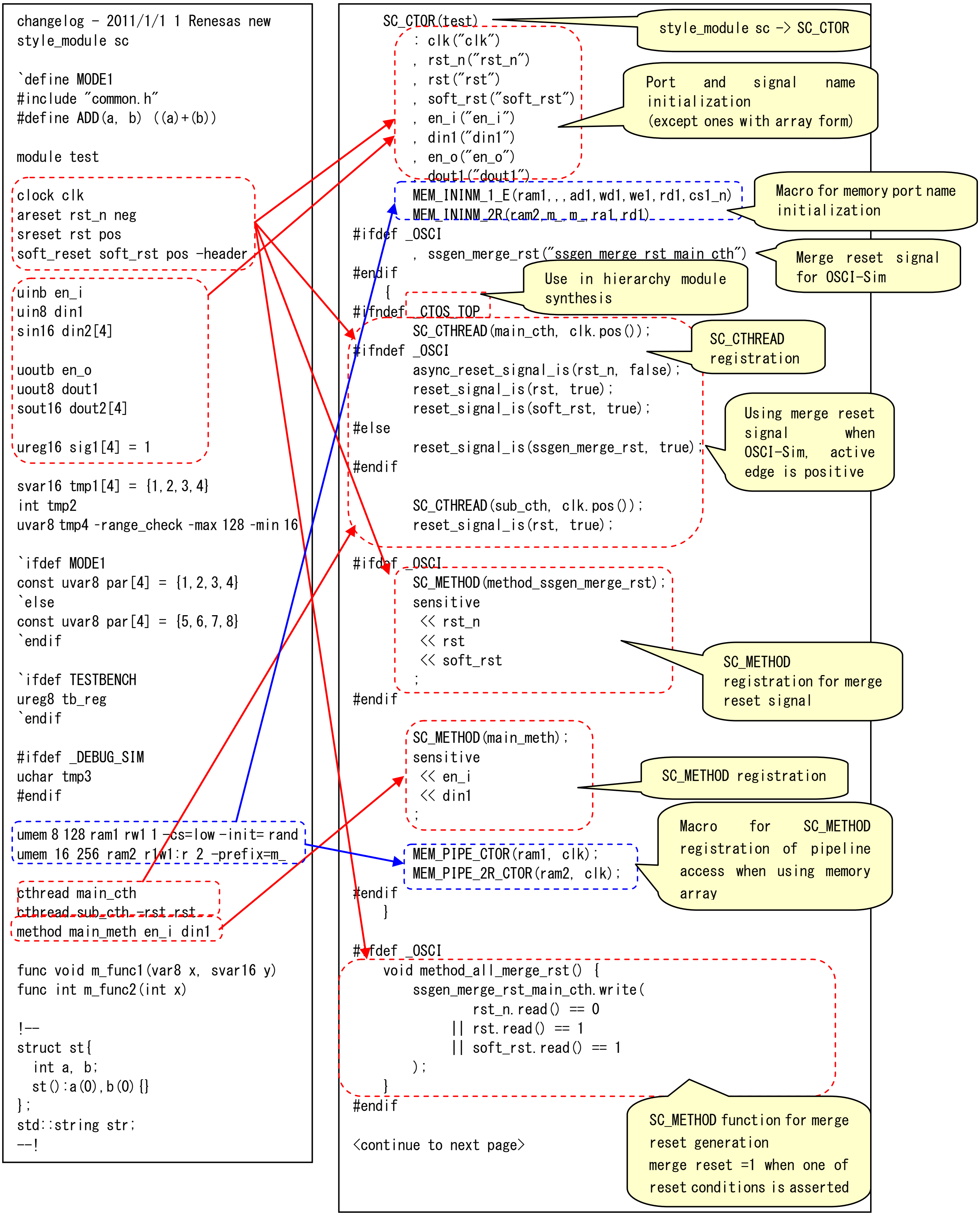
test.h (1/4)

```
//=================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=================================================
#ifndef TEST_H
#define TEST_H

// - 2011/1/1 1 Renesas new

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

#include "common.h"
#define ADD(a, b) ((a)+(b))

MEM_PTR(ram1, sc_uint<8>);
MEM_PTR(ram2, sc_uint<16>);

SC_MODULE(test) {
    sc_in < bool > clk;
    sc_in < bool > rst_n;
    sc_in < bool > rst;
    sc_signal < bool > soft_rst;
    sc_in < bool > en_i;
    sc_in < sc_uint<8> > din1;
    sc_in < sc_int<16> > din2[4];
    sc_out < bool > en_o;
    sc_out < sc_uint<8> > dout1;
    sc_out < sc_int<16> > dout2[4];

    MEM_DEF_1_E(ram1, sc_uint<8>, 7,1,,,ad1,wd1,we1,rd1,cs1_n);
    MEM_DEF_2R(ram2, sc_uint<16>, 8,2,m_,m_,ra1,rd1);

    sc_signal < sc_uint<16> > sig1[4];
    sc_int<16> tmp1[4];
    static const sc_uint<8> par[4];
    int tmp2;
    sc_uint<8> tmp4;

#ifdef _DEBUG_SIM
    unsigned char tmp3;
#endif

#ifdef _OSCI
    sc_signal < bool > ssgen_merge_rst_main_cth;
#endif

    struct st{
      int a, b;
      st():a(0),b(0) {}
    };
    std::string str;

<continue to next page>
```

Module name
MODE1 is adopted
Include guard
Macroname = Modulename(Capital letter)_H
SystemC header
Ssgen library header
Macro of extern declaration
to memory array pointer
style_module sc -> SC_MODULE
Memory port/array definition
Effective only when _DEBUG_SIM
macro is specified
Merge reset signal for
OSCI-Sim
Effective only when _OSCI
macro is specified
The description of free area
Is output as it is

test.in

test.h (2/4)



```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init= rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst
method main_meth en_i din1

func void m_func1(var8 x, svar16 y)
func int m_func2(int x)

!--
struct st{
  int a, b;
  st():a(0),b(0) {}
};
std::string str;
--!
```

```
    SC_CTOR(test)
        : clk("clk")
        , rst_n("rst_n")
        , rst("rst")
        , soft_rst("soft_rst")
        , en_i("en_i")
        , din1("din1")
        , en_o("en_o")
        , dout1("dout1")
        MEM_ININM_1_E(ram1,,,ad1,wd1,we1,rd1,cs1_n)
        MEM_ININM_2R(ram2,m_,m_,ra1,rd1)
#ifdef _OSCI
        , ssgen_merge_rst("ssgen_merge_rst_main_cth")
#endif
     {
#ifndef _CTOS_TOP
        SC_CTHREAD(main_cth, clk.pos());
#ifndef _OSCI
        async_reset_signal_is(rst_n, false);
        reset_signal_is(rst, true);
        reset_signal_is(soft_rst, true);
#else
        reset_signal_is(ssgen_merge_rst, true);
#endif

        SC_CTHREAD(sub_cth, clk.pos());
        reset_signal_is(rst, true);

#ifdef _OSCI
        SC_METHOD(method_ssgen_merge_rst);
        sensitive
         << rst_n
         << rst
         << soft_rst
        ;
#endif

        SC_METHOD(main_meth);
        sensitive
         << en_i
         << din1
        ;

        MEM_PIPE_CTOR(ram1, clk);
        MEM_PIPE_2R_CTOR(ram2, clk);
#endif
     }

#ifdef _OSCI
    void method_all_merge_rst() {
        ssgen_merge_rst_main_cth.write(
                rst_n.read() == 0
             || rst.read() == 1
             || soft_rst.read() == 1
        );
    }
#endif

<continue to next page>
```

style_module sc -> SC_CTOR

Port and signal name initialization (except ones with array form)

Macro for memory port name initialization

Merge reset signal for OSCI-Sim

Use in hierarchy module synthesis

SC_CTHREAD registration

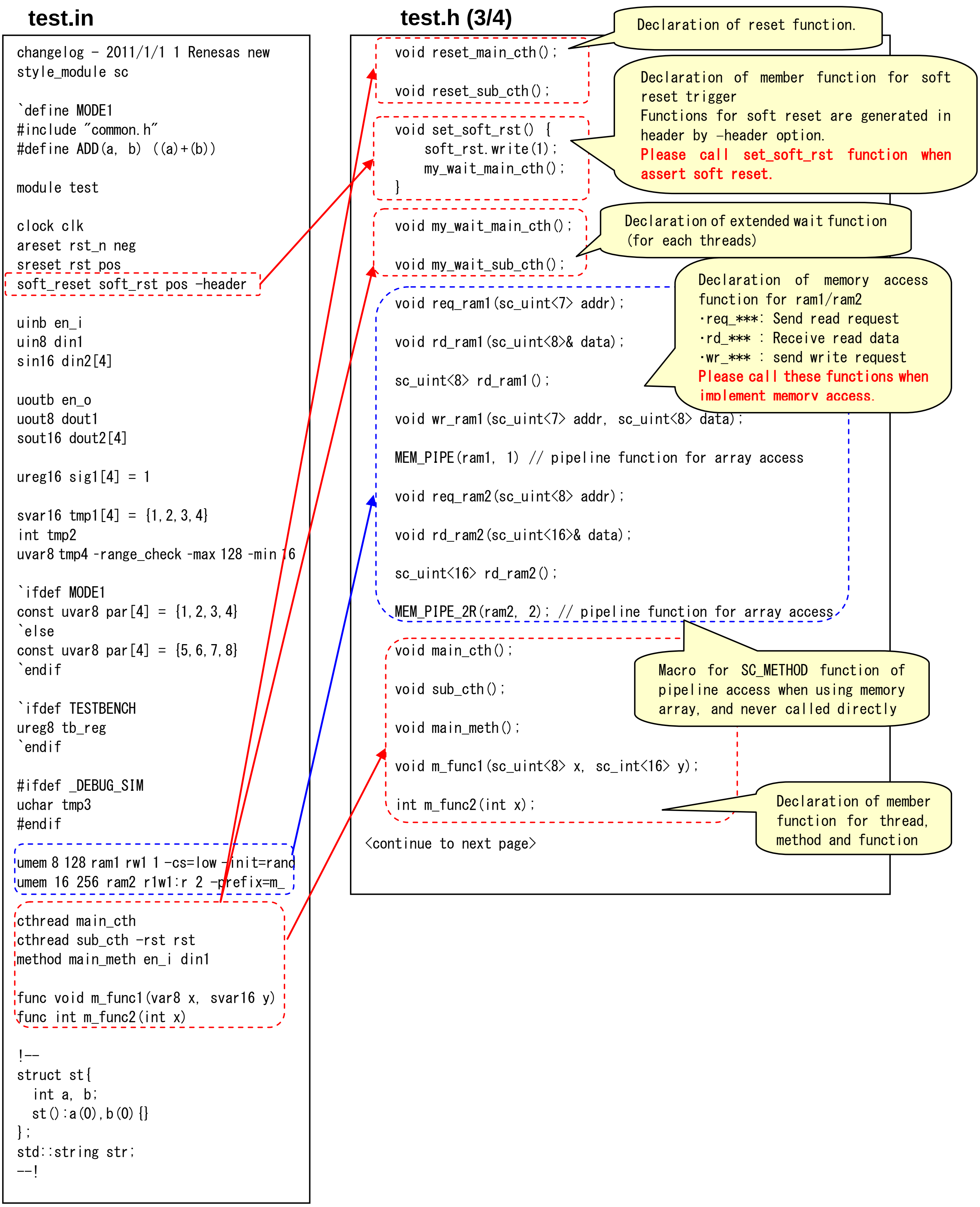
Using merge reset signal when OSCI-Sim, active edge is positive

SC_METHOD registration for merge reset signal

SC_METHOD registration

Macro for SC_METHOD registration of pipeline access when using memory array

SC_METHOD function for merge reset generation merge reset =1 when one of reset conditions is asserted



test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst
method main_meth en_i din1

func void m_func1(var8 x, svar16 y)
func int m_func2(int x)

!--
struct st{
  int a, b;
  st():a(0),b(0){}
};
std::string str;
--!
```

test.h (3/4)

```
void reset_main_cth();

void reset_sub_cth();

void set_soft_rst() {
    soft_rst.write(1);
    my_wait_main_cth();
}

void my_wait_main_cth();

void my_wait_sub_cth();

void req_ram1(sc_uint<7> addr);

void rd_ram1(sc_uint<8>& data);

sc_uint<8> rd_ram1();

void wr_ram1(sc_uint<7> addr, sc_uint<8> data);

MEM_PIPE(ram1, 1) // pipeline function for array access

void req_ram2(sc_uint<8> addr);

void rd_ram2(sc_uint<16>& data);

sc_uint<16> rd_ram2();

MEM_PIPE_2R(ram2, 2); // pipeline function for array access

void main_cth();

void sub_cth();

void main_meth();

void m_func1(sc_uint<8> x, sc_int<16> y);

int m_func2(int x);

<continue to next page>
```

Declaration of reset function.

Declaration of member function for soft reset trigger
Functions for soft reset are generated in header by -header option.
Please call set_soft_rst function when assert soft reset.

Declaration of extended wait function
(for each threads)

Declaration of memory access function for ram1/ram2
・req_***: Send read request
・rd_*** : Receive read data
・wr_*** : send write request
Please call these functions when implement memory access.

Macro for SC_METHOD function of pipeline access when using memory array, and never called directly

Declaration of member function for thread, method and function

test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst
method main_meth en_i din1

func void m_func1(var8 x, svar16 y)
func int m_func2(int x)

!--
struct st{
  int a, b;
  st():a(0),b(0){}
};
std::string str;
--!
```

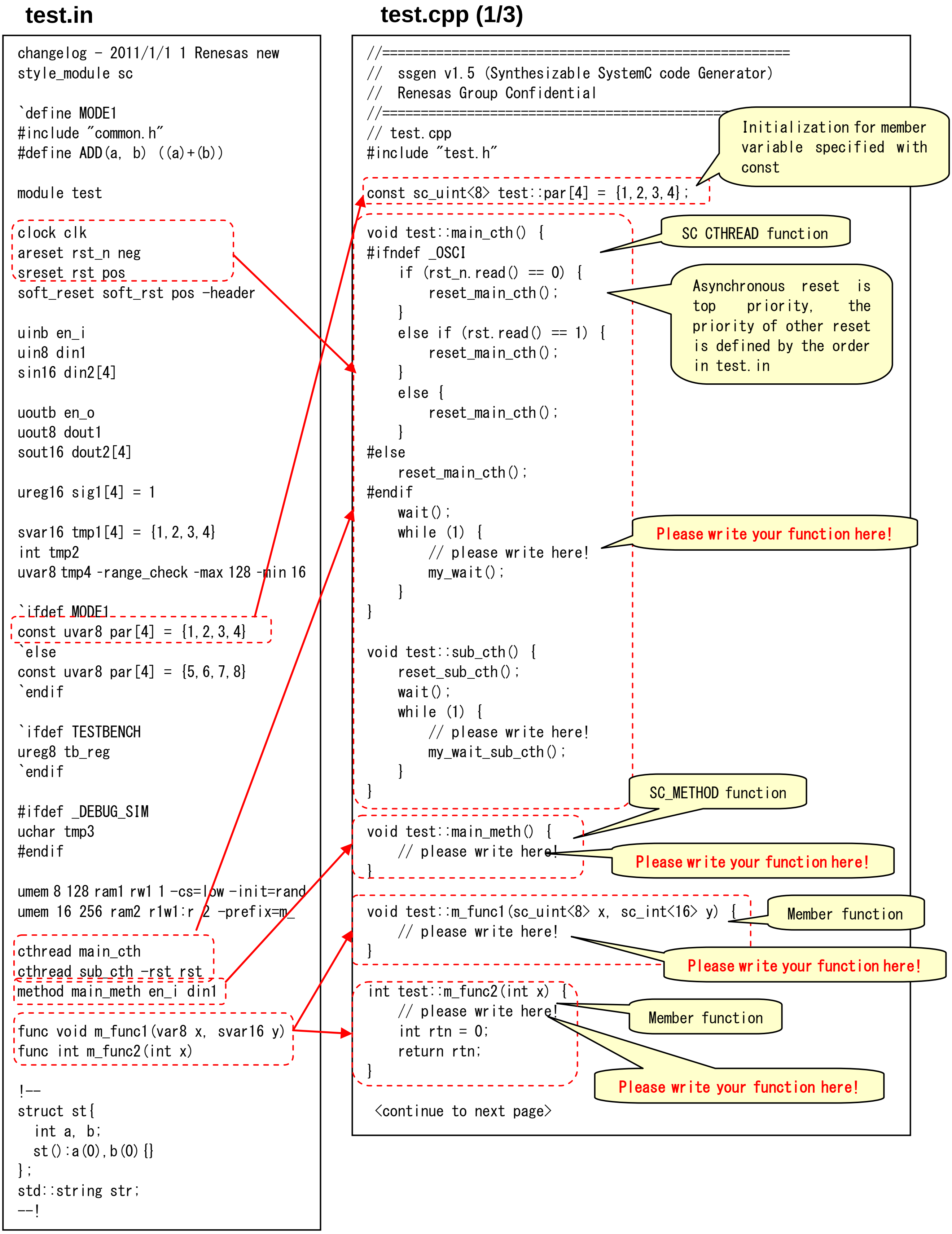
test.h (4/E)

```
#if !defined(__CTOS__) && !defined(CALYPTO_SYSC)
    void vcd_trace(ssgen_trace_file* tf, int depth = HIER_MAX) {
        m_tf = tf;
        if (tf != 0 && depth > 0) {
            std::string nm = std::string(name());
            sc_trace(tf, clk, nm + ".clk");
            sc_trace(tf, rst_n, nm + ".rst_n");
            sc_trace(tf, rst, nm + ".rst");
            sc_trace(tf, soft_rst, nm + ".soft_rst");
            sc_trace(tf, en_i, nm + ".en_i");
            sc_trace(tf, din1, nm + ".din1");
            for (int i0 = 0; i0 < 4; i0++) {
                std::ostringstream indx;
                indx << "(" << i0 << ")";
                sc_trace(tf, din2[i0], nm + ".din2" + indx.str());
            }
            sc_trace(tf, en_o, nm + ".en_o");
            sc_trace(tf, dout1, nm + ".dout1");
            for (int i0 = 0; i0 < 4; i0++) {
                std::ostringstream indx;
                indx << "(" << i0 << ")";
                sc_trace(tf, dout2[i0], nm + ".dout2" + indx.str())
            }
            for (int i0 = 0; i0 < 4; i0++) {
                std::ostringstream indx;
                indx << "(" << i0 << ")";
                sc_trace(tf, sig1[i0], nm + ".sig1" + indx.str());
            }
#ifdef _MEM_MODEL
            sc_trace(tf, ram1_ad1, nm + ".ram1_ad1");
            sc_trace(tf, ram1_wd1, nm + ".ram1_wd1");
            sc_trace(tf, ram1_we1, nm + ".ram1_we1");
            sc_trace(tf, ram1_rd1, nm + ".ram1_rd1");
            sc_trace(tf, ram1_cs1_n, nm + ".ram1_cs1_n");
#endif
#ifdef _MEM_MODEL
            sc_trace(tf, m_ram2_ra1, nm + ".m_ram2_ra1");
            sc_trace(tf, m_ram2_rd1, nm + ".m_ram2_rd1");
#endif
        }
    }
#endif // __CTOS__
};

#ifdef __CTOS__
SC_MODULE_EXPORT(test);
#endif

#endif // TEST_H
```

Description of wave dump for port and signal
For array, all elements are dumped by "for" sentence

Description of wave dump for memory port ram2 is specified with prefix "m_"

Necessary description for CtoS execution



test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst
method main_meth en_i din1

func void m_func1(var8 x, svar16 y)
func int m_func2(int x)

!--
struct st{
  int a, b;
  st():a(0),b(0){}
};
std::string str;
--!
```

test.cpp (1/3)

```
//==================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//==================================================
// test.cpp
#include "test.h"

const sc_uint<8> test::par[4] = {1,2,3,4};

void test::main_cth() {
#ifndef _OSCI
    if (rst_n.read() == 0) {
        reset_main_cth();
    }
    else if (rst.read() == 1) {
        reset_main_cth();
    }
    else {
        reset_main_cth();
    }
#else
    reset_main_cth();
#endif
    wait();
    while (1) {
        // please write here!
        my_wait();
    }
}

void test::sub_cth() {
    reset_sub_cth();
    wait();
    while (1) {
        // please write here!
        my_wait_sub_cth();
    }
}

void test::main_meth() {
    // please write here!
}

void test::m_func1(sc_uint<8> x, sc_int<16> y) {
    // please write here!
}

int test::m_func2(int x) {
    // please write here!
    int rtn = 0;
    return rtn;
}

 <continue to next page>
```

Initialization for member variable specified with const

SC CTHREAD function

Asynchronous reset is top priority, the priority of other reset is defined by the order in test.in

Please write your function here!

SC_METHOD function

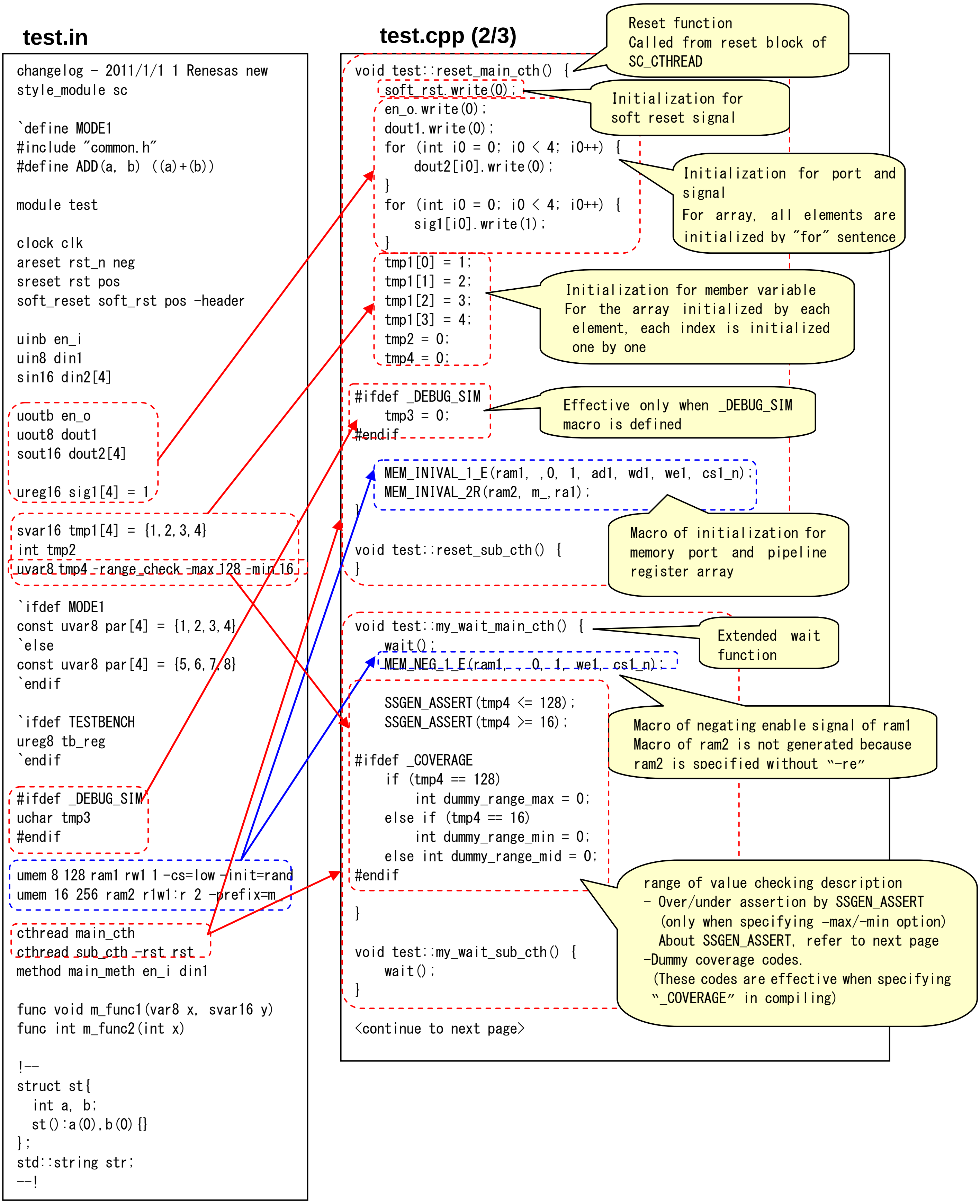
Please write your function here!

Member function

Please write your function here!

Member function

Please write your function here!



test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m

cthread main_cth
cthread sub_cth -rst rst
method main_meth en_i din1

func void m_func1(var8 x, svar16 y)
func int m_func2(int x)

!--
struct st{
  int a, b;
  st():a(0),b(0){}
};
std::string str;
--!
```

test.cpp (2/3)

```
void test::reset_main_cth() {
    soft_rst.write(0);
    en_o.write(0);
    dout1.write(0);
    for (int i0 = 0; i0 < 4; i0++) {
        dout2[i0].write(0);
    }
    for (int i0 = 0; i0 < 4; i0++) {
        sig1[i0].write(1);
    }
    tmp1[0] = 1;
    tmp1[1] = 2;
    tmp1[2] = 3;
    tmp1[3] = 4;
    tmp2 = 0;
    tmp4 = 0;
#ifdef _DEBUG_SIM
    tmp3 = 0;
#endif

    MEM_INIVAL_1_E(ram1, ,0, 1, ad1, wd1, we1, cs1_n);
    MEM_INIVAL_2R(ram2, m_,ra1);
}

void test::reset_sub_cth() {
}

void test::my_wait_main_cth() {
    wait();
    MEM_NEG_1_E(ram1, , 0, 1, we1, cs1_n);

    SSGEN_ASSERT(tmp4 <= 128);
    SSGEN_ASSERT(tmp4 >= 16);

#ifdef _COVERAGE
    if (tmp4 == 128)
        int dummy_range_max = 0;
    else if (tmp4 == 16)
        int dummy_range_min = 0;
    else int dummy_range_mid = 0;
#endif

}

void test::my_wait_sub_cth() {
    wait();
}

<continue to next page>
```

Reset function
Called from reset block of
SC_CTHREAD

Initialization for
soft reset signal

Initialization for port and
signal
For array, all elements are
initialized by "for" sentence

Initialization for member variable
For the array initialized by each
element, each index is initialized
one by one

Effective only when _DEBUG_SIM
macro is defined

Macro of initialization for
memory port and pipeline
register array

Extended wait
function

Macro of negating enable signal of ram1
Macro of ram2 is not generated because
ram2 is specified without "-re"

range of value checking description
- Over/under assertion by SSGEN_ASSERT
  (only when specifying -max/-min option)
  About SSGEN_ASSERT, refer to next page
-Dummy coverage codes.
  (These codes are effective when specifying
  "_COVERAGE" in compiling)

test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst
method main_meth en_i din1

func void m_func1(var8 x, svar16 y)
func int m_func2(int x)

!--
struct st{
  int a, b;
  st():a(0),b(0){}
};
std::string str;
--!
```

test.cpp (3/E)

```
void test::req_ram1(sc_uint<7> addr) {
    MEM_REQ_1_E(ram1, , 0, ad1, cs1_n);
}

void test::rd_ram1(sc_uint<8>& data) {
    MEM_RD_1(ram1, 1, , rd1);
}

sc_uint<8> test::rd_ram1() {
    sc_uint<8> data;
    MEM_RD_1(ram1, 1, , rd1);
}

void test::wr_ram1(sc_uint<7> addr, sc_uint<8> data) {
    MEM_WR_1_E(ram1, , 1, 0, ad1, wd1, we1, cs1_n);
 }

void test::req_ram2(sc_uint<8> addr) {
    MEM_REQ_2(ram2, m_, ra1);
}

void test::rd_ram2(sc_uint<16>& data) {
    MEM_RD_2(ram2, 2, m_, rd1);
}

sc_uint<16> test::rd_ram2() {
    sc_uint<16> data;
    MEM_RD_2(ram2, 2, m_, rd1);
    return data;
}
```

memory access function for ram1/ram2
・req_***: Send read request
・rd_*** : Receive read data
・wr_*** : send write request

Assertion macro SSGEN_ASSERT

・Extended macro of "assert"
・Definition of SSGEN_ASSERT is in ssgenlib.h
・You can switch force-quit of simulation by specifying macro "_DEBUG_SIM" in compiling
・You can use this macro for assertion in your code.

```
#if !defined(__CTOS__) && !defined(CALYPTO_SYSC)
#ifdef _DEBUG_SIM
#define SSGEN_ASSERT(a) ¥
  if(!(a)) SC_REPORT_WARNING( "assert check", #a );
#else
#define SSGEN_ASSERT(a) assert((a));
#endif

#else
#define SSGEN_ASSERT(a)
#endif
```

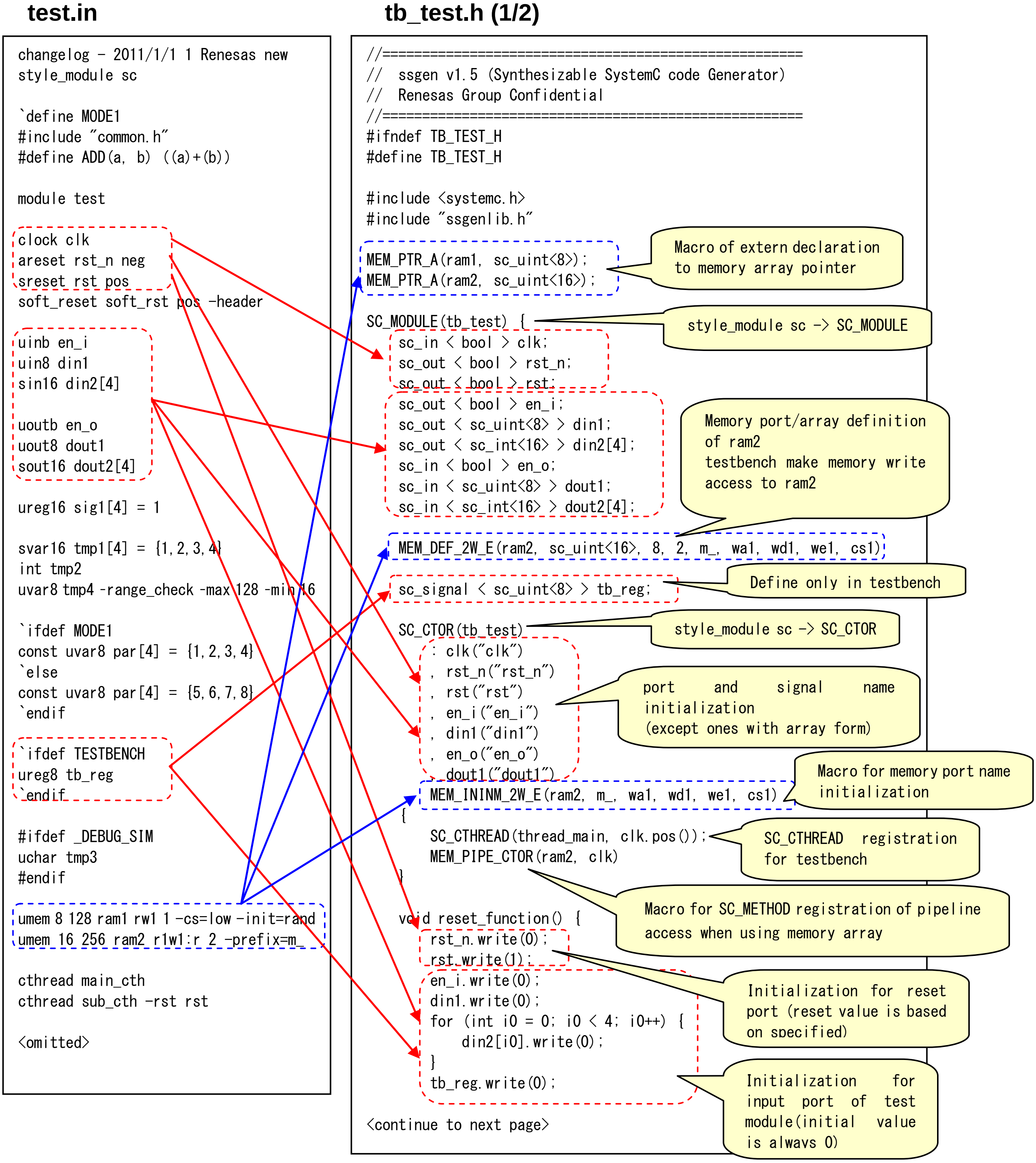
When specifying "_DEBUG_SIM" in compiling, only output message and not force-quit simulation.

When not specifying "_DEBUG_SIM", force-quit simulation by assert.

Nothing when except simulation

8.1.2 testbench description

Here shows testbench description(tb_test.h/tb_test.cpp/main_test.cpp) generated from module definition file test.in. (the description of test.in is omitted partially)



test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst

<omitted>
```

tb_test.h (1/2)

```
//=====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=====================================================
#ifndef TB_TEST_H
#define TB_TEST_H

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

MEM_PTR_A(ram1, sc_uint<8>);
MEM_PTR_A(ram2, sc_uint<16>);

SC_MODULE(tb_test) {
    sc_in < bool > clk;
    sc_out < bool > rst_n;
    sc_out < bool > rst;
    sc_out < bool > en_i;
    sc_out < sc_uint<8> > din1;
    sc_out < sc_int<16> > din2[4];
    sc_in < bool > en_o;
    sc_in < sc_uint<8> > dout1;
    sc_in < sc_int<16> > dout2[4];

    MEM_DEF_2W_E(ram2, sc_uint<16>, 8, 2, m_, wa1, wd1, we1, cs1)

    sc_signal < sc_uint<8> > tb_reg;

    SC_CTOR(tb_test)
        : clk("clk")
        , rst_n("rst_n")
        , rst("rst")
        , en_i("en_i")
        , din1("din1")
        , en_o("en_o")
        , dout1("dout1")
        MEM_ININM_2W_E(ram2, m_, wa1, wd1, we1, cs1)
    {
        SC_CTHREAD(thread_main, clk.pos());
        MEM_PIPE_CTOR(ram2, clk)
    }

    void reset_function() {
        rst_n.write(0);
        rst.write(1);
        en_i.write(0);
        din1.write(0);
        for (int i0 = 0; i0 < 4; i0++) {
            din2[i0].write(0);
        }
        tb_reg.write(0);

<continue to next page>
```

Macro of extern declaration to memory array pointer

style_module sc -> SC_MODULE

Memory port/array definition of ram2
testbench make memory write access to ram2

Define only in testbench

style_module sc -> SC_CTOR

port and signal name initialization
(except ones with array form)

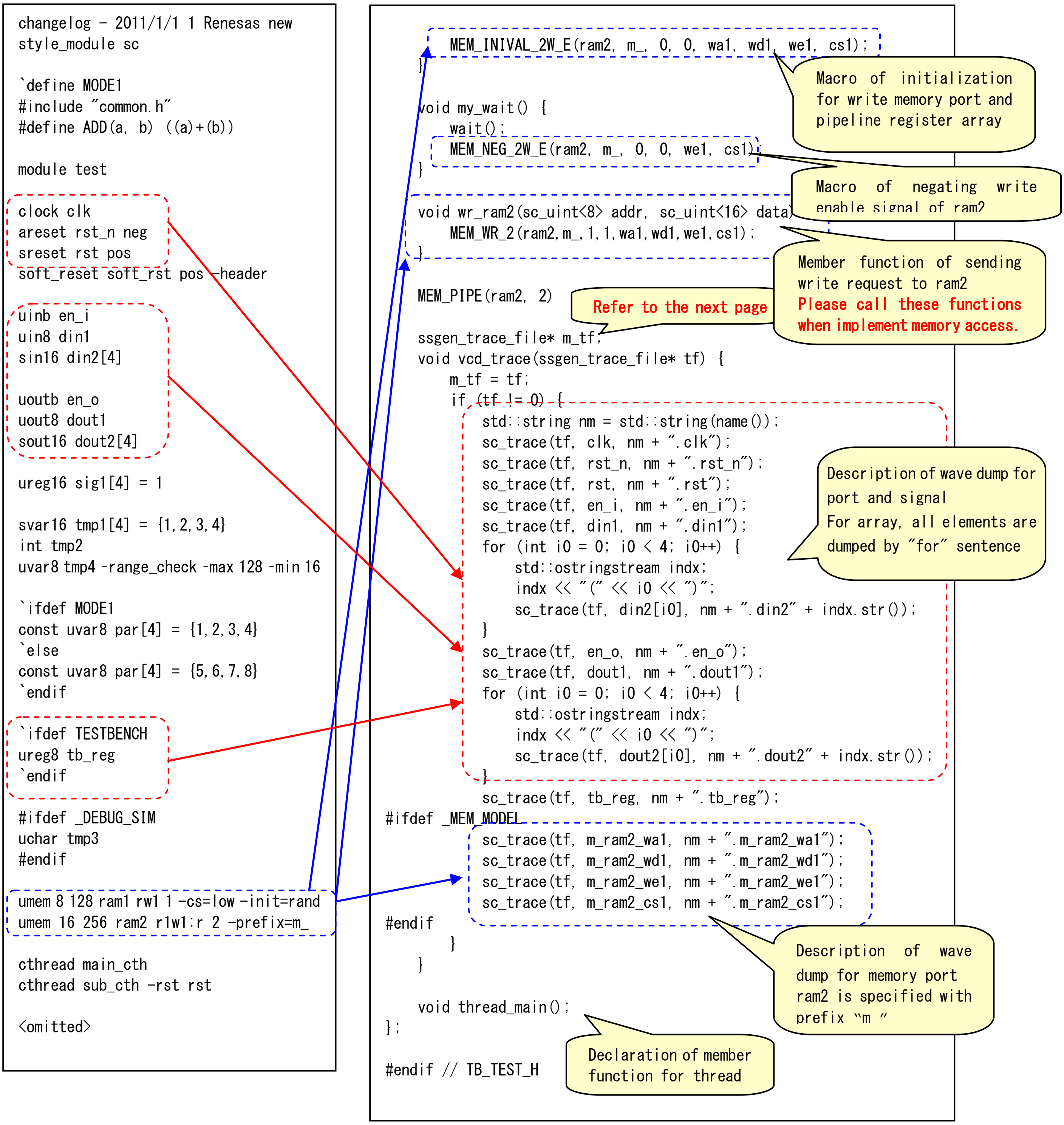
Macro for memory port name initialization

SC_CTHREAD registration for testbench

Macro for SC_METHOD registration of pipeline access when using memory array

Initialization for reset port (reset value is based on specified)

Initialization for input port of test module(initial value is always 0)



test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst

<omitted>
```

tb_test.h (2/E)

```
    MEM_INIVAL_2W_E(ram2, m_, 0, 0, wa1, wd1, we1, cs1);
  }

  void my_wait() {
    wait();
    MEM_NEG_2W_E(ram2, m_, 0, 0, we1, cs1);
  }

  void wr_ram2(sc_uint<8> addr, sc_uint<16> data)
    MEM_WR_2(ram2,m_,1,1,wa1,wd1,we1,cs1);
  }

  MEM_PIPE(ram2, 2)

  ssgen_trace_file* m_tf;
  void vcd_trace(ssgen_trace_file* tf) {
    m_tf = tf;
    if (tf != 0) {
      std::string nm = std::string(name());
      sc_trace(tf, clk, nm + ".clk");
      sc_trace(tf, rst_n, nm + ".rst_n");
      sc_trace(tf, rst, nm + ".rst");
      sc_trace(tf, en_i, nm + ".en_i");
      sc_trace(tf, din1, nm + ".din1");
      for (int i0 = 0; i0 < 4; i0++) {
        std::ostringstream indx;
        indx << "(" << i0 << ")";
        sc_trace(tf, din2[i0], nm + ".din2" + indx.str());
      }
      sc_trace(tf, en_o, nm + ".en_o");
      sc_trace(tf, dout1, nm + ".dout1");
      for (int i0 = 0; i0 < 4; i0++) {
        std::ostringstream indx;
        indx << "(" << i0 << ")";
        sc_trace(tf, dout2[i0], nm + ".dout2" + indx.str());
      }
      sc_trace(tf, tb_reg, nm + ".tb_reg");
#ifdef _MEM_MODEL
      sc_trace(tf, m_ram2_wa1, nm + ".m_ram2_wa1");
      sc_trace(tf, m_ram2_wd1, nm + ".m_ram2_wd1");
      sc_trace(tf, m_ram2_we1, nm + ".m_ram2_we1");
      sc_trace(tf, m_ram2_cs1, nm + ".m_ram2_cs1");
#endif
    }
  }

  void thread_main();
};

#endif // TB_TEST_H
```

Macro of initialization for write memory port and pipeline register array

Macro of negating write enable signal of ram2

Member function of sending write request to ram2
Please call these functions when implement memory access.

Refer to the next page

Description of wave dump for port and signal
For array, all elements are dumped by "for" sentence

Description of wave dump for memory port ram2 is specified with prefix "m "

Declaration of member function for thread

test.in

```
changelog - 2011/1/1 1 Renesas new
style_module sc

`define MODE1
#include "common.h"
#define ADD(a, b) ((a)+(b))

module test

clock clk
areset rst_n neg
sreset rst pos
soft_reset soft_rst pos -header

uinb en_i
uin8 din1
sin16 din2[4]

uoutb en_o
uout8 dout1
sout16 dout2[4]

ureg16 sig1[4] = 1

svar16 tmp1[4] = {1,2,3,4}
int tmp2
uvar8 tmp4 -range_check -max 128 -min 16

`ifdef MODE1
const uvar8 par[4] = {1,2,3,4}
`else
const uvar8 par[4] = {5,6,7,8}
`endif

`ifdef TESTBENCH
ureg8 tb_reg
`endif

#ifdef _DEBUG_SIM
uchar tmp3
#endif

umem 8 128 ram1 rw1 1 -cs=low -init=rand
umem 16 256 ram2 r1w1:r 2 -prefix=m_

cthread main_cth
cthread sub_cth -rst rst

<omitted>
```

tb_test.cpp

```
//=====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=====================================================
// tb_test.cpp
#include "tb_test.h"

void tb_test::thread_main() {
    reset_function();
    wait();
    // please write here!

    my_wait();
    sc_stop();
    my_wait();
}
```

SC_CTHREAD function

Please write your function here!

Extended class of VCD trace **ssgen_trace_file**

・This class inherits the vcd_trace_file class of SystemC library.
・m_tf (ssgen_trace_file* m_tf;) is only declared in testbench.
・You can control VCP file dump by using member functions on()/off() of this class.

Example.

```
void tb_test::thread_main() { // SC_CTHREAD of testbench
    reset_function();
    wait();
    // please write here!
    for(int i=0;i<100+10;i++){
      …
      if(m_tf != NULL){ // Please describe this code if you use on()/off().
        if(i < 45){
          m_tf->on(); // VCD file dump ON
        }else if(i < 80){
          m_tf->off(); // VCD file dump OFF
        }else{
          m_tf->on(); // VCD file dump ON
        }
      }
      …
      my_wait();
    }
    sc_stop();
    my_wait();
}
```

main_test.cpp (1/4)

```
//=======================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=======================================================
// main_test.cpp
#include "test.h"
#include "tb_test.h"
#include "mem_rw1.h"
#include "mem_r1w1.h"

sc_uint<8> *ptr_ram1;
sc_uint<16> *ptr_ram2;

int sc_main(int argc, char *argv[]) {
    bool vcd_dump = 0;
    int  vcd_depth = HIER_MAX;
    for (int i = 1; i < argc; i++) {
        // vcd dump control
        if (strcmp(argv[i], "-vcd") == 0) {
            vcd_dump = 1;
            if (i < argc - 1 && *argv[i+1] != '-') {
                vcd_depth = atoi(argv[i+1]);
                i++;
            }
        }
        // command line sample (you can customize)
        else if (strcmp(argv[i], "-mode") == 0) {
            i++;
            if (i < argc) {
                if (strcmp(argv[i], "0") == 0) {
                }
                else {
                }
            }
        }
    }

    sc_set_time_resolution(1, SC_PS);
    sc_clock clk("clk", 10, SC_NS);

    ptr_ram1 = new sc_uint<8> [128];
    ptr_ram2 = new sc_uint<16> [256];

    test test0("test0");
    tb_test tb_test0("tb_test0");

#ifdef _MEM_MODEL
    mem_rw1<sc_uint<7>, sc_uint<8>, 1, 1, 0, 128> ram1("ram1", ptr_ram1, -1);
#endif

#ifdef _MEM_MODEL
    mem_r1w1<sc_uint<8>, sc_uint<16>, 2, 1, 1, 1, 256> ram2("ram2", ptr_ram2, 0);
#endif
```

<continue to next page>

Include header file
·test.h : header file of test module
·tb_test.h : header file of testbench

Include memory model
・mem_rw1.h : single port memory model
・mem_r1w1.h : 2port memory model

Declaration of memory array pointer

ON/OFF switching of VCD dump
To the command line argument of simulation
・"- vcd" is specified, VCD dump is ON for all hierarchies
・"- vcd 2" is specified, VCD dump is ON for 2 level hierarchies

Please customize another command line argument ("-mode" is reference)

Clock generation

Memory array allocation

Instantiation of test module and testbench

Instantiation of memory model
(only when _ MEM_MODEL macro is specified)
mem_rw1 : single port memory
mem_r1w1 : 2 port memory
memory model template argument is:
<bit width of address, bit width of data,
latency, we level, cs level,
re level (only when mem_r1w1),
number of words>

main_test.cpp (2/4)

```
    sc_signal < bool > rst_n;
    sc_signal < bool > rst;
    sc_signal < bool > en_i;
    sc_signal < sc_uint<8> > din1;
    sc_signal < sc_int<16> > din2[4];
    sc_signal < bool > en_o;
    sc_signal < sc_uint<8> > dout1;
    sc_signal < sc_int<16> > dout2[4];
```

Signals between ports of test module and testbench

```
#ifdef _MEM_MODEL
    sc_signal < sc_uint<7> > ram1_ad1;
    sc_signal < sc_uint<8> > ram1_wd1;
    sc_signal < sc_uint<8> > ram1_rd1;
    sc_signal < bool > ram1_we1;
    sc_signal < bool >ram1_cs1_n;
#endif
```

Signals between ports of test module and single port memory model

```
#ifdef _MEM_MODEL
    sc_signal < sc_uint<8> > m_ram2_wa1;
    sc_signal < sc_uint<16> > m_ram2_wd1;
    sc_signal < bool > m_ram2_we1;
    sc_signal < bool > m_ram2_cs1;
    sc_signal < sc_uint<8> > m_ram2_ra1;
    sc_signal < sc_uint<16> > m_ram2_rd1;
    sc_signal < bool >m_ram2_re1;
#endif
```

Signals between ports of test module and test bench and 2port memory model

```
    test0.clk(clk);
    test0.rst_n(rst_n);
    test0.rst(rst);
    test0.en_i(en_i);
    test0.din1(din1);
#ifndef _MODE_RTL
    for (int i0 = 0; i0 < 4; i0++) {
        test0.din2[i0](din2[i0]);
    }
#else
    test0.din2_0(din2[0]);
    test0.din2_1(din2[1]);
    test0.din2_2(din2[2]);
    test0.din2_3(din2[3]);
#endif
    test0.en_o(en_o);
    test0.dout1(dout1);
#ifndef _MODE_RTL
    for (int i0 = 0; i0 < 4; i0++) {
        test0.dout2[i0](dout2[i0]);
    }
#else
    test0.dout2_0(dout2[0]);
    test0.dout2_1(dout2[1]);
    test0.dout2_2(dout2[2]);
    test0.dout2_3(dout2[3]);
#endif
```

Signal connections with test module

When connect to RTL by CoSim (_MODE_MACRO is effective), array port must be separated to scalar port

<continue to next page>

main_test.cpp (3/4)

```
#ifdef _MEM_MODEL
    test0.ram1_ad1(ram1_ad1);
    test0.ram1_wd1(ram1_wd1);
    test0.ram1_rd1(ram1_rd1);
    test0.ram1_we1(ram1_we1);
    test0.ram1_cs1(ram1_cs1_n);
#endif
```

Signal connection with test module for single port memory

```
#ifdef _MEM_MODEL
    test0.m_ram2_ra1(m_ram2_ra1);
    test0.m_ram2_rd1(m_ram2_rd1);
#endif
```

Signal connection with test module for 2 port memory read

```
    tb_test0.clk(clk);
    tb_test0.rst_n(rst_n);
    tb_test0.rst(rst);
    tb_test0.en_i(en_i);
    tb_test0.din1(din1);
    for (int i0 = 0; i0 < 4; i0++) {
        tb_test0.din2[i0](din2[i0]);
    }    tb_test0.en_o(en_o);
    tb_test0.dout1(dout1);
    for (int i0 = 0; i0 < 4; i0++) {
        tb_test0.dout2[i0](dout2[i0]);
    }
```

Signal connections with testbench

```
#ifdef _MEM_MODEL
    tb_test0.m_ram2_wa1(m_ram2_wa1);
    tb_test0.m_ram2_wd1(m_ram2_wd1);
    tb_test0.m_ram2_we1(m_ram2_we1);
    tb_test0.m_ram2_cs1(m_ram2_cs1);

#endif
```

Signal connections with testbench for 2 port memory write

```
#ifdef _MEM_MODEL
    ram1.clk(clk);
    ram1.ad1(ram1_ad1);
    ram1.wd1(ram1_wd1);
    ram1.we1(ram1_we1);
    ram1.rd1(ram1_rd1);
    ram1.cs1(ram1_cs1_n);
#endif
```

Signal connections with single port memory

```
#ifdef _MEM_MODEL
    ram2.clk(clk);
    ram2.wa1(m_ram2_wa1);
    ram2.wd1(m_ram2_wd1);
    ram2.we1(m_ram2_we1);
    ram2.ra1(m_ram2_ra1);
    ram2.rd1(m_ram2_rd1);
    ram2.re1(m_ram2_re1);
    m_ram2_re1.write(1);
#endif
```

Signal connections with 2port memory

Read enable signal of ram2 is always 1 because ram2 does not have read enable

<continue to next page>

main_test.cpp (4/E)

```
    ssgen_trace_file *tf = NULL;
    if (vcd_dump == 1) tf = create_ssgen_trace_file("test");
#ifndef _MODE_RTL
    test0.vcd_trace(tf, vcd_depth);
#endif
    tb_test0.vcd_trace(tf);
#ifdef _MEM_MODEL
    ram1.vcd_trace(tf);
#endif
#ifdef _MEM_MODEL
    ram2.vcd_trace(tf);
#endif

    sc_start();

    if (vcd_dump == 1) {
        sc_close_vcd_trace_file(tf);
        ssgen_trace_post("test.vcd");
    }

    return 0;
}
```

Open test.vcd when "-vcd" is set to command line option

Start simulation

8.1.3 memory model

Here shows memory model description (mem_rw1.h/mem_r1w1.h) generated with the specification of the command line option "- mem" when ssgen is executed.

mem_rw1.h is Single port memory model and mem_r1w1.h is 2port memory model.

2port memory model reports warning message when read access and write access are occurred at same clock cycle and same address. However, when read access has no read enable like ram2 in described above example, this warning message may be shown once immediately after simulation start. This phenomenon is produced when the first memory access is “write access to address 0”.

All memory models which are generated by ssgen have a function storing previous value of read data port. So, if you don’t expect to store previous value of read data, please change memory model directly. (Please refer to pink balloon as an example.)

mem_rw1.h (1/3)

```
//====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//====================================================
#ifndef MEM_R1W1_H
#define MEM_R1W1_H

template <typename T_ADDR, typename T_DATA, int T_LAT=1, int T_ACTW=1, int T_ACTC=T_ACTW, int T_WORD=0>
class mem_rw1 : public sc_module {
public:
    sc_in <bool> clk;
    sc_in < T_ADDR > ad1;
    sc_in < T_DATA > wd1;
    sc_in < bool > we1;
    sc_out < T_DATA > rd1;
    sc_in < bool > cs1;

    sc_signal < T_ADDR > reg_ad[T_LAT];
    sc_signal < T_DATA > reg_wd[T_LAT];
    sc_signal < bool > reg_we[T_LAT];
    sc_signal < bool > reg_cs[T_LAT];

    T_DATA* ptr_mem;

    SC_HAS_PROCESS(mem_rw1);

    mem_rw1(sc_module_name nm, T_DATA* ptr)
        : sc_module(nm)
        , clk("clk")
        , ad1("ad1")
        , wd1("wd1")
        , we1("we1")
        , rd1("rd1")
        , cs1("cs1")
        , ptr_mem(ptr)
    {
        if (T_LATENCY < 1 || T_LATENCY > 4) {
            cout << "memory latency must be from 1 to 4." << endl;
            sc_assert(1);
            exit(0);
        }

<continue to next page>
```

Class template arguments:
T_ADDR: data type of address
T_DATA: data type of data
T_LAT: latency (1～4)
T_ACTW: level of write enable (1 or 0)
T_ACTC: level of chip select (1 or 0)
T_WORD: number of words

Buffer for latency

Latency should be between 1 and 4

mem_rw1.h (2/3)

```
        T_DATA val = 0;
        int wid = val.length();
        int num = (wid + 15) >> 4;
        if (init > 0) {
            val = get_init(num, init);
        }
        for (int i = 0; i < T_WORD; i++) {
            if (init == -1) {
                val = get_init(num, init);
            }
            ptr_mem[i] = val;
        }

        SC_CTHREAD(thread_main, clk.pos());
    }

    T_DATA get_init(int num, int init) {
        T_DATA result = 0;
        sc_biguint<16> part;
        for (int i = 0; i < num; i++) {
            if (init == -1) {
                part = (sc_biguint<16>)rand();
            }
            else {
                part = (sc_biguint<16>)init;
            }

            if (i == 0) {
                result = part;
            }
            else {
                result = result | (part << (16*i));
            }
        }
        return result;
    }

    void thread_main() {
        rd1.write(0);
        for (int i = 0; i < T_LAT; i++) {
            reg_ad[i].write(0);
            reg_wd[i].write(0);
            reg_we[i].write(!T_ACTW);
            reg_cs[i].write(!T_ACTC);
        }
        wait();
        while (1) {
            if (T_LAT == 1) {
                if (T_WORD != 0) {
                    if (ad1.read() >= T_WORD) {
                        cout << "[Error @" << sc_time_stamp()
                             << "] Read/Write access over size: "
                             << name() << "'s address = " << ad1.read() << endl;
                        sc_stop();
                        wait();
                    }
                }
                if (cs1.read() == T_ACTC) {
                    if (we1.read() == T_ACTW) {
                        ptr_mem[(int)ad1.read()] = wd1.read();
                    }
                    else {
                        rd1.write(ptr_mem[(int)ad1.read()]);
                    }
                }
            }

<continue to next page>
```

Initialization of memory array

Function generating initial value

If you don' t expect to store previous value of read data, please add the following code after while(1)
rd1.write(get_init(1, -1));

When latency is 1

Out of bounds check for number of words

Write access

Read access

mem_rw1.h (3/E)

```
            else {
                reg_ad[T_LAT-2].write(ad1.read());
                reg_wd[T_LAT-2].write(wd1.read());
                reg_we[T_LAT-2].write(we1.read());
                reg_cs[T_LAT-2].write(cs1.read());
                for (int i = 0; i < T_LAT-2; i++) {
                    reg_ad[i].write(reg_ad[i+1].read());
                    reg_wd[i].write(reg_wd[i+1].read());
                    reg_we[i].write(reg_we[i+1].read());
                    reg_cs[i].write(reg_cs[i+1].read());
                }
                if (T_WORD != 0) {
                    if (reg_ad[0].read() >= T_WORD) {
                        cout << "[Error @" << sc_time_stamp()
                             << "] Read/Write access over size: "
                             << name() << "'s address = " << reg_ad[0].read() << endl;
                        sc_stop();
                        wait();
                    }
                }
                if (reg_cs[0].read() == T_ACTC) {
                    if (reg_we[0].read() == T_ACTW) {
                        ptr_mem[(int)reg_ad[0].read()] = reg_wd[0].read();
                    }
                    else {
                        rd1.write(ptr_mem[(int)reg_ad[0].read()]);
                    }
                }
            }
            wait();
        }
    }

    void vcd_trace(ssgen_trace_file* tf, int depth = HIER_MAX) {
        if (tf != 0 && depth > 0) {
            std::string nm = std::string(name());
            sc_trace(tf, clk, nm + ".clk");
            sc_trace(tf, ad1, nm + ".ad1");
            sc_trace(tf, wd1, nm + ".wd1");
            sc_trace(tf, we1, nm + ".we1");
            sc_trace(tf, rd1, nm + ".rd1");
            sc_trace(tf, cs1, nm + ".cs1");
        }
    }
};

#endif // MEM_RW1_H
```

When latency is between 2 and 4

Out of bounds check for number of words

Write access

Read access

Description of wave dump for all memory ports

mem_r1w1.h (1/5)

```
//======================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//======================================================
#ifndef MEM_R1W1_H
#define MEM_R1W1_H

template <typename T_ADDR, typename T_DATA, int T_LAT=1, int T_ACTW=1, int T_ACTC=T_ACTW, int T_ACTR=T_ACTW,
          int T_WORD=0>
class mem_r1w1 : public sc_module {
public:
    sc_in <bool> clk;
    sc_in < T_ADDR > wa1;
    sc_in < T_DATA > wd1;
    sc_in < bool > we1;
    sc_in < bool > cs1;
    sc_in < T_ADDR > ra1;
    sc_out < T_DATA > rd1;
    sc_in < bool > re1;

    sc_signal < T_ADDR > reg_wa[T_LAT];
    sc_signal < T_DATA > reg_wd[T_LAT];
    sc_signal < bool > reg_we[T_LAT];
    sc_signal < bool > reg_cs[T_LAT];
    sc_signal < T_ADDR > reg_ra[T_LAT];
    sc_signal < bool > reg_re[T_LAT];

    T_DATA* ptr_mem;
    T_ADDR  pre_addr;
    bool    pre_match;

    SC_HAS_PROCESS(mem_r1w1);

<continue to next page>
```

Class template arguments:
T_ADDR: data type of address
T_DATA: data type of data
T_LAT: latency (1～4)
T_ACTW: level of write enable (1 or 0)
T_ACTC: level of chip select (1 or 0)
T_ACTR: level of read enable (1 or 0)
T_WORD: number of words

Buffer for latency

mem_r1w1.h (2/5)

```
mem_r1w1(sc_module_name nm, T_DATA* ptr, int init = 0)
    : sc_module(nm)
    , clk("clk")
    , wa1("wa1")
    , wd1("wd1")
    , we1("we1")
    , cs1("cs1")
    , ra1("ra1")
    , rd1("rd1")
    , re1("re1")
    , ptr_mem(ptr)
    , pre_addr(0)
    , pre_match(false)
{
    if (T_LAT < 1 || T_LAT > 4) {
        cout << "memory latency must be from 1 to 4." << endl;
        sc_assert(1);
        exit(0);
    }

    T_DATA val = 0;
    int wid = val.length();
    int num = (wid + 15) >> 4;
    if (init > 0) {
        val = get_init(num, init);
    }
    for (int i = 0; i < T_WORD; i++) {
        if (init == -1) {
            val = get_init(num, init);
        }
        ptr_mem[i] = val;
    }

    SC_CTHREAD(thread_main, clk.pos());
}

T_DATA get_init(int num, int init) {
    T_DATA result = 0;
    sc_biguint<16> part;
    for (int i = 0; i < num; i++) {
        if (init == -1) {
            part = (sc_biguint<16>)rand();
        }
        else {
            part = (sc_biguint<16>)init;
        }

        if (i == 0) {
            result = part;
        }
        else {
            result = result | (part << (16*i));
        }
    }
    return result;
}
```

Latency should be between 1 and 4

Initialization of memory array

Function generating initial value

<continue to next page>

mem_r1w1.h (3/5)

```
    void thread_main() {
        rd1.write(0);
        for (int i = 0; i < T_LAT; i++) {
            reg_wa[i].write(0);
            reg_wd[i].write(0);
            reg_we[i].write(!T_ACTW);
            reg_cs[i].write(!T_ACTC);
            reg_ra[i].write(0);
            reg_re[i].write(!T_ACTR);
        }
        wait();
        while (1) {
            if (T_LAT == 1) {
                if (T_WORD != 0) {
                    if (wa1.read() >= T_WORD) {
                        cout << "[Error @" << sc_time_stamp()
                             << "] Write access over size: "
                             << name() << "'s address = " << wa1.read() << endl;
                        sc_stop();
                        wait();
                    }
                    if (ra1.read() >= T_WORD) {
                        cout << "[Error @" << sc_time_stamp()
                             << "] Read access over size: "
                             << name() << "'s address = " << ra1.read() << endl;
                        sc_stop();
                        wait();
                    }
                }
                if (we1.read() == T_ACTW && cs1.read() == T_ACTC) {
                    ptr_mem[(int)wa1.read()] = wd1.read();
                }
                if (re1.read() == T_ACTR) {
                    rd1.write(ptr_mem[(int)ra1.read()]);
                }

                // conflict check
                if (we1.read() == T_ACTW
                 && cs1.read() == T_ACTC
                 && re1.read() == T_ACTR
                 && wa1.read() == ra1.read()
                 && (pre_match == false || pre_addr != wa1.read())) {
                    pre_addr = wa1.read();
                    pre_match = true;
                    cout << "[Warning @" << sc_time_stamp()
                         << "] Read/Write access conflict: "
                         << name() << "'s address = " << wa1.read() << endl;
                }
                else {
                    pre_match =false;
                }
            }

<continue to next page>
```

If you don' t expect to store previous value of read data, please add the following code after while(1)
rd1.write(get_init(1, -1));

Out of bounds check for number of words

When latency is 1

Write access

Read access

Conflict check for Read access and Write access

mem_r1w1.h (4/5)

```
          else {
            reg_wa[T_LAT-2].write(wa1.read());
            reg_wd[T_LAT-2].write(wd1.read());
            reg_we[T_LAT-2].write(we1.read());
            reg_cs[T_LAT-2].write(cs1.read());
            reg_ra[T_LAT-2].write(ra1.read());
            reg_re[T_LAT-2].write(re1.read());

            for (int i = 0; i < T_LAT-2; i++) {
                reg_wa[i].write(reg_wa[i+1].read());
                reg_wd[i].write(reg_wd[i+1].read());
                reg_we[i].write(reg_we[i+1].read());
                reg_cs[i].write(reg_cs[i+1].read());
                reg_ra[i].write(reg_ra[i+1].read());
                reg_re[i].write(reg_re[i+1].read());
            }

            if (T_WORD != 0) {
                if (reg_wa[0].read() >= T_WORD) {
                    cout << "[Error @" << sc_time_stamp()
                         << "] Write access over size: "
                         << name() << "'s address = " << reg_wa[0].read() << endl;
                    sc_stop();
                    wait();
                }
                if (reg_ra[0].read() >= T_WORD) {
                    cout << "[Error @" << sc_time_stamp()
                         << "] Read access over size: "
                         << name() << "'s address = " << reg_ra[0].read() << endl;
                    sc_stop();
                    wait();
                }
            }

            if (reg_we[0].read() == T_ACTW && reg_cs[0].read() == T_ACTC) {
                ptr_mem[(int)reg_wa[0].read()] = reg_wd[0].read();
            }
            if (reg_re[0].read() == T_ACTR) {
                rd1.write(ptr_mem[(int)reg_ra[0].read()]);
            }

            // conflict check
            if (reg_we[0].read() == T_ACTW
             && reg_cs[0].read() == T_ACTC
             && reg_re[0].read() == T_ACTR
             && reg_wa[0].read() == reg_ra[0].read()
             && (pre_match == false || pre_addr != reg_wa[0].read())) {
                pre_addr = reg_wa[0].read();
                pre_match = true;
                cout << "[Warning @" << sc_time_stamp()
                     << "] Read/Write access conflict: "
                     << name() << "'s address = " << reg_wa[0].read() << endl;
            }
            else {
                pre_match =false;
            }
        }
        wait();
    }
}
```

When latency is between 2 and 4

Out of bounds check for number of words

Write access

Read access

Conflict check for Read access and Write access

<continue to next page>

mem_r1w1.h (5/E)

```
    void vcd_trace(ssgen_trace_file* tf, int depth = HIER_MAX) {
        if (tf != 0 && depth > 0) {
            std::string nm = std::string(name());
            sc_trace(tf, clk, nm + ".clk");
            sc_trace(tf, wa1, nm + ".wa1");
            sc_trace(tf, wd1, nm + ".wd1");
            sc_trace(tf, we1, nm + ".we1");
            sc_trace(tf, cs1, nm + ".cs1");
            sc_trace(tf, ra1, nm + ".ra1");
            sc_trace(tf, rd1, nm + ".rd1");
            sc_trace(tf, re1, nm + ".re1");
        }
    }
};

#endif // MEM_R1W1_H
```

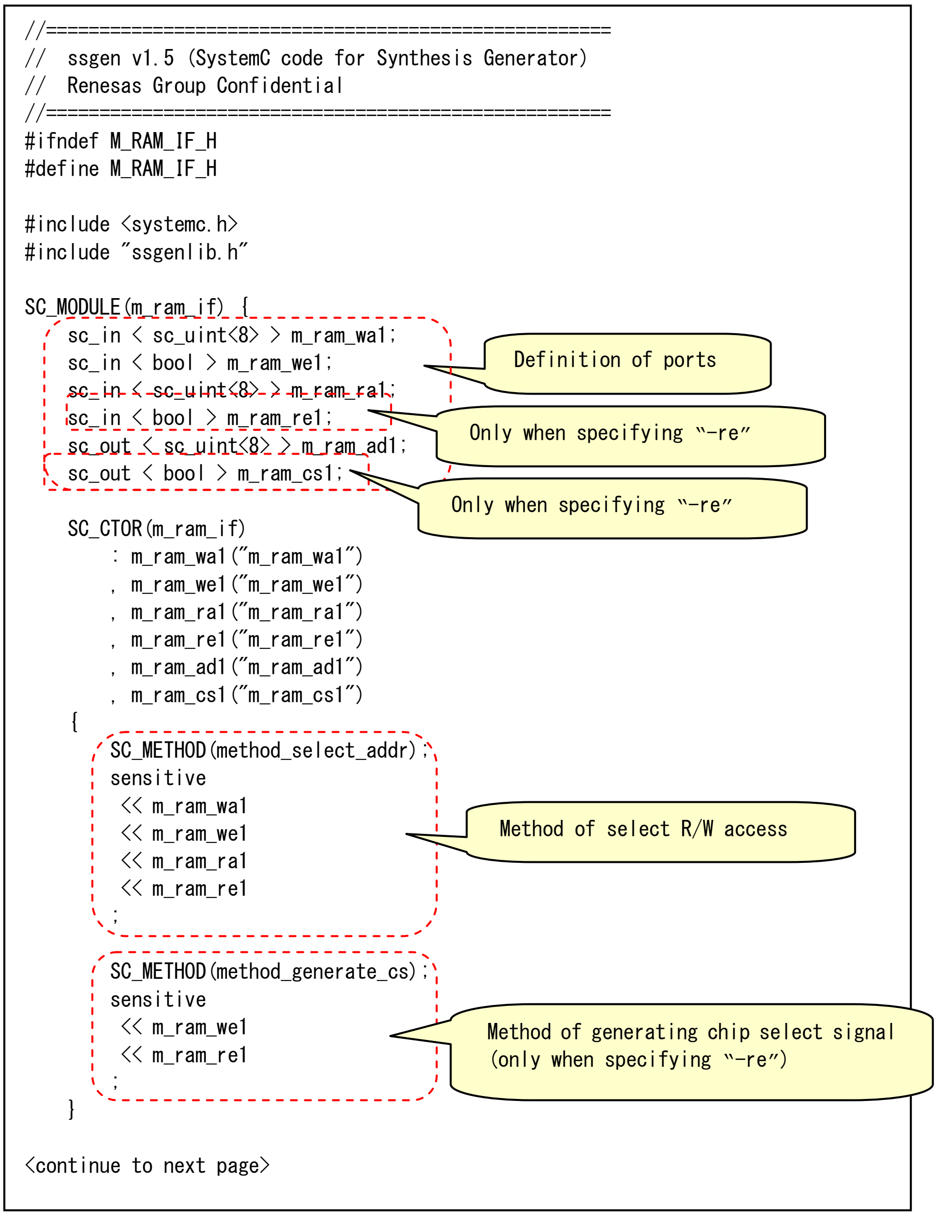
Description of wave dump for all memory ports

8.1.4 memory interface module

When specifying “rw1:r” or “rw1:w” to {u|s}mem command, in order to arbitrate between Read access and Write access, ssgen generates a memory interface module.

If you specify “umem 8 256 ram rw1:r –re –prefix=m_”, the following module is generated.

m_ram_if.h (1/2)



```
//====================================================
//  ssgen v1.5 (SystemC code for Synthesis Generator)
//  Renesas Group Confidential
//====================================================
#ifndef M_RAM_IF_H
#define M_RAM_IF_H

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

SC_MODULE(m_ram_if) {
    sc_in < sc_uint<8> > m_ram_wa1;
    sc_in < bool > m_ram_we1;
    sc_in < sc_uint<8> > m_ram_ra1;
    sc_in < bool > m_ram_re1;
    sc_out < sc_uint<8> > m_ram_ad1;
    sc_out < bool > m_ram_cs1;

    SC_CTOR(m_ram_if)
        : m_ram_wa1("m_ram_wa1")
        , m_ram_we1("m_ram_we1")
        , m_ram_ra1("m_ram_ra1")
        , m_ram_re1("m_ram_re1")
        , m_ram_ad1("m_ram_ad1")
        , m_ram_cs1("m_ram_cs1")
    {
        SC_METHOD(method_select_addr);
        sensitive
         << m_ram_wa1
         << m_ram_we1
         << m_ram_ra1
         << m_ram_re1
        ;

        SC_METHOD(method_generate_cs);
        sensitive
         << m_ram_we1
         << m_ram_re1
        ;
    }

<continue to next page>
```

Definition of ports
Only when specifying "-re"
Only when specifying "-re"
Method of select R/W access
Method of generating chip select signal
(only when specifying "-re")

m_ram_if.h (2/E)

```
#if !defined(__CTOS__) && !defined(CALYPTO_SYSC)
    void vcd_trace(ssgen_trace_file* tf, int depth = HIER_MAX) {
        if (tf != 0 && depth > 0) {
            std::string nm = std::string(name());
#ifdef _MEM_MODEL
            sc_trace(tf, m_ram_wa1, nm + ".m_ram_wa1");
            sc_trace(tf, m_ram_we1, nm + ".m_ram_we1");
            sc_trace(tf, m_ram_ra1, nm + ".m_ram_ra1");
            sc_trace(tf, m_ram_re1, nm + ".m_ram_re1");
            sc_trace(tf, m_ram_ad1, nm + ".m_ram_ad1");
            sc_trace(tf, m_ram_cs1, nm + ".m_ram_cs1");
#endif
        }
    }
#endif // !defined(__CTOS__) && !defined(CALYPTO_SYSC)

    void method_select_addr();
    void method_generate_cs();
};

#ifdef __CTOS__
SC_MODULE_EXPORT(m_ram_if);
#endif

#endif // M_RAM_IF_H#endif // M_RAM_IF_H
```

Description of wave dump
for all ports

m_ram_if.cpp

```
//=====================================================
//  ssgen v1.5 (SystemC code for Synthesis Generator)
//  Renesas Group Confidential
//=====================================================
// m_ram_if.cpp
#include "m_ram_if.h"

void m_ram_if::method_select_addr() {
    if (m_ram_we1.read() == 1)
        m_ram_ad1.write(m_ram_wa1.read());
    else if (m_ram_re1.read() == 1)
        m_ram_ad1.write(m_ram_ra1.read());
    else m_ram_ad1.write(0);
}

void m_ram_if::method_generate_cs() {
    if (m_ram_we1.read() == 1 && m_ram_re1.read() == 1) {
        cout << "[Warning @" << sc_time_stamp()
         << "] Read/Write access conflict: "
         << name() << "'s address = " << m_ram_wa1.read() << endl;
    }
    m_ram_cs1.write(m_ram_we1.read() | m_ram_re1.read());
}
```

Select R/W address
Write access has a priority over Read access

Conflict check between R/W accesses

Generate chip select signal
(only when "-re" option)

8.1.5 CtoS script

Here shows CtoS script (run_ctos_test.sh, ctos_test.tcl) generated from module definition file test.in with the specification of command line option “-ctos” when ssgen is executed.

Method of executing CtoS

%s> run_ctos_test.sh

run_ctos_test.sh

```
#!/bin/csh -f
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
source /common/appl/dotfiles/cadence.CSHRC_ctos_v12.20-s201
bs -M 500 -os RHEL5 ctos ctos_test.tcl -log ctos_test.log
```

Use CtoS v12.2 by default
It is possible to change CtoS setting file by env_ctos command.

ctos_test.tcl (1/2)

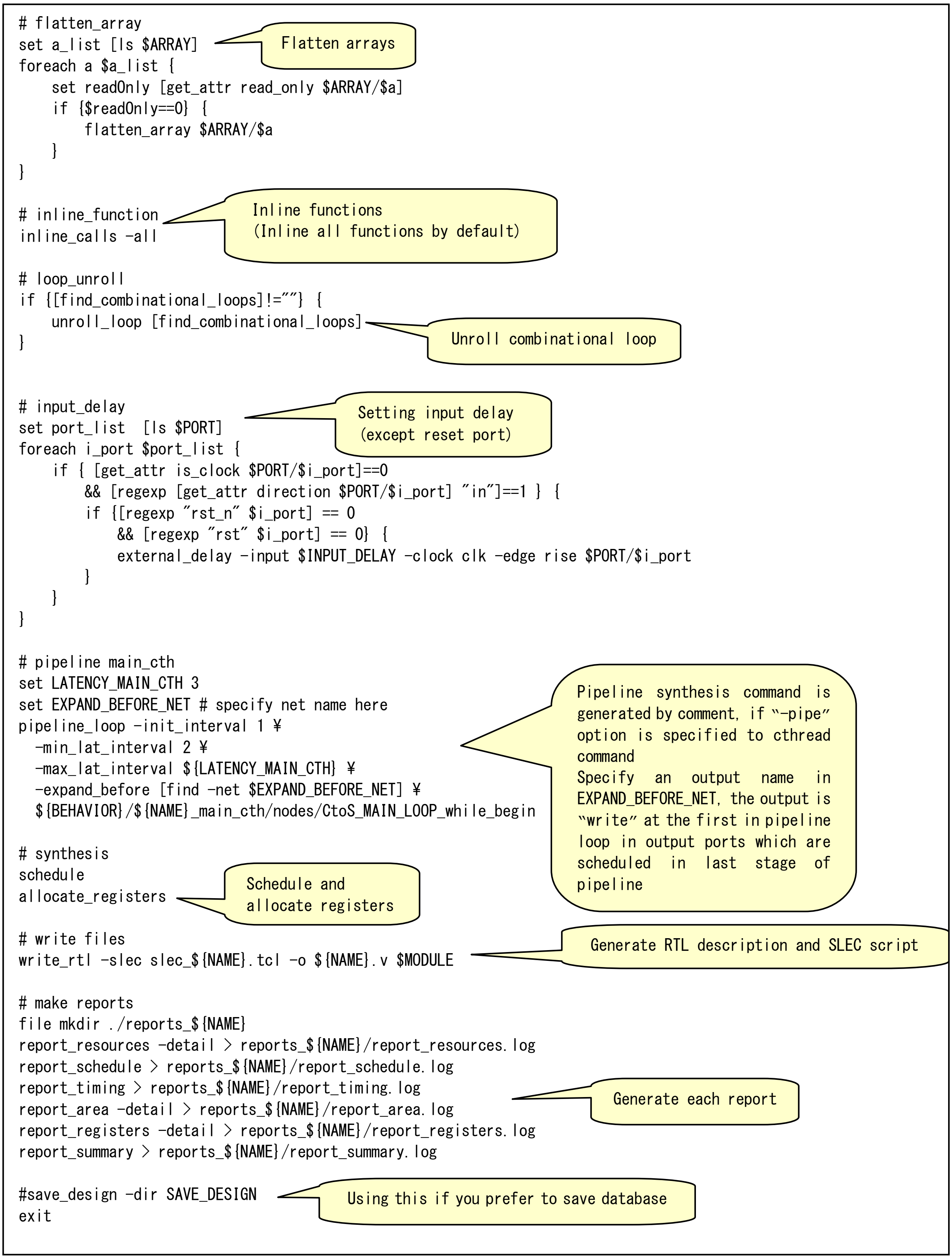
```
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
# parameters
set PERIOD 5000
set INPUT_DELAY 0
set TARGET_LIB tutorial.lbr

# set variables
set NAME test
set MODULE /designs/$NAME/modules/$NAME
set ARRAY $MODULE/arrays
set BEHAVIOR $MODULE/behaviors
set PORT $MODULE/terms

# preparation
new_design $NAME
set_attr source_files "test.cpp" [get_design]
set_attr compile_flags " -w -D_MEM_MODEL -I/common/appl/Renesas/SystemC/utility/ssgen" [get_design]
set_attr top_module_path $NAME [get_design]
set_attr verilog_rtl_model_suffix "" [get_design]
set_attr auto_write_models false [get_design]
set_attr low_power_clock_gating true [get_design]
set_attr tech_lib_names $TARGET_LIB [get_design]
set_attr reset_all_registers true [get_design]
set_attr verilog_pragma_keyword "synopsys" [get_design]
#set_attr enable_resource_sharing false [get_design]
#set_attr default_speed_grade 90 [get_design]

# slec attribute
set_attr enable_var_correspondences true [get_design]
#set_attr enable_slec_verification true [get_design]
#set_attr optimize_enable_propagate_function_args false [get_design]

define_clock -name clk -period $PERIOD
build

<continue to next page>
```

Setting clock period and input delay
Default value is 5000ps and 0ps

Setting technology library file(default is tutorial)

Module name

ctos_test.tcl (2/E)



```
# flatten_array
set a_list [ls $ARRAY]
foreach a $a_list {
    set readOnly [get_attr read_only $ARRAY/$a]
    if {$readOnly==0} {
        flatten_array $ARRAY/$a
    }
}

# inline_function
inline_calls -all

# loop_unroll
if {[find_combinational_loops]!=""} {
    unroll_loop [find_combinational_loops]
}


# input_delay
set port_list  [ls $PORT]
foreach i_port $port_list {
    if { [get_attr is_clock $PORT/$i_port]==0
        && [regexp [get_attr direction $PORT/$i_port] "in"]==1 } {
        if {[regexp "rst_n" $i_port] == 0
            && [regexp "rst" $i_port] == 0} {
            external_delay -input $INPUT_DELAY -clock clk -edge rise $PORT/$i_port
        }
    }
}

# pipeline main_cth
set LATENCY_MAIN_CTH 3
set EXPAND_BEFORE_NET # specify net name here
pipeline_loop -init_interval 1 ¥
  -min_lat_interval 2 ¥
  -max_lat_interval ${LATENCY_MAIN_CTH} ¥
  -expand_before [find -net $EXPAND_BEFORE_NET] ¥
  ${BEHAVIOR}/${NAME}_main_cth/nodes/CtoS_MAIN_LOOP_while_begin

# synthesis
schedule
allocate_registers

# write files
write_rtl -slec slec_${NAME}.tcl -o ${NAME}.v $MODULE

# make reports
file mkdir ./reports_${NAME}
report_resources -detail > reports_${NAME}/report_resources.log
report_schedule > reports_${NAME}/report_schedule.log
report_timing > reports_${NAME}/report_timing.log
report_area -detail > reports_${NAME}/report_area.log
report_registers -detail > reports_${NAME}/report_registers.log
report_summary > reports_${NAME}/report_summary.log

#save_design -dir SAVE_DESIGN
exit
```

Flatten arrays

Inline functions
(Inline all functions by default)

Unroll combinational loop

Setting input delay
(except reset port)

Pipeline synthesis command is generated by comment, if "-pipe" option is specified to cthread command
Specify an output name in EXPAND_BEFORE_NET, the output is "write" at the first in pipeline loop in output ports which are scheduled in last stage of pipeline

Schedule and
allocate registers

Generate RTL description and SLEC script

Generate each report

Using this if you prefer to save database

8.1.6 SLEC script

Here shows SLEC script (run_slec_test_sc.sh, slec_test_sc.tcl, run_slec_test_eq.sh) generated from module definition file test.in with the specification of command line option “-slec” when ssgen is executed.

Method of executing SLEC

%s> run_slec_test_sc.sh ## SystemC X-propagation check

%s> run_slec_test_eq.sh ## SystemC-RTL equivalence check (Need to execute CtoS before SLEC)

run_slec_test_sc.sh

```
#!/bin/csh -f
##====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##====================================================
source /common/appl/dotfiles/slec.CSHRC_7.1-prod-5
bs -os RHEL5 -M 500 slec slec_test_sc.tcl -w calypto_test_sc -l slec_test_sc.log
```

Use SLEC 7.1-prod-5 by default
It is possible to change SLEC setting file by env_slec command.

slec_test_sc.tcl

```
##====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##====================================================
set_verification_mode -system_manual -property_checks
set_global exit_on_error 1
set_global solver_cache_location none
set_global flop_checking_at_reset concrete
set_global osci_compliant_initial_value 0
config_trace_files -simdump -auxsignals -dumpmem

limit -realtime 2h

build_design -spec -w -I/common/appl/Renesas/SystemC/utility/ssgen -D_MEM_MODEL -DASYNC_RESET_SUPPORT test.cpp
build_design -impl -w -I/common/appl/Renesas/SystemC/utility/ssgen -D_MEM_MODEL -DASYNC_RESET_SUPPORT test.cpp

create_waveform -name ACTIVE_HIGH -bitwidth 1 {1}
create_waveform -name ACTIVE_LOW -bitwidth 1 {0}
create_waveform -name ALWAYS_LOW  -bitwidth 1 {0+}
create_waveform -name ALWAYS_HIGH -bitwidth 1 {1+}

create_constraint -reset -waveform ACTIVE_HIGH spec.rst
create_constraint -reset -waveform ACTIVE_HIGH impl.rst
create_constraint -waveform ALWAYS_LOW spec.rst
create_constraint -waveform ALWAYS_LOW impl.rst

create_constraint -reset -waveform ACTIVE_LOW spec.rst_n
create_constraint -reset -waveform ACTIVE_LOW impl.rst_n
create_constraint -waveform ALWAYS_HIGH spec.rst_n
create_constraint -waveform ALWAYS_HIGH impl.rst_n

create_namemap_rule -default
foreach var [find -inst -seq -spec -hier -short_name -attr "local_var_attr"] {
  unmap -flop spec.$var impl.$var
}

check_properties -spec -prop -oob_access
#check_properties -spec -prop -user_assert

verify -mode full_proof
quit
```

run_slec_test_eq.sh

```
#!/bin/csh -f
##======================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##======================================================
source /common/appl/dotfiles/slec.CSHRC_7.1-prod-5

set MODULE = test

rm -rf SLEC_*.tcl

bs -os RHEL5 -M 50 ctos2slec -min_latency_maps slec_${MODULE}.tcl.xml slec_${MODULE}_eq.tcl

perl -i -pe 's/^(set_verification_mode .*)/$1¥nlimit -realtime 2h/' slec_${MODULE}_eq.tcl
perl -i -pe 's/^(set_verification_mode .*)/$1¥nset_global ldb_gen_maps 0/' slec_${MODULE}_eq.tcl
perl -i -pe 's/^(set_verification_mode .*)/$1¥nset_global flop_checking_at_reset concrete/' slec_${MODULE}_eq.tcl
perl -i -pe 's/^(set_verification_mode .*)/$1¥nset_global osci_compliant_initial_value 0/' slec_${MODULE}_eq.tcl
perl -i -pe 's/^(set_verification_mode .*)/$1¥nset_global solver_cache_location none/' slec_${MODULE}_eq.tcl

bs -os RHEL5 -M 500 slec slec_${MODULE}_eq.tcl -w calypto_${MODULE}_eq -l slec_${MODULE}_eq.log
```

Use SLEC 7.1-prod-5 by default
It is possible to change SLEC setting file by
env_slec command.

CtoS generate XML file for SLEC

8.1.7 Checker script

Here shows 1Team:System script (run_1team.sh) and SSChecker script (run_sschecker_test.sh) generated from module definition file test.in with the specification of command line option “-checker” when ssgen is executed.

Method of executing 1Team:System

%s> run_1team.sh

Method of executing SSChecker

%s> run_sschecker_test.sh

run_1team.sh

```
#!/bin/csh -f
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
source /common/appl/dotfiles/1TeamSystem.CSHRC_1.16.7
bs -M 500 -os RHEL5 spyglass -template=Renesas/Synth ¥
   -I/common/appl/Renesas/SystemC/utility/ssgen ¥
   -D_MEM_MODEL ¥
   *.cpp
```

Use 1Team:System 1.16.7 by default
It is possible to change 1Team:System setting file by env_1team command.

run_sschecker_test.sh

```
#!/bin/csh -f
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
set SSCHECKER = /common/appl/Renesas/SystemC/utility/SSChecker/SSChecker.pl
$SSCHECKER test.cpp -rpt test.rpt
```

Use SSChecker of REL-EWS by default
It is possible to change path of SSChecker by env_sschecker command.

8.1.8 Macro function for memory access

Here shows the table of macro functions for the memory access generated to SystemC module description and testbench description by specifying umem/smem command. The macro function name is different according to memory type (rw1/r1w1:r/r1w1:w/rw1:r/rw1:w/rw2:a/rw2:b), enable generation existence (-cs/-re option specification existence), -nowd/-noad option specification existence and -ponly option specification existence.

2 port memory (r1w1:r, r1w1:w) of switching array access and port access (-ponly is not specified)

Memory type	2port read (r1w1:r)		2port write (r1w1:w)	
Enable generation	without -re	with -re	without -cs	with -cs
Memory pointer	MEM_PTR (for module) / MEM_PTR_A (for testbench)			
Define port&array	MEM_DEF_2R	MEM_DEF_2R_E	MEM_DEF_2W	MEM_DEF_2W_E
Initialize port name	MEM_ININM_2R	MEM_ININM_2R_E	MEM_ININM_2W	MEM_ININM_2W_E
Initialize value	MEM_INIVAL_2R[(*2)]	MEM_INIVAL_2R_E[(*2)]	MEM_INIVAL_2W[(*1)]	MEM_INIVAL_2W_E[(*1)]
Negate enable	-	MEM_NEG_2R_E	MEM_NEG_2W	MEM_NEG_2W_E
Register Pipe function	MEM_PIPE_2R_CTOR		MEM_PIPE_CTOR	
Define Pipe function	MEM_PIPE_2R		MEM_PIPE	
Request function	MEM_REQ_2[(*2)]	MEM_REQ_2_E[(*2)]	-	
Read function	MEM_RD_2		-	
Write function	-		MEM_WR_2[(*1)]	MEM_WR_2_E[(*1)]

(*1) [_NOWD] when specify "–nowd" option, [_NOAD] when specify "-noad" option, [_NOAW] when specify "-nowd" option and "-noad" option

(*2) [_NOAD] when specify "-noad" option

2 port memory (r1w1:r, r1w1:w) of only port access (-ponly is specified)

Memory type	2port read (r1w1:r)		2port write (r1w1:w)	
Enable generation	without -re	with -re	without -cs	with -cs
Memory pointer	None (for module) / MEM_PTR_A (for testbench)			
Define port&array	MEM_DEF_2R_P	MEM_DEF_2R_E_P	MEM_DEF_2W_P	MEM_DEF_2W_E_P
Initialize port name	MEM_ININM_2R_P	MEM_ININM_2R_E_P	MEM_ININM_2W_P	MEM_ININM_2W_E_P
Initialize value	MEM_INIVAL_2R[(*2)]_P	MEM_INIVAL_2R_E[(*2)]_P	MEM_INIVAL_2W[(*1)]_P	MEM_INIVAL_2W_E[(*1)]_P
Negate enable	-	MEM_NEG_2R_E_P	MEM_NEG_2W_P	MEM_NEG_2W_E_P
Request function	MEM_REQ_2[(*2)]_P	MEM_REQ_2_E[(*2)]_P	-	
Read function	MEM_RD_2_P		-	
Write function	-		MEM_WR_2[(*1)]_P	MEM_WR_2_E[(*1)]_P

(*1) [_NOWD] when specify "–nowd" option, [_NOAD] when specify "-noad" option, [_NOAW] when specify "-nowd" option and "-noad" option

(*2) [_NOAD] when specify "-noad" option

Single port memory (rw1, rw1:r, rw1:w) and dual port memory (rw2:a, rw2:b) of switching array access and port access (-ponly is not specified)

Memory type	Single port and dual port (rw1, rw2:a, rw2:b)		Single port read (rw1:r)		Single port write (rw1:w)
Enable generation	without -cs	with -cs	without -re	with -re	without -cs
Memory pointer	MEM_PTR (for module) / MEM_PTR_A (for testbench)				
Define port&array	MEM_DEF_1	MEM_DEF_1_E	MEM_DEF_1R	MEM_DEF_1R_E	MEM_DEF_1W
Initialize port name	MEM_ININM_1	MEM_ININM_1_E	MEM_ININM_1R	MEM_ININM_1R_E	MEM_ININM_1W
Initialize value	MEM_INIVAL_1[(*1)]	MEM_INIVAL_1_E[(*1)]	MEM_INIVAL_1R	MEM_INIVAL_1R_E	MEM_INIVAL_1W
Negate enable	MEM_NEG_1	MEM_NEG_1_E	-	MEM_NEG_1R_E	MEM_NEG_1W
Register Pipe function	MEM_PIPE_CTOR		MEM_PIPE_2R_CTOR		MEM_PIPE_CTOR
Define Pipe function	MEM_PIPE		MEM_PIPE_2R		MEM_PIPE
Request function	MEM_REQ_1[(*2)]	MEM_REQ_1_E[(*2)]	MEM_REQ_1R	MEM_REQ_1R_E	-
Read function	MEM_RD_1		MEM_RD_1R		-
Write function	MEM_WR_1[(*1)]	MEM_WR_1_E[(*1)]	-		MEM_WR_1

(*1) [_NOWD] when specify "–nowd" option, [_NOAD] when specify "-noad" option, [_NOAW] when specify "-nowd" option and "-noad" option

(*2) [_NOAD] when specify "-noad" option

Single port memory (rw1, rw1:r, rw1:w) and dual port memory (rw2:a, rw2:b) of only port access (-ponly is specified)

Memory type	Single port and dual port (rw1, rw2:a, rw2:b)		Single port read (rw1:r)		Single port write (rw1:w)
Enable generation	without -cs	with -cs	without -re	with -re	without -cs
Memory pointer	None (for module) / MEM_PTR_A (for testbench)				
Define port&array	MEM_DEF_1_P	MEM_DEF_1_E_P	MEM_DEF_1R_P	MEM_DEF_1R_E_P	MEM_DEF_1W_P
Initialize port name	MEM_ININM_1_P	MEM_ININM_1_E_P	MEM_ININM_1R_P	MEM_ININM_1R_E_P	MEM_ININM_1W_P
Initialize value	MEM_INIVAL_1_[(*1)]P	MEM_INIVAL_1_E[(*1)]_P	MEM_INIVAL_1R_P	MEM_INIVAL_1R_E_P	MEM_INIVAL_1W_P
Negate enable	MEM_NEG_1_P	MEM_NEG_1_E_P	-	MEM_NEG_1R_E_P	MEM_NEG_1W_P
Request function	MEM_REQ_1[(*2)]_P	MEM_REQ_1_E[(*2)]_P	MEM_REQ_1R_P	MEM_REQ_1R_E_P	-
Read function	MEM_RD_1_P		MEM_RD_1R_P		-
Write function	MEM_WR_1[(*1)] _P	MEM_WR_1_E[(*1)]_P	-		MEM_WR_1_P

(*1) [_NOWD] when specify "–nowd" option, [_NOAD] when specify "-noad" option, [_NOAW] when specify "-nowd" option and "-noad" option

(*2) [_NOAD] when specify "-noad" option

All macro functions are described in ssgenlib.h that is the library header of ssgen. Please refer to Chapter 9 for details of each macro function.

8.2 Example of output file of hierarchy generation mode

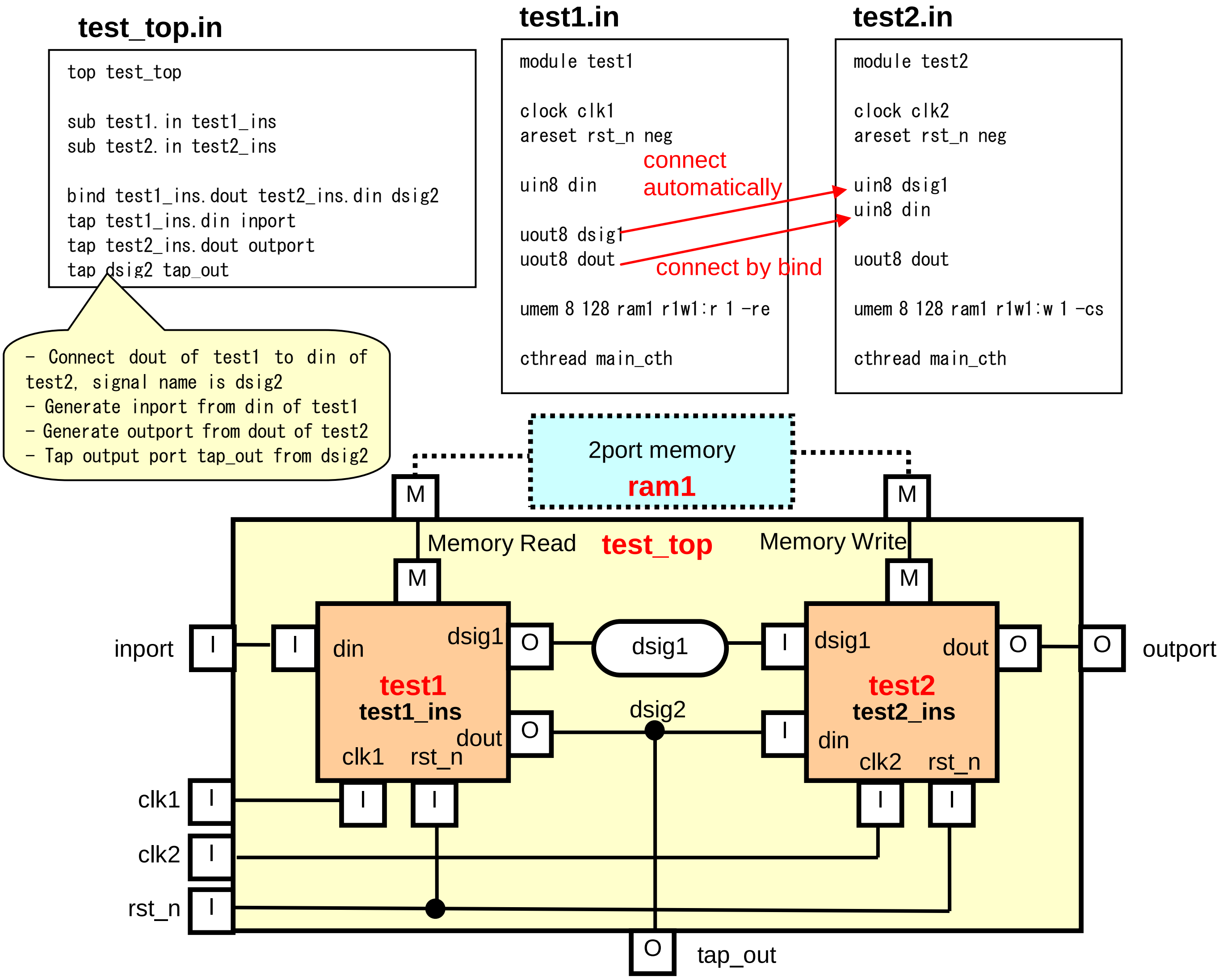
Here shows the SystemC description, the OSCI-Sim script, the VCS-MX script, the IES script, and the CtoS script of module test_top generated from hierarchy definition file test_top.in. The execution of ssgen is

```
%s> ssgen.pl -in test_top.in -mem -osci -vcs -ies -ctos
```

Though testbench and memory model are generated, the explanation is omitted because it is similar to module generation mode (Only think test_top hierarchy to be DUT) .

Moreover, in hierarchy generation mode, a module definition file equivalent to module test_top is generated, shows it also.

In this example, the following hierarchy definition file (test_top.in) and two module definition files (test1.in,test2.in) as lower hierarchy are used.



test_top.in

```
top test_top

sub test1.in test1_ins
sub test2.in test2_ins

bind test1_ins.dout test2_ins.din dsig2
tap test1_ins.din inport
tap test2_ins.dout outport
tap dsig2 tap_out
```

- Connect dout of test1 to din of test2, signal name is dsig2
- Generate inport from din of test1
- Generate outport from dout of test2
- Tap output port tap_out from dsig2

test1.in

```
module test1

clock clk1
areset rst_n neg

uin8 din

uout8 dsig1
uout8 dout

umem 8 128 ram1 r1w1:r 1 -re

cthread main_cth
```

test2.in

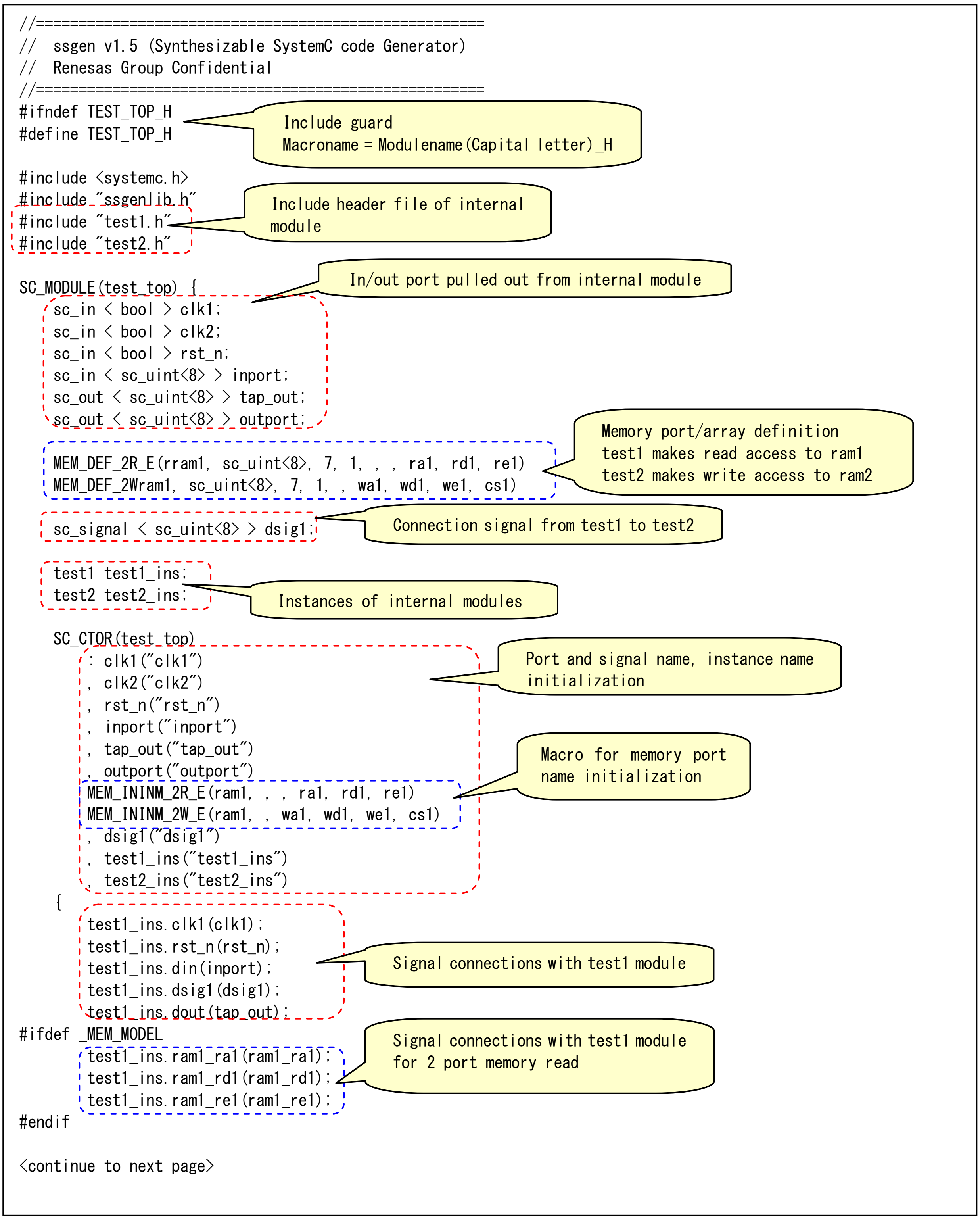
```
module test2

clock clk2
areset rst_n neg

uin8 dsig1
uin8 din

uout8 dout

umem 8 128 ram1 r1w1:w 1 -cs

cthread main_cth
```

Module framework

8.2.1 SystemC description of hierarchy module

Here shows SystemC description of Hierarchy module (test_top.h/test_top.cpp) generated from hierarchy definition file test_top.in.

test_top.h (1/2)



```
//=====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=====================================================
#ifndef TEST_TOP_H
#define TEST_TOP_H

#include <systemc.h>
#include ”ssgenlib.h”
#include ”test1.h”
#include ”test2.h”

SC_MODULE(test_top) {
    sc_in < bool > clk1;
    sc_in < bool > clk2;
    sc_in < bool > rst_n;
    sc_in < sc_uint<8> > inport;
    sc_out < sc_uint<8> > tap_out;
    sc_out < sc_uint<8> > outport;

    MEM_DEF_2R_E(rram1, sc_uint<8>, 7, 1, , , ra1, rd1, re1)
    MEM_DEF_2Wram1, sc_uint<8>, 7, 1, , wa1, wd1, we1, cs1)

    sc_signal < sc_uint<8> > dsig1;

    test1 test1_ins;
    test2 test2_ins;

    SC_CTOR(test_top)
        : clk1(”clk1”)
        , clk2(”clk2”)
        , rst_n(”rst_n”)
        , inport(”inport”)
        , tap_out(”tap_out”)
        , outport(”outport”)
        MEM_ININM_2R_E(ram1, , , ra1, rd1, re1)
        MEM_ININM_2W_E(ram1, , wa1, wd1, we1, cs1)
        , dsig1(”dsig1”)
        , test1_ins(”test1_ins”)
        , test2_ins(”test2_ins”)
    {
        test1_ins.clk1(clk1);
        test1_ins.rst_n(rst_n);
        test1_ins.din(inport);
        test1_ins.dsig1(dsig1);
        test1_ins.dout(tap_out);
#ifdef _MEM_MODEL
        test1_ins.ram1_ra1(ram1_ra1);
        test1_ins.ram1_rd1(ram1_rd1);
        test1_ins.ram1_re1(ram1_re1);
#endif

<continue to next page>
```

Include guard
Macroname = Modulename(Capital letter)_H

Include header file of internal module

In/out port pulled out from internal module

Memory port/array definition
test1 makes read access to ram1
test2 makes write access to ram2

Connection signal from test1 to test2

Instances of internal modules

Port and signal name, instance name initialization

Macro for memory port name initialization

Signal connections with test1 module

Signal connections with test1 module
for 2 port memory read

test_top.h (2/E)

```
        test2_ins.clk2(clk2);
        test2_ins.rst_n(rst_n);
        test2_ins.dsig1(dsig1);
        test2_ins.din(tap_out);
        test2_ins.dout(outport);
#ifdef _MEM_MODEL
        test2_ins.ram1_wa1(ram1_wa1);
        test2_ins.ram1_wd1(ram1_wd1);
        test2_ins.ram1_we1(ram1_we1);
        test2_ins.ram1_cs1(ram1_cs1);
#endif
    }

#if !defined(__CTOS__) && !defined(CALYPTO_SYSC)
    void vcd_trace(ssgen_trace_file* tf, int depth = HIER_MAX) {
        if (tf != 0 && depth > 0) {
            std::string nm = std::string(name());
            sc_trace(tf, clk1, nm + ”.clk1”);
            sc_trace(tf, clk2, nm + ”.clk2”);
            sc_trace(tf, rst_n, nm + ”.rst_n”);
            sc_trace(tf, inport, nm + ”.inport”);
            sc_trace(tf, tap_out, nm + ”.tap_out”);
            sc_trace(tf, outport, nm + ”.outport”);
            sc_trace(tf, dsig1, nm + ”.dsig1”);
#ifdef _MEM_MODEL
            sc_trace(tf, ram1_ra1, nm + ”.ram1_ra1”);
            sc_trace(tf, ram1_rd1, nm + ”.ram1_rd1”);
            sc_trace(tf, ram1_re1, nm + ”.ram1_re1”);
#endif
#ifdef _MEM_MODEL
            sc_trace(tf, ram1_wa1, nm + ”.ram1_wa1”);
            sc_trace(tf, ram1_wd1, nm + ”.ram1_wd1”);
            sc_trace(tf, ram1_we1, nm + ”.ram1_we1”);
            sc_trace(tf, ram1_cs1, nm + ”.ram1_cs1”);
#endif
            test1_ins.vcd_trace(tf, depth-1);
            test2_ins.vcd_trace(tf, depth-1);
        }
    }
#endif // !defined(__CTOS__) && !defined(CALYPTO_SYSC)
};

#ifdef __CTOS__
SC_MODULE_EXPORT(test_top);
#endif

#endif // TEST_TOP_H
```

Signal connections with test2 module

Signal connections with test2 module
for 2 port memory write

Description of wave dump
for port and signal

Description of wave
dump for memory port

Call wave dump function of
lower hierarchy

Necessary
description for
CtoS execution

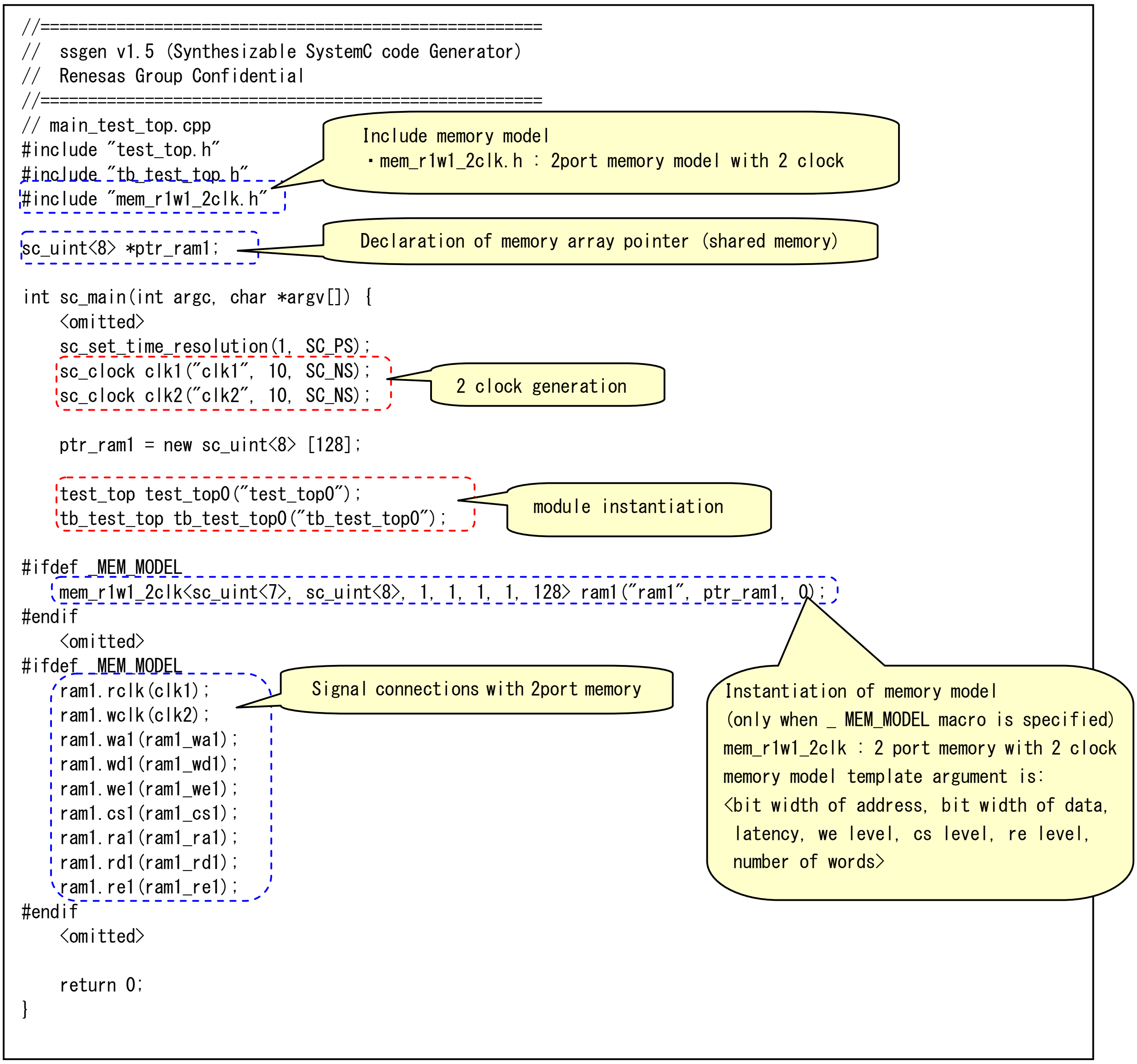
test_top.cpp

```
//=====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=====================================================
// test_top.cpp
#include ”test_top.h”
```

8.2.2 testbench description

tb_test_top.h, tb_test_top.cpp and main_test_top.cpp are generated as testbench file from hierarchy definition file test_top.in. In this example, because clock of test1 (clk1) has different name with clock of test2 (clk2), two clock generation descriptions are generated in testbench while 2 clocks are connected to shared memory (ram1). Here shows clock generation description, memory model instantiation and port connections of memory model.

main_test_top.cpp



```
//=====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=====================================================
// main_test_top.cpp
#include "test_top.h"
#include "tb_test_top.h"
#include "mem_r1w1_2clk.h"

sc_uint<8> *ptr_ram1;

int sc_main(int argc, char *argv[]) {
    <omitted>
    sc_set_time_resolution(1, SC_PS);
    sc_clock clk1("clk1", 10, SC_NS);
    sc_clock clk2("clk2", 10, SC_NS);

    ptr_ram1 = new sc_uint<8> [128];

    test_top test_top0("test_top0");
    tb_test_top tb_test_top0("tb_test_top0");

#ifdef _MEM_MODEL
    mem_r1w1_2clk<sc_uint<7>, sc_uint<8>, 1, 1, 1, 1, 128> ram1("ram1", ptr_ram1, 0);
#endif
    <omitted>
#ifdef _MEM_MODEL
    ram1.rclk(clk1);
    ram1.wclk(clk2);
    ram1.wa1(ram1_wa1);
    ram1.wd1(ram1_wd1);
    ram1.we1(ram1_we1);
    ram1.cs1(ram1_cs1);
    ram1.ra1(ram1_ra1);
    ram1.rd1(ram1_rd1);
    ram1.re1(ram1_re1);
#endif
    <omitted>

    return 0;
}
```

Include memory model
・mem_r1w1_2clk.h : 2port memory model with 2 clock

Declaration of memory array pointer (shared memory)

2 clock generation

module instantiation

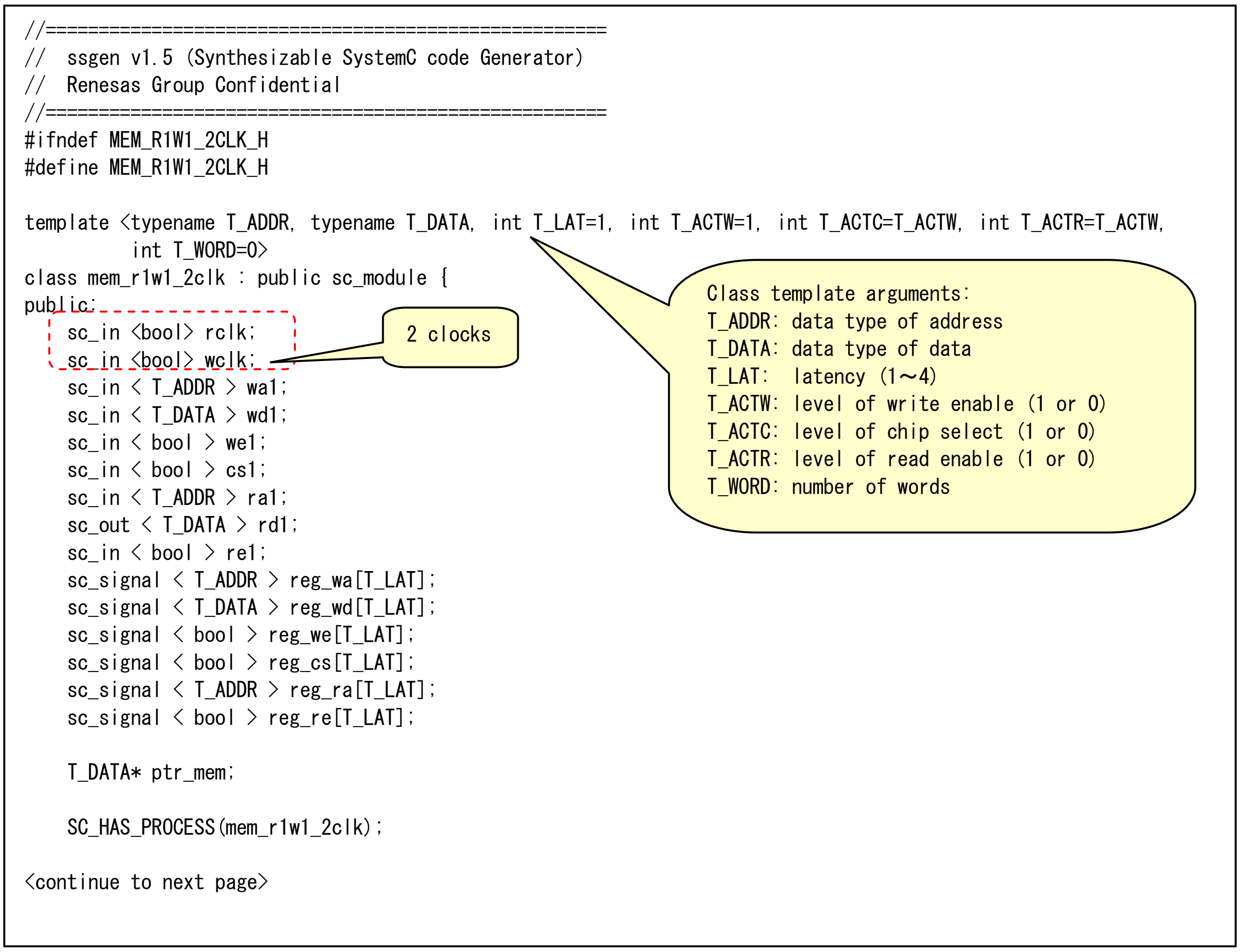
Signal connections with 2port memory

Instantiation of memory model
(only when _ MEM_MODEL macro is specified)
mem_r1w1_2clk : 2 port memory with 2 clock
memory model template argument is:
<bit width of address, bit width of data,
latency, we level, cs level, re level,
number of words>

8.2.3 memory model

Here shows 2port memory model with 2 clock (mem_r1w1_2clk.h) generated with the specification of the command line option "- mem" when ssgen is executed. This model has read access thread and write access thread respectively, because clock of read access is different to clock of write access. Thus, this model cannot detect conflict access between read and write.

mem_r1w1_2clk.h (1/5)



```
//=====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//=====================================================
#ifndef MEM_R1W1_2CLK_H
#define MEM_R1W1_2CLK_H

template <typename T_ADDR, typename T_DATA, int T_LAT=1, int T_ACTW=1, int T_ACTC=T_ACTW, int T_ACTR=T_ACTW,
          int T_WORD=0>
class mem_r1w1_2clk : public sc_module {
public:
    sc_in <bool> rclk;
    sc_in <bool> wclk;
    sc_in < T_ADDR > wa1;
    sc_in < T_DATA > wd1;
    sc_in < bool > we1;
    sc_in < bool > cs1;
    sc_in < T_ADDR > ra1;
    sc_out < T_DATA > rd1;
    sc_in < bool > re1;
    sc_signal < T_ADDR > reg_wa[T_LAT];
    sc_signal < T_DATA > reg_wd[T_LAT];
    sc_signal < bool > reg_we[T_LAT];
    sc_signal < bool > reg_cs[T_LAT];
    sc_signal < T_ADDR > reg_ra[T_LAT];
    sc_signal < bool > reg_re[T_LAT];

    T_DATA* ptr_mem;

    SC_HAS_PROCESS(mem_r1w1_2clk);

<continue to next page>
```

2 clocks

Class template arguments:
T_ADDR: data type of address
T_DATA: data type of data
T_LAT: latency (1～4)
T_ACTW: level of write enable (1 or 0)
T_ACTC: level of chip select (1 or 0)
T_ACTR: level of read enable (1 or 0)
T_WORD: number of words

mem_r1w1_2clk.h (2/5)

```
    mem_r1w1_2clk(sc_module_name nm, T_DATA* ptr, int init = 0)
        : sc_module(nm)
        , rclk("rclk")
        , wclk("wclk")
        , wa1("wa1")
        , wd1("wd1")
        , we1("we1")
        , cs1("cs1")
        , ra1("ra1")
        , rd1("rd1")
        , re1("re1")
        , ptr_mem(ptr)
    {
        if (T_LAT < 1 || T_LAT > 4) {
            cout << "memory latency must be from 1 to 4." << endl;
            sc_assert(1);
            exit(0);
        }

        T_DATA val = 0;
        int wid = val.length();
        int num = (wid + 15) >> 4;
        if (init > 0) {
            val = get_init(num, init);
        }
        for (int i = 0; i < T_WORD; i++) {
            if (init == -1) {
                val = get_init(num, init);
            }
            ptr_mem[i] = val;
        }

        SC_CTHREAD(thread_read, rclk.pos());
        SC_CTHREAD(thread_write, wclk.pos());
    }

    T_DATA get_init(int num, int init) {
        T_DATA result = 0;
        sc_biguint<16> part;
        for (int i = 0; i < num; i++) {
            if (init == -1) {
                part = (sc_biguint<16>)rand();
            }
            else {
                part = (sc_biguint<16>)init;
            }

            if (i == 0) {
                result = part;
            }
            else {
                result = result | (part << (16*i));
            }
        }
        return result;
    }

<continue to next page>
```

2 threads

mem_r1w1_2clk.h (3/5)

```
    void thread_read() {
        rd1.write(0);
        for (int i = 0; i < T_LAT; i++) {
            reg_ra[i].write(0);
            reg_re[i].write(!T_ACTR);
        }
        wait();
        while (1) {
            if (T_LAT == 1) {
                if (T_WORD != 0) {
                    if (ra1.read() >= T_WORD) {
                        cout << "[Error @" << sc_time_stamp()
                             << "] Read access over size: "
                             << name() << "'s address = " << ra1.read() << endl;
                        sc_stop();
                        wait();
                    }
                }
                if (re1.read() == T_ACTR) {
                    rd1.write(ptr_mem[(int)ra1.read()]);
                }
            }
            else {
                reg_ra[T_LAT-2].write(ra1.read());
                reg_re[T_LAT-2].write(re1.read());

                for (int i = 0; i < T_LAT-2; i++) {
                    reg_ra[i].write(reg_ra[i+1].read());
                    reg_re[i].write(reg_re[i+1].read());
                }

                if (T_WORD != 0) {
                    if (reg_ra[0].read() >= T_WORD) {
                        cout << "[Error @" << sc_time_stamp()
                             << "] Read access over size: "
                             << name() << "'s address = " << reg_ra[0].read() << endl;
                        sc_stop();
                        wait();
                    }
                }

                if (reg_re[0].read() == T_ACTR) {
                    rd1.write(ptr_mem[(int)reg_ra[0].read()]);
                }
            }
            wait();
        }
    }

<continue to next page>
```

Read access thread

If you don’t expect to store previous value of read data, please add the following code after while(1)
rd1.write(get_init(1, -1));

When latency is 1

Out of bounds check for number of words

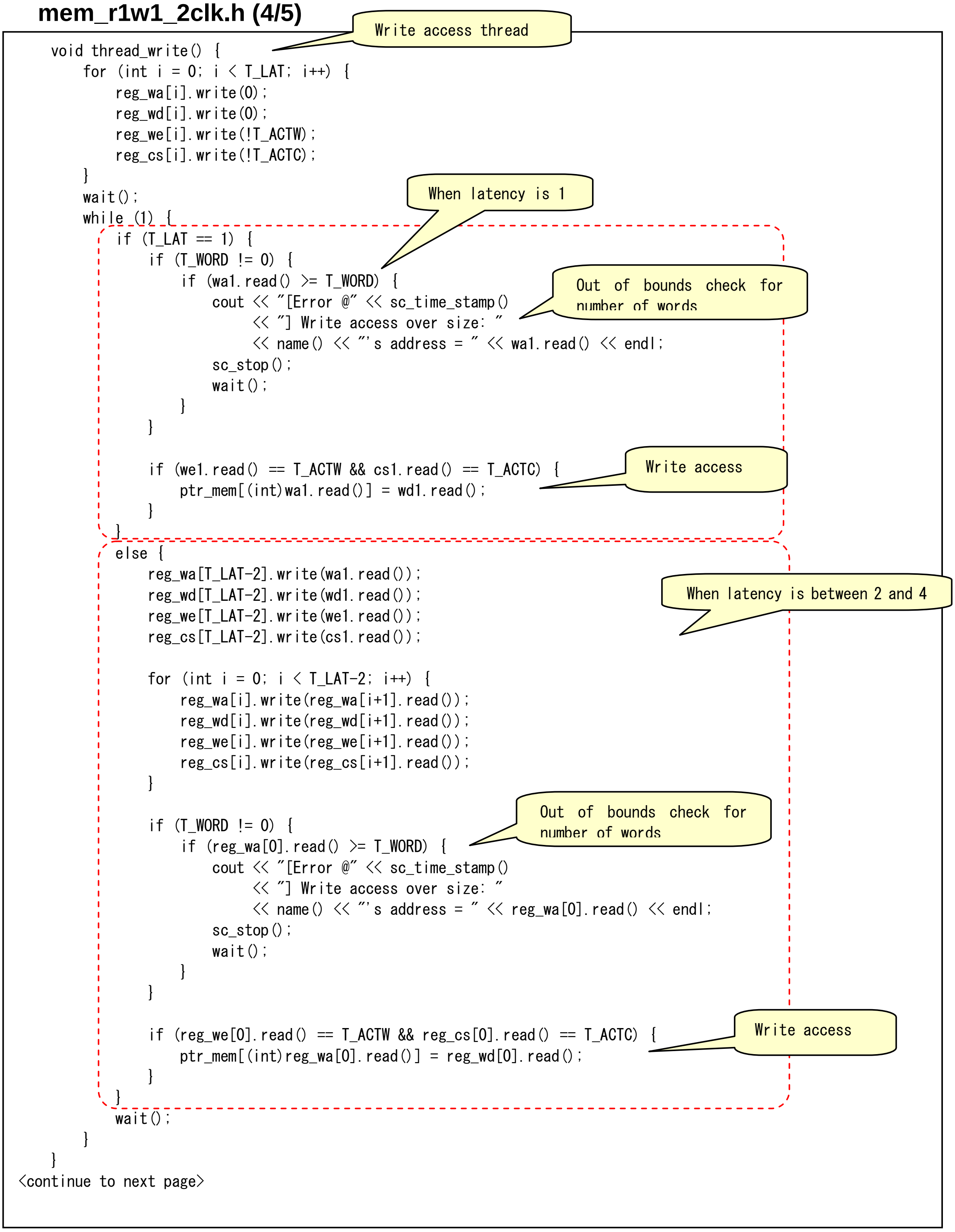
Read access

When latency is between 2 and 4

Out of bounds check for number of words

Read access

mem_r1w1_2clk.h (4/5)



```
  void thread_write() {
      for (int i = 0; i < T_LAT; i++) {
          reg_wa[i].write(0);
          reg_wd[i].write(0);
          reg_we[i].write(!T_ACTW);
          reg_cs[i].write(!T_ACTC);
      }
      wait();
      while (1) {
          if (T_LAT == 1) {
              if (T_WORD != 0) {
                  if (wa1.read() >= T_WORD) {
                      cout << "[Error @" << sc_time_stamp()
                           << "] Write access over size: "
                            << name() << "'s address = " << wa1.read() << endl;
                      sc_stop();
                      wait();
                  }
              }

              if (we1.read() == T_ACTW && cs1.read() == T_ACTC) {
                  ptr_mem[(int)wa1.read()] = wd1.read();
              }
          }
          else {
              reg_wa[T_LAT-2].write(wa1.read());
              reg_wd[T_LAT-2].write(wd1.read());
              reg_we[T_LAT-2].write(we1.read());
              reg_cs[T_LAT-2].write(cs1.read());

              for (int i = 0; i < T_LAT-2; i++) {
                  reg_wa[i].write(reg_wa[i+1].read());
                  reg_wd[i].write(reg_wd[i+1].read());
                  reg_we[i].write(reg_we[i+1].read());
                  reg_cs[i].write(reg_cs[i+1].read());
              }

              if (T_WORD != 0) {
                  if (reg_wa[0].read() >= T_WORD) {
                      cout << "[Error @" << sc_time_stamp()
                           << "] Write access over size: "
                           << name() << "'s address = " << reg_wa[0].read() << endl;
                      sc_stop();
                      wait();
                  }
              }

              if (reg_we[0].read() == T_ACTW && reg_cs[0].read() == T_ACTC) {
                  ptr_mem[(int)reg_wa[0].read()] = reg_wd[0].read();
              }
          }
          wait();
      }
  }
<continue to next page>
```

Write access thread
When latency is 1
Out of bounds check for number of words
Write access
When latency is between 2 and 4
Out of bounds check for number of words
Write access

mem_r1w1_2clk.h (5/E)

```
    void vcd_trace(ssgen_trace_file* tf, int depth = HIER_MAX) {
        if (tf != 0 && depth > 0) {
            std::string nm = std::string(name());
            sc_trace(tf, rclk, nm + ".rclk");
            sc_trace(tf, wclk, nm + ".wclk");
            sc_trace(tf, wa1, nm + ".wa1");
            sc_trace(tf, wd1, nm + ".wd1");
            sc_trace(tf, we1, nm + ".we1");
            sc_trace(tf, cs1, nm + ".cs1");
            sc_trace(tf, ra1, nm + ".ra1");
            sc_trace(tf, rd1, nm + ".rd1");
            sc_trace(tf, re1, nm + ".re1");
        }
    }
};

#endif // MEM_R1W1_2CLK_H
```

Description of wave dump for all memory ports

8.2.4 Simulation execution script

Here shows the simulation execution script generated with the specification of command line option "-osci", "- vcs", and "- ies" when ssgen is executed. Makefile and Makefile.defs are generated by specifying “-osci”. run_vcs.sh, vcs_lsfsh_sc, vcs_lsfsh_rtl, test_top_vcs.map and vpd.ucli are generated by specifying “-vcs”. And run_ies.sh, ncverilog_lsfsh_sc, ncverilog_lsfsh_rtl, test_top_ies.map and probe.tcl are generated by specifying "- ies".

･OSCI-Sim

Execution method

%s> make

%s> run.exe

Makefile

```
##====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##====================================================
TARGET_ARCH = linux
CC     = g++
OPT    = -m32 -Wall
## please add your include header path to USRDIR, if any
USRDIR =  -I/common/appl/Renesas/SystemC/utility/ssgen
MACRO  = -D_DEBUG_SIM -D_OSCI -D_MEM_MODEL
#GCOV   = -fprofile-arcs -ftest-coverage
CFLAGS = $(OPT) $(MACRO) $(GCOV)

MODULE = run
SRCS := $(wildcard *.cpp)

include ./Makefile.defs
```

Please set the include path when there is an include file needed when compiling

Makefile.defs

```
##====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##====================================================
SYSTEMC = /common/appl/Renesas/SystemC/SystemC-2.2

INCDIR = -I. -I$(SYSTEMC)/include
LIBDIR = -L. -L$(SYSTEMC)/lib-$(TARGET_ARCH)

LIBS   = -lm $(EXTRA_LIBS) -lsystemc
OBJDIR = obj

OBJS := $(SRCS:.cpp=.o)
OBJS := $(addprefix obj/, $(notdir $(OBJS)))

EXE    = $(MODULE).exe

.SUFFIXES: .cpp .o .x .exe

default : $(OBJDIR) $(EXE)

$(OBJDIR):
          @mkdir -p $@

$(EXE): $(OBJS)
          $(CC) $(CFLAGS) $(INCDIR) $(USRDIR) $(LIBDIR) -o $@ $(OBJS) $(LIBS) 2>&1 | c++filt

$(OBJDIR)/%.o: %.cpp
          $(CC) $(CFLAGS) $(INCDIR) $(USRDIR) -c $< -o $@

clean::
          rm -f $(OBJS) *~ $(EXE) core

ultraclean: clean
          rm -f Makefile.deps

Makefile.deps:
          $(CC) $(CFLAGS) $(INCDIR) $(USRDIR) -M $(SRCS) >> Makefile.deps

-include Makefile.deps
```

Although there is no necessity for modify this file basically, if you need to change the path of SystemC package or g++ version(default is v4.1.2), please modify.

Use SystemC 2.2 of REL-EWS by default
It is possible to change path of SystemC by env_systemc command.

･VCS-MX

Execution method

```
%s> ln -s vcs_lsfsh_sc vcs_lsfsh    // When SystemC model simulation
%s> ln  -s vcs_lsfsh_rtl vcs_lsfsh    // When synthesized RTL simulation
%s> run_vcs.sh
```

run_vcs.sh

```
#!/bin/csh -f
##======================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##======================================================
source /common/appl/dotfiles/vcs_mx.CSHRC_2011.12-sp1-1
bs -M 500 -os RHEL5 vcs_lsfsh
```

Use VCS-MX 2011.12 by default
It is possible to change VCS-MX setting by env_vcs command.

It is possible to customize memory size and OS name of bs command by "-vcs" option
(In default, memory size is 500 and OS name is RHEL5)

vcs_lsfsh_sc (for SystemC Simulation)

```
#!/bin/csh -f
##======================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##======================================================
set VG_GNU_PACKAGE = /common/appl/Synopsys/vg_gnu_package/2011.12/linux
source ${VG_GNU_PACKAGE}/source_me_gcc4_32.csh

rm -rf AN.DB DVEfiles csrc simv.daidir simv.vdb simv .X_dummy2.v

## please add your include header path to "-I", if any
## please add sub module file, if any
syscan -sysc=2.2 -cpp g++ -cc gcc ¥
  -cflags "-I/common/appl/Renesas/SystemC/utility/ssgen -D_MEM_MODEL -D_DEBUG_SIM" ¥
#  -cflags "-fprofile-arcs -ftest-coverage" ¥
  *.cpp

vcs -sysc=2.2 -timescale=1ns/1ps -cpp g++ -cc gcc ¥
  -ldflags "" ¥
#  -ldflags "-fprofile-arcs -ftest-coverage" ¥
#  -debug_pp

simv ¥
# -ucli -ucli2Proc -do vpd.ucli -systemcrun arg

#gcov -o csrc/sysc XXX.cpp > XXX_gcov.log
```

Use g++ v4.2.2 by default
It is possible to change g++ setting by env_vcs_gcc command.

Please set the include path when there is an include file needed when compiling

Necessary for gcov code coverage

Necessary for VPD file dump

When you get VPD file dump, "-ucli -ucli2Proc -do vpd.ucli" should be specified. However, the only port/signal whose name is initialized in constructor is correctly displayed by DVE
When you want to give command argument(ex. -vcd 1), please specify it after "-systemcrun"

Use for getting report of gcov code coverage

vcs_lsfsh_rtl (for synthesized RTL simulation)

```
#!/bin/csh -f
##======================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##======================================================
set VG_GNU_PACKAGE = /common/appl/Synopsys/vg_gnu_package/2011.12/linux
source ${VG_GNU_PACKAGE}/source_me_gcc4_32.csh

rm -rf AN.DB DVEfiles csrc simv.daidir simv.vdb simv .X_dummy2.v
if ( -e test_top.h ) mv test_top.h test_top.h.t
if ( -e test_top.cpp ) mv test_top.cpp test_top.cpp.t

vlogan -sysc=2.2 +v2k test_top.v -sc_model test_top -sc_portmap test_top_vcs.map
## please add sub module file, if any
vlogan +v2k test1.v test2.v
## please add your include header path to "-I", if any
syscan -sysc=2.2 -cpp g++ -cc gcc ¥
  -cflags "-I/common/appl/Renesas/SystemC/utility/ssgen -D_MEM_
  *.cpp

vcs -sysc=2.2 -timescale=1ns/1ps -cpp g++ -cc gcc ¥
  +warn=noSC-TCMM-V5 ¥
  -ldflags "" ¥
#  -debug_pp

simv ¥
#  -ucli -ucli2Proc -do vpd.ucli -systemcrun arg

if ( -e test_top.h.t ) mv test_top.h.t test_top.h
if ( -e test_top.cpp.t ) mv test_top.cpp.t test_top.cpp
```

Use g++ v4.2.2 by default
It is possible to change g++ setting by env_vcs_gcc command.

Specify map file of CoSim

Modules under one from top hierarchy (test1.v,test2.v) are included as compile target at first. If there also exist modules lower than them, please add the files manually.

Necessary for VPD file dump

When you get VPD file dump, "-ucli -ucli2Proc -do vpd.ucli" should be specified.
When you want to give command argument, please specify it after "-systemcrun"

test_top_vcs.map (Map file for CoSim)

```
##======================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##======================================================
clk1  1  bit  sc_clock
clk2  1  bit  sc_clock
rst_n  1  bit  bool
inport  8  bitvector  sc_uint
tap_out  8  bitvector  sc_uint
outport  8  bitvector  sc_uint
ram1_ra1  7  bitvector  sc_uint
ram1_rd1  8  bitvector  sc_uint
ram1_re1  1  bit  bool
ram1_wa1  7  bitvector  sc_uint
ram1_wd1  8  bitvector  sc_uint
ram1_we1  1  bit  bool
ram1_cs1  1  bit  bool
```

This file is necessary for doing CoSim with synthesized RTL
File name is *modulename*_vcs.map

vpd.ucli

```
##======================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##======================================================
synopsys::scope .
set filename dump.vpd
set fid [synopsys::dump -file $filename -type VPD]
synopsys::dump -add "." -depth 0 -fid $fid
puts "==== Waveform file generation is enabled ( $filename ) ===="
synopsys::run
```

This file is necessary for VPD file dump
Dump file name is dump.vpd

·IES

Execution method

%s> ln -s ncverilog_lsfsh_sc ncverilog_lsfsh   // When SystemC model simulation

%s> ln –s ncverilog_lsfsh_rtl ncverilog_lsfsh   // When synthesized RTL simulation

%s> run_ies.sh

run_ies.sh

```
#!/bin/csh -f
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
source /common/appl/dotfiles/cadence.CSHRC_ius12.10s004
bs -M 500 -os RHEL5 ncverilog_lsfsh
```

Use IES 12.10 by default
It is possible to change IES setting by env_ies command.

It is possible to customize memory size and OS name of bs command by ″-ies″ option
(In default, memory size is 500 and OS name is RHEL5)

ncverilog_lsfsh_sc (for SystemC simulation)

```
#!/bin/csh -f
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
rm -rf INCA_libs irun.log ncsc.log wave.shm

## please add your include header path to ″-I″, if any
## please add sub module file, if any
irun ¥
  *.cpp ¥
  -sysc ¥
  -sctop sc_main ¥
  -I/common/appl/Renesas/SystemC/utility/ssgen ¥
  -I. ¥
  -D_DEBUG_SIM ¥
  -D_MEM_MODEL ¥
#  -Wcxx,-fprofile-arcs,-ftest-coverage ¥
#  -Wld,-fprofile-arcs,-ftest-coverage ¥
#  +systemc_args+″″ ¥
#  -access r -input probe.tcl


#gcov -o INCA_libs/irun.nc/ncsc_run/ncsc_obj XXX.cpp > XXX_gcov.log
```

Please set the include path when there is an include file needed when compiling

Necessary for gcov code coverage

Use for giving command argument(ex. -vcd 1)

Use for making wave dump(SHM database)
However, the only port/signal whose name is initialized in constructor is correctly displayed by simvision

Use for getting report of gcov code coverage

ncverilog_lsfsh_rtl (for synthesized RTL simulation)

```
#!/bin/csh -f
##====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##====================================================
rm -rf INCA_libs irun.log ncsc.log wave.shm
if ( -e test_top.h ) mv test_top.h test_top.h.t
if ( -e test_top.cpp ) mv test_top.cpp test_top

## please add sub module file, if any
ncverilog -c test_top.v test1.v test2.v

ncshell -NOCOMPILE ¥
  -import verilog -into systemc test_top -file test_top_ies.map

## please add your include header path to "-I", if any
irun ¥
  *.cpp ¥
  -sysc ¥
  -sctop sc_main ¥
  -timescale 1ps/1ps ¥
  -I/common/appl/Renesas/SystemC/utility/ssgen ¥
  -D_MODE_RTL ¥
  -D_MEM_MODEL ¥
#  +systemc_args+"" ¥
#  -access r -input probe.tcl

if ( -e test_top.h.t ) mv test_top.h.t test_top.h
if ( -e test_top.cpp.t ) mv test_top.cpp.t test_top.cpp
```

Modules under one from top hierarchy (test1.v,test2.v) are included as compile target at first. If there also exist modules lower than them. please add the files manually.

Use for giving command argument(ex. -vcd 1)

Use for making wave dump(SHM database)

test_top_ies.map (Map file for CoSim)

```
##====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##====================================================
-sctype "clk1:bool"
-sctype "clk2:bool"
-sctype "rst_n:bool"
-sctype "inport:sc_uint<8>"
-sctype "tap_out:sc_uint<8>"
-sctype "outport:sc_uint<8>"
-sctype "ram1_ra1:sc_uint<7>"
-sctype "ram1_rd1:sc_uint<8>"
-sctype "ram1_re1:bool"
-sctype "ram1_wa1:sc_uint<7>"
-sctype "ram1_wd1:sc_uint<8>"
-sctype "ram1_we1:bool"
-sctype "ram1_cs1:bool"
```

This file is necessary for doing CoSim with synthesized RTL
File name is *modulename*_ies.map

probe.tcl

```
##====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##====================================================
database -open waves -default
probe -create -all -depth all
run
exit
```

This file is necessary for making wave dump with IES

8.2.5 CtoS script

Here shows CtoS script (run_ctos_test_top.sh, ctos_test_top.tcl) generated from hierarchy definition file test_top.in with the specification of command line option "-ctos" when ssgen is executed.

Method of executing CtoS

%s> run_ctos_test_top.sh

run_ctos_test_top.sh

```
#!/bin/csh -f
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
source /ssv/3md/HYOKA/SETSUGI/dotfile/cadence.CSHRC_ctos_v12.20-s201
bs -M 500 -os RHEL5 ctos ctos_test_top.tcl -log ctos_test_top.log
```

Use CtoS v12.2 by default
It is possible to change CtoS setting file by env_ctos command.

ctos_test_top.tcl

```
##=====================================================
##  ssgen v1.5 (Synthesizable SystemC code Generator)
##  Renesas Group Confidential
##=====================================================
# parameters
set PERIOD 5000

# set variables
set NAME test_top
set MODULE /designs/$NAME/modules/$NAME
set ARRAY $MODULE/arrays

# preparation
new_design $NAME
set_attr source_files "test_top.cpp" [get_design]
set_attr compile_flags " -w -D_MEM_MODEL -I/common/appl/Renesas/SystemC/utility/ssgen -D_CTOS_TOP" [get_design]
set_attr top_module_path $NAME [get_design]
set_attr verilog_rtl_model_suffix "" [get_design]
set_attr auto_write_models false [get_design]
set_attr low_power_clock_gating true [get_design]
define_clock -name clk1 -period $PERIOD
define_clock -name clk2 -period $PERIOD
build

<omitted>

# write files
write_rtl -o ${NAME}.v $MODULE
exit
```

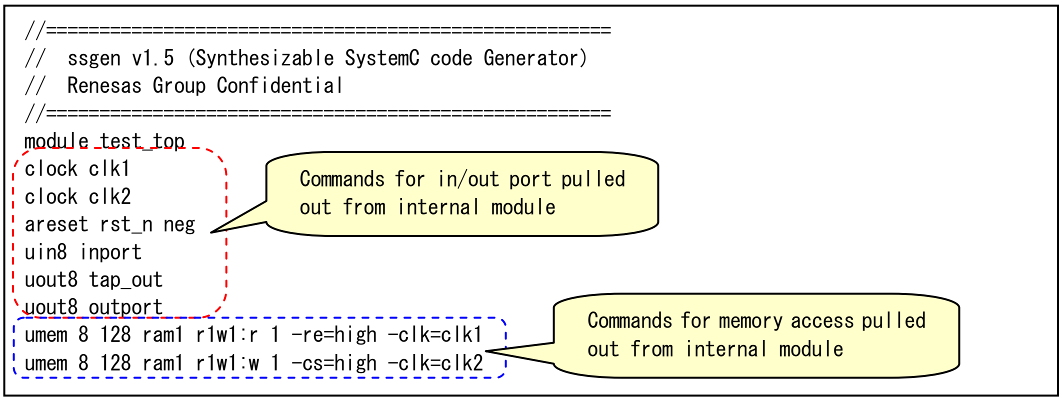
Set _CTOS_TOP macro
(Disable process registration of test1 and test2)

Generate RTL description

8.2.6 module definition file

Here shows module definition file (mod_test_top.in) equivalent to test_top module generated from hierarchy definition file test_top.in. Please specify this file with sub command in the hierarchy definition file equivalent to upper hierarchy module which has test_top module as internal module.

mod_test_top.in



```
//====================================================
//  ssgen v1.5 (Synthesizable SystemC code Generator)
//  Renesas Group Confidential
//====================================================
module test_top
clock clk1
clock clk2
areset rst_n neg
uin8 inport
uout8 tap_out
uout8 outport
umem 8 128 ram1 r1w1:r 1 -re=high -clk=clk1
umem 8 128 ram1 r1w1:w 1 -cs=high -clk=clk2
```

Commands for in/out port pulled out from internal module

Commands for memory access pulled out from internal module

The above module definition file has not cthread command and is assumed to be used only for hierarchy generation mode (with sub command). So, ssgen detected the following error if you input this file to ssgen.

```
[E110:mod_test_top.in] no SC_CTHREAD is defined.
aborted parsing here.
```

9. Macro function for memory access

This chapter picks up some macro functions for memory access and shows the internal implementation. If you would like to confirm all implementation completely, please refer ssgen library file "ssgenlib.h".

・Memory pointer

MEM_PTR[_A]

```
//-- for testbench
#define MEM_PTR_A(NAME， DTYPE) ¥
extern DTYPE *ptr_##NAME；

//-- for model
#ifndef _MEM_MODEL
#define MEM_PTR(NAME， DTYPE) ¥
extern DTYPE *ptr_##NAME；
#else
#define MEM_PTR(NAME， DTYPE)
#endif
```

Declare extern of pointer when _MEM_MODEL macro is not specified

・Define port and array

MEM_DEF_{1|2R|2W|1R|1W}[_E][_P]

The case of MEM_DEF_1_E

```
#ifdef _MEM_MODEL
#define MEM_DEF_1_E(NAME， DTYPE， AWID， LAT， IPRE， OPRE， AD1， WD1， WE1， RD1， CS1) ¥
    sc_out < sc_uint<AWID> > OPRE ## NAME ## _ ## AD1； ¥
    sc_out < DTYPE > OPRE ## NAME ## _ ## WD1； ¥
    sc_out < bool > OPRE ## NAME ## _ ## WE1； ¥
    sc_in < DTYPE > IPRE ##NAME ## _ ## RD1； ¥
    sc_out < bool > OPRE ## NAME ## _ ## CS1；
#else
#define MEM_DEF_1_E(NAME， DTYPE， AWID， LAT， IPRE， OPRE， AD1， WD1， WE1， RD1， CS1) ¥
    sc_signal < sc_uint<AWID> > addr_ ## NAME[LAT+1]； ¥
    sc_signal < DTYPE > data_ ## NAME[LAT+1]； ¥
    sc_signal < bool > rw_ ## NAME[LAT+1]； // 0：R， 1：W
#endif
```

Declare memory ports when _MEM_MODEL macro is specified

Declare registers for pipeline access of memory array when _MEM_MODEL macro is not specified

・Initialize port name

MEM_ININM_{1|2R|2W|1R|1W}[_E][_P]

The case of MEM_ININM_1_E

```
#define CTOR_NM(nm)  nm(#nm)
#ifdef _MEM_MODEL
#define MEM_ININM_1_E(NAME, IPRE, OPRE, AD1, WD1, WE1, RD1, CS1) ¥
    ,CTOR_NM(OPRE ## NAME ## _ ## AD1) ¥
    ,CTOR_NM(OPRE ## NAME ## _ ## WD1) ¥
    ,CTOR_NM(OPRE ## NAME ## _ ## WE1) ¥
    ,CTOR_NM(IPRE ## NAME ## _ ## RD1) ¥
    ,CTOR_NM(OPRE ## NAME ## _ ## CS1)
#else
#define MEM_ININM_1_E(NAME, PRE, WESUF, CSSUF)
#endif
```

Initialize port name by constructor initializer when _MEM_MODEL macro is specified

・Initialize value

MEM_INIVAL_{1|2R|2W|1R|1W}[_E][_NOWD|_NOAD|_NOAW][_P]

The case of MEM_INIVAL_1_E

```
#ifdef _MEM_MODEL
#define MEM_INIVAL_1_E(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    OPRE ## NAME ## _ ## AD1 .write(0); ¥
    OPRE ## NAME ## _ ## WD1 .write(0); ¥
    OPRE ## NAME ## _ ## WE1 .write(WEV); ¥
    OPRE ## NAME ## _ ## CS1 .write(CSV);
#else
#define MEM_INIVAL_1_E(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    addr_ ## NAME[0].write(0); ¥
    data_ ## NAME[0].write(0); ¥
    rw_ ## NAME[0].write(0);
#endif
```

Initialize port value when _MEM_MODEL macro is specified

Initialize pipeline register value when _MEM_MODEL macro is specified

The case of MEM_INIVAL_1_E_NOWD

```
#ifdef _MEM_MODEL
#define MEM_INIVAL_1_E_NOWD(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    OPRE ## NAME ## _ ## AD1 .write(0); ¥
    OPRE ## NAME ## _ ## WE1 .write(WEV); ¥
    OPRE ## NAME ## _ ## CS1 .write(CSV);
#else
#define MEM_INIVAL_1_E_NOWD(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    addr_ ## NAME[0].write(0); ¥
    data_ ## NAME[0].write(0); ¥
    rw_ ## NAME[0].write(0);
#endif
```

No initialization of write data port (WD1)

・Negate enable signal

MEM_NEG_{1|2R|2W|1R|1W}[_E][_P]

The case of MEM_NEG_1_E

```
#ifdef _MEM_MODEL
#define MEM_NEG_1_E(NAME, OPRE, WEV, CSV, WE1, CS1) ¥
    OPRE ## NAME ## _ ## WE1 .write(WEV); ¥
    OPRE ## NAME ## _ ## CS1 .write(CSV);
#else
#define MEM_NEG_1_E(NAME, OPRE, WEV, CSV, WE1, CS1) ¥
  rw_ ## NAME[0].write(0);
#endif
```

Negate enable signal when _MEM_MODEL macro is specified

・Register and Define function for pipeline access to array

MEM_PIPE[_2R]_CTOR

MEM_PIPE[_2R]

The case of MEM_PIPE_CTOR・MEM_PIPE

```
#ifndef _MEM_MODEL
#define MEM_PIPE_CTOR(NAME, CLK) ¥
    SC_METHOD(method_ ## NAME ## _pipe); ¥
    sensitive << (CLK).pos();
#define MEM_PIPE(NAME, LAT) ¥
  void method_ ## NAME ## _pipe() { ¥
    if (rw_ ## NAME[LAT].read() == 1) { ¥
      ptr_ ## NAME[(int)addr_ ## NAME[LAT].read()] = data_ ## NAME[LAT].read(); ¥
    } ¥
    for (int i = 0; i < LAT; i++) { ¥
      addr_ ## NAME[i+1].write(addr_ ## NAME[i].read()); ¥
      data_ ## NAME[i+1].write(data_ ## NAME[i].read()); ¥
      rw_ ## NAME[i+1].write(rw_ ## NAME[i].read()); ¥
    } ¥
  }
#else
#define MEM_PIPE_CTOR(NAME, CLK)
#define MEM_PIPE(NAME, LAT)
#endif
```

Register to SC_METHOD and define function when _MEM_MODEL macro is not defined

Write data to memory array based on latency

Shift registers

・Read access (Request and Data)

MEM_REQ_{1|2|1R}[_E][_NOAD][_P]

MEM_RD_{1|2|1R}[_P]

The case of MEM_REQ_1_E・MEM_RD_1

```
#ifdef _MEM_MODEL
#define MEM_REQ_1_E(NAME, OPRE, CSV, AD1, CS1) ¥
    OPRE ## NAME ## _ ## AD1 .write(addr); ¥
    OPRE ## NAME ## _ ## CS1 .write(CSV);
#define MEM_RD_1(NAME, LAT, IPRE, RD1) ¥
    data = IPRE ## NAME ## _ ## RD1 .read();
#else
#define MEM_REQ_1_E(NAME, OPRE, CSV, AD1, CS1) ¥
    addr_ ## NAME[0].write(addr); ¥
    rw_ ## NAME[0].write(0);
#define MEM_RD_1(NAME, LAT, IPRE, RD1) ¥
    data = ptr_ ## NAME[(int)addr_ ## NAME[LAT].read()];
#endif
```

Port access when _MEM_MODEL macro is specified

Array access when _MEM_MODEL macro is not specified

- **Write access**

MEM_WR_{1|2}[_E][_NOWD|_NOAD|_NOAW][_P]

The case of MEM_WR_1_E

```
#ifdef _MEM_MODEL
#define MEM_WR_1_E(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    OPRE ## NAME ## _ ## AD1 .write(addr); ¥
    OPRE ## NAME ## _ ## WD1 .write(data); ¥
    OPRE ## NAME ## _ ## WE1 .write(WEV); ¥
    OPRE ## NAME ## _ ## CS1 .write(CSV);
#else
#define MEM_WR_1_E(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    addr_ ## NAME[0].write(addr); ¥
    data_ ## NAME[0].write(data); ¥
    rw_ ## NAME[0].write(1);
#endif
```

Port access when _MEM_MODEL macro is specified

Array access when _MEM_MODEL macro is not specified

The case of MEM_WR_1_E_NOWD

```
#ifdef _MEM_MODEL
#define MEM_WR_1_E_NOWD(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    OPRE ## NAME ## _ ## AD1 .write(addr); ¥
    OPRE ## NAME ## _ ## WE1 .write(WEV); ¥
    OPRE ## NAME ## _ ## CS1 .write(CSV);
#else
#define MEM_WR_1_E_NOWD(NAME, OPRE, WEV, CSV, AD1, WD1, WE1, CS1) ¥
    addr_ ## NAME[0].write(addr); ¥
    data_ ## NAME[0].write(data); ¥
    rw_ ## NAME[0].write(1);
#endif
```

No assignment of write data port (WD1)

10. Output Message

This chapter shows output message list of ssgen. The head character of the message number in "No." column indicates the message level. The declared meaning is as follows.

- E: Error (Abort execution)
- W: Warning (Continue execution)
- I: Information (Continue execution)

In "Output condition" column, the bold string means command name of ssgen.

No.	Message	Category	Output condition
E001	*"command"* should be set before "module"	Illegal com order	**changelog, style_module, `include, `define, #include, #define, env_systemc, env_vcs, env_ies, env_ctos, env_vcs_gcc, space_indent, mem_suffix, style_alloc, vcd_trace** is specified after **module** in module definition file
E002	"*command*" should be set after "module"	Illegal com order	・Condition1 When **clock** is not specified in **`include** file, **clock** is specified before **module** in module definition file ・Condition2 When **clock** is specified in **`include** file, **areset, sreset, soft_reset, {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem, method, func, free area, `ifdef/`else/`endif, #ifdef/#endif, cthread** is specified before **module** in module definition file
E003	"*command*" should be set after "clock"	Illegal com order	**areset, sreset, soft_reset, {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem, method, func, free area, `ifdef/`else/`endif, #ifdef/#endif, cthread** is specified before **clock** in module definition file
E006	"*command*" should be set before "top"	Illegal com order	**changelog, style_module, env_systemc, env_vcs, env_ies, env_ctos, env_vcs_gcc, space_indent, mem_suffix, style_alloc, vcd_trace, `include** is specified after **top** in hierarchy definition file
E007	sub should be set after "top"	Illegal com order	**sub** is specified before **top** in hierarchy definition file
E008	"*command*" should be set after "sub"	Illegal com order	**bind, tap, `ifdef** is specified before 1st **sub** in hierarchy definition file
E009	not support nesting with "`ifdef"	Illegal com order	**`ifdef/#ifdef** is described inside **`ifdef**

No.	Message	Category	Output condition
E010	not support nesting with "#ifdef"	Illegal com order	**`ifdef/#ifdef** is described inside **#ifdef**
E011	"*command*" should be set after "`ifdef"	Illegal com order	**`else/`endif** is specified before **`ifdef**
E012	"`endif" should be set	Illegal com order	**`endif** is not specified
E013	"*command*" should not be set in "TESTBENCH"	Illegal com order	A command except **{u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, func, free area** is specified inside TESTBENCH macro
E014	"#endif" should be set after "#ifdef"	Illegal com order	**#endif** is specified before **#ifdef**
E015	"#endif" should be set	Illegal com order	**#endif** is not specified
E016	"*command*" should not be set in "#ifdef macro"	Illegal com order	A command except **{u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, func, free area** is specified inside _DEBUG* macro
E017	"!--" should not be set with another word	Illegal com order	**"!--"** and **"--!"** are specified in same line
E018	"--!" should be set after "!--"	Illegal com order	**"--!"** is specified before **"!--"**
E019	"--!" should be set	Illegal com order	**"--!"** is not specified
E020	"*command*" should be set in "TESTBENCH"	Illegal com order	**{u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, func, free area** is specified outside TESTBENCH macro in hierarchy definition file
E021	"*command*" should not be set in "*macroname*" macro	Illegal com order	A command except **areset, sreset, soft_reset, {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char, [u]short, [u]int, {u|s}evN, {u|s}mem, method, func, free area** is specified inside **`ifdef** macro (except **`ifdef** TESTBENCH)
E022	sub should be set before "bind/tap/`ifdef"	Illegal com order	2nd **sub** is specified after **bind/tap/`ifdef** in hierarchy definition file
E023	“*command*” should not be set in “_SLEC_BBOX”	Illegal com order	A command except **{u|s}inN, {u|s}outN** is specified inside _SLEC_BBOX macro
E101	"*command*" multiple settings [*file* : *line*]	Illegal com num	**style_module, module, top, env_systemc, env_vcs, env_ies, env_ctos, env_vcs_gcc, space_indent, mem_suffix, style_alloc, vcd_trace** is specified multi times
E102	not support more than 1 level nesting	Illegal com num	**`include** is specified in **`include** file or **–include** file
E103	"*command*" should not be set in include file	Illegal com num	**module, cthread, top, sub, tap, bind** is specified in **`include** file
E104	no clock is defined	Illegal com num	**clock** is not specified
E105	not support multiple clock	Illegal com num	**clock** is specified multi times
E106	areset should be defined only once	Illegal com num	**areset** is specified multi times
E107	only soft_reset is defined as reset	Illegal com num	only **soft_reset** is specified as reset

No.	Message	Category	Output condition
E110	no SC_CTHREAD is defined	Illegal comnum	**cthread** is not specified
E112	"else" should not be set to "`ifdef TESTBENCH"	Illegal comnum	**`else** is specified for **`ifdef** TESTBENCH
E113	no module is defined	Illegal comnum	**module** is not specified
E114	sub "*subname*" in #ifdef must not have output port	Illegal comnum	Module which is specified in **sub** command in **#ifdef** area has output ports
E201	Invalid command "*command*"	Illegal comformat	Invalid command name is used

No.	Message	Category	Output condition
E202	"*command*" has invalid format	Illegal com format	**changelog** has no argument **style_module** has no argument or 2 or more arguments **`include** has no argument or 2 or more arguments **`define** has no argument **#define** has no arguments **#include** has no argument or 2 or more arguments **env_systemc** has no argument or 2 or more arguments **env_vcs** has no argument or 2 or more arguments **env_ies** has no argument or 2 or more arguments **env_ctos** has no argument or 2 or more arguments **env_vcs_gcc** has no argument or 2 or more arguments **space_indent** has no argument or 2 or more arguments **mem_suffix** has no argument or 2 or more arguments **style_alloc** has no argument or 2 or more arguments **vcd_trace** has no argument or 2 or more arguments **module** has no argument or 2 or more arguments **clock** has no argument or 2 or more arguments **areset/sreset** has no or 1 argument or 3 or more arguments **soft_reset** has no or 1 argument or 4 or more arguments **{u|s}inN** has no argument ore 2 or more arguments **{u|s}outN** has no argument **{u|s}regN** has no argument **{u|s}varN** has no argument **[u]char/[u]short/[u]int** has not argument **{u|s}evN** has no argument **`ifdef** has no argument or 2 or more arguments **`else** has argument **`endif** has argument **#ifdef** has no argument ore 2 or more arguments **#endif** has argument **{u|s}mem** has not right argument order "data width size name type latency" **cthread** has no argument **method** has no or 1 argument **func** has not right format "returntype funcname ()" **top** has no argument or 2 or more arguments **sub** has no or 1 argument **bind** has no argument or 1 or 4 or more arguments **tap** has no argument or 3 or more arguments
E203	width should be more than 0	Illegal com format	bit width of **{u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, {u|s}evN** is 0
E205	const should not be set to "*command*"	Illegal com format	**const** is specified with a command except **{u|s}varN, [u]char/[u]short/[u]int**
E206	option (*option*) is invalid	Illegal com format	invalid option is specified to **{u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem, cthread**

<table>
<tr><th>No.</th><th>Message</th><th>Category</th><th>Output condition</th></tr>
<tr><td>E207</td><td>option (<i>option</i>) is set again</td><td>Illegal com format</td><td>same option is specified to <b>{u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem, cthread</b> multi times</td></tr>
<tr><td>E208</td><td>should not set prefix option and iprefix/oprefix at the same time</td><td>Illegal com format</td><td>prefix option and iprefix/oprefix option are specified to <b>{u|s}mem</b> at the same time</td></tr>
<tr><td>E209</td><td>“<i>macro</i>” is not defined by `define command</td><td>Illegal com format</td><td>undefined macro is specified to command parameters of <b>{u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem and func</b></td></tr>
<tr><td>E210</td><td>It is impossible to calculate constant value in macroname “<i>macro</i>”</td><td>Illegal com format</td><td>constant value cannot be calculated from the contents of <b>`define</b> command.</td></tr>
<tr><td>E301</td><td>set sc or c++ to style_module
set static or dynamic to style_alloc</td><td>Illegal com argument</td><td>string except <b>sc/SC/c++/C++</b> is specified to <b>style_module</b>, or string except <b>static/dynamic</b> is specified to <b>style_alloc</b></td></tr>
<tr><td>E303</td><td>edge type should be "pos" or "neg"</td><td>Illegal com argument</td><td>string except <b>pos/neg</b> is specified to edge of <b>areset/sreset/soft_reset</b></td></tr>
<tr><td>E304</td><td>array element should be more than 1</td><td>Illegal com argument</td><td>array num of <b>{u|s}in, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int</b> is 1 or less</td></tr>
<tr><td>E305</td><td>not support 4D array or more array</td><td>Illegal com argument</td><td>array dimension of <b>{u|s}in, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN</b> is 4 or more</td></tr>
<tr><td>E306</td><td>init value has invalid format</td><td>Illegal com argument</td><td>Initial value of <b>{u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN</b> is invalid
(check only whether a string of initial value has “=”)</td></tr>
<tr><td>E307</td><td>no initialize ("n") should not be set when you set "const"</td><td>Illegal com argument</td><td>a command with <b>const</b> is specified with ‘n’ for initial value</td></tr>
<tr><td>E308</td><td>please use only "_DEBUG*" or “_SLEC_BBOX” for macro</td><td>Illegal com argument</td><td>string except _DEBUG* and _SLEC_BBOX is specified to <b>#ifdef</b></td></tr>
<tr><td>E309</td><td>memory width should be more than 1</td><td>Illegal com argument</td><td>bit width of <b>{u|s}mem</b> is 1 or less</td></tr>
<tr><td>E310</td><td>memory size should be more than 1</td><td>Illegal com argument</td><td>size of <b>{u|s}mem</b> is 1 or less</td></tr>
<tr><td>E311</td><td>memory type should be rw1|r1w1:r|r1w1:w|rw1:r|rw1:w</td><td>Illegal com argument</td><td>type of <b>{u|s}mem</b> is not rw1/r1w1:r/r1w1:w/rw1:r/rw1:w</td></tr>
<tr><td>E312</td><td>{cs|re|we|nowd|noad} should be set to memory type {rw1|r1w1:r|r1w1:w|rw1:r|rw1:w}</td><td>Illegal com argument</td><td>“re” is specified to <b>{u|s}mem</b> when type is “rw1”, ”cs/we/nowd” is specified when type is r1w1:r, ”re” is specified when type is “r1w1:w”, “cs/we/nowd”/noad is specified when type is “rw1:r”, “re/cs/nowd/noad” is specified when type is “rw1:w”</td></tr>
<tr><td>E313</td><td>memory latency should be between 1 and 4</td><td>Illegal com argument</td><td>latency of <b>{u|s}mem</b> is not between 1 and 4</td></tr>
<tr><td>E314</td><td>memory prefix is invalid</td><td>Illegal com argument</td><td>prefix of {u|s}mem is empty string or string with ‘_’ as head character, sc_(SC_), MEM_, ssgen_</td></tr>
<tr><td>E315</td><td>set high or low to active of memory enable</td><td>Illegal com argument</td><td>active level of cs/re/we of <b>{u|s}mem</b> is not “high” nor “low”</td></tr>
<tr><td>E316</td><td>"<i>sensitivity</i>" should be defined as port/signal above this line</td><td>Illegal com argument</td><td>not defined name as port/signal is specified to sensitivity of <b>method</b></td></tr>
</table>

No.	Message	Category	Output condition
E317	"*sensitivity*" is set to sensitivity list again	Illegal com argument	same port/signal is specified to sensitivity of **method** multi times
E318	not support edge trigger SC_METHOD	Illegal com argument	edge trigger is specified to sensitivity of **method**
E322	tap target "*internalsignal*" does not exist [in instance "*instancename*"]	Illegal com argument	a signal that is not used in internal module is specified to **tap** (“*instancename*” is added to message when it is specified)
E323	"*internalsignal*" is set to tap again	Illegal com argument	same internal signal is specified to **tap** multi times
E324	please use only "TESTBENCH" for macro	Illegal com argument	macro name except　TESTBENCH is specified to **`ifdef** in hierarchy definition file
E325	set signal name without array element description	Illegal com argument	name with array form is specified to **tap/bind**
E326	"*clockname*" does not exist	Illegal com argument	a clock that does not exist in module is specified to **cthread**
E327	"*resetname*" does not exist	Illegal com argument	a reset that does not exist in module is specified to **cthread**
E328	"*resetname*" is set to reset list again	Illegal com argument	same reset is specified to **cthread** multi times
E329	"*threadname*" does not exist	Illegal com argument	a thread that does not exist in module is specified to th option of **{u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/ [u]int, {u|s}evN, {u|s}mem**
E330	th option should not be set to "*command*" in TESTBENCH macro	Illegal com argument	In TESTBENCH macro, th option is specified to **{u|s}regN, {u|s}varN, [u]char/[u]short/ [u]int**
E331	bind target "*startsignal/endsignal*" does not exist in instance "*startinstence/endinstance*"	Illegal com argument	a signal that does not exist in internal module is specified as start point or end point to **bind**
E332	start point "*startsignal*" is not output port of instance "*startinstance*"	Illegal com argument	a signal that is input port in internal module is specified as start point to **bind**
E333	end point "*endsignal*" is not input port of instance "*endinstance*"	Illegal com argument	a signal that is output port in internal module is specified as end point to **bind**
E336	There is an inappropriate setting in suffix setting "*SuffixSettingFile*"	Illegal com argument	suffix configuration file specified to **{u|s}mem** or **mem_suffix** has inappropriate setting. ・a port identifier that does not exist is specified (ex. clk) ・a port identifier is specified multi times ・same suffix is specified to two or more port identifier ・specified suffix has a character other than alphanumeric character and underscore.
E337	space of indent should be between 1 and 10	Illegal com argument	0 or more than 10 is specified to **space_indent**
E338	“*threadname*” has only soft reset as reset.	Illegal com argument	asynchronous reset or synchronous reset is not specified to **cthread** with soft reset.
E339	option “*option*” should not be set in "#ifdef macro " or "`ifdef TESTBENCH"	Mismatch among com	unsupported option is specified in **#ifdef** or **`ifdef TESTBENCH**
E340	bind command should be specified with "instance.port" form	Mismatch among com	name of instantiation is not specified in start signal or end signal of **bind**

No.	Message	Category	Output condition
E341	signal of fixed port "*signalname*" should not be set to tap command	Mismatch among com	fixed port generated by **bind** is specified to **tap**
E342	not support specifying tap for memory type of rw1:r|rw1:w	Mismatch among com	rw1:1/rw1:w memory is specified to **tap**
E343	option "-debug_trace" should not set to variable whose width is more than 999	Mismatch among com	-debug_trace is specified to a variable whose width is 1000 or more than
E344	don't specify 3rd argument of bind command for debug module "*instancename*" without port fixed	Mismatch among com	3rd argument (signal name) of **bind** command is specified in **#ifdef** area
E401	not support multiple access to one memory except for R/W access	Mismatch among com	multi access to same memory except the combination of (r1w1:r & r1w1:w) or (rw1:r & rw1:w) or (rw2:a & rw2:b) is defined
E402	{2port memory|single port memory} "*name*" has incompatible setting between R/W	Mismatch among com	(r1w1:r & r1w1:w) or (rw1:r & rw1:w) to same memory have mismatch setting as follows ・sign ・data width ・size ・latency ・prefix ・ponly existence ・suffix configuration file ・initial value ・synchronize clock (only when rw1:r & rw1:w)
E403	reset signal "*name*" has incompatible setting in multiple modules	Mismatch among com	reset which connects to multiple modules has mismatch setting as follows ・synchronous / asynchronous ・active level
E404	input port "*name*" has incompatible setting in multiple modules	Mismatch among com	input port which connects to multiple modules has mismatch as follows ・sign ・bit width ・array element
E405	signal "*name*" has incompatible setting between output and input in multiple modules	Mismatch among com	connection signal among internal modules has mismatch ・sign ・bit width ・array element
E406	signal "*name*" is driven from multiple modules	Mismatch among com	there are multi same name output ports among internal modules or a signal which is a input port of internal signal is specified as end point of **bind** multi times
E407	bind target "*startsignal*" is connected to multiple signals	Mismatch among com	two or more signals have the same start point
E408	signal of fixed port "*signalname*" should not be same as the other signal	Mismatch among com	two or more fixed ports generated by **bind** have the same name
E411	macro "name" defines the different contents in multiple modules	Mismatch among com	define macro has the different contents in multiple modules
E412	soft reset "resetname" should not be set to multiple thread	Mismatch among com	soft reset is specified to th option of multiple **cthread** commands
E413	"resetname" is not set to any threads as reset signal	Mismatch among com	reset is not specified to th option of any **cthread** commands

No.	Message	Category	Output condition
E414	not support specifying “rw1:r” and “rw1:w” to one memory in a module	Mismatch among com	(rw1:r & rw1:w) is set to one memory in one module definition file.
E415	don't specify sub "*subname*" inside #ifdef that is specified also outside	Mismatch among com	Module definition file which is specified in **sub** command in **#ifdef** area is also specified in **sub** command outside **#ifdef** area
E416	port "*portname*" of "*instancename*" in #ifdef should be bound to the other port	Mismatch among com	Port of debug module cannot be bound to the any other ports which are generated in **sub** commands outside **#ifdef** area
E501	"*name*" may be identical with reserved word.	Illegal name	name of command argument is identical with reserved word(*1) ・target command of module generation mode is: **`define, #define, module, clock, areset, sreset, soft_reset, {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem, cthread, method,** name of **func** ・target command of hierarchy generation mode is: **top,** instance name of **sub**, signal name of **bind**, output name of **tap, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int,** name of **func**
E502	"*name*" violates the naming rule	Illegal name	name of command argument violates the naming rule (*2-1, *2-2) ・target command of module generation mode is: **#define**, **module, clock, areset, sreset, soft_reset, {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem, cthread, method,** name of **func** ・target command of hierarchy generation mode is: **top,** instance name of **sub,** signal name of **bind**, output port name of **tap, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int,** name of **func**

No.	Message	Category	Output condition
E601	"*name*" has been already defined [*file* : *line*]	Name overlapping	same name is used for command argument multi times ・target command of module generation mode is: **`define, #define, module, clock, areset, sreset, soft_reset, {u|s}inN, {u|s}outN, {u|s}regN, {u|s}varN, [u]char/[u]short/[u]int, {u|s}evN, {u|s}mem, cthread, method,** name of **func** however, same name among multi **func** commands does not become an error. ・target command of hierarchy generation mode is: **top**, instance name of **sub**, signal name of **bind**, output port name of **tap**, #define of internal modules, **module** of internal modules, **{u|s}mem** of internal modules ・target command for testbench generation is (in `ifdef TESTBENCH macro): **{u|s}regN, {u|s}varN, [u]char/[u]short/[u]int,** name of **func** - with commands in `ifdef TESTBENCH, target command in module generation mode is : **#define**, **clock**, **sreset**, **areset**, **{u|s}inN**, **{u|s}outN**, **{u|s}mem**, thread name (**thread_main**) of testbench module ・with commands in `ifdef TESTBENCH, target command for testbench generation in hierarchy generation mode is : output name of **tap**, **#define** of internal modules, name of input/output port, **{u|s}mem** of internal modules, thread name (**thread_main**) of testbench module
E603	clock and reset have same name "*name*" in multiple modules	Name overlapping	clock and reset have same name among multi modules
W001	`define macro name "macroname" is not used	Unused com	**`define** macro name is not used for **`ifdef**
W002	SC_CTHREAD "threadname" has no reset process	Information	"n" is specified to th option of **cthread** command
W003	not recommend to specify "-header/-reset_header/-wait_header" with specifying -pipe option to cthread	Information	When pipe option is specified to any one of **cthread** commands, ・reset_header option or wait_header option is specified to **cthread** command ・header option is specified to **{u|s}mem** command ・header option is specified to **soft_reset** command
I001	the existed file "*filename*" was renamed to "*filename*"	Information	When generate file that has already existed
I002	"*name*" cannot be registered for VCD dump because the width is more than 999	Information	When bit width of port/signal is 1000 or more

No.	Message	Category	Output condition
I003	because ssgen generates only one SC_CTHREAD in testbench, please make SC_CTHREADs for every clocks, if necessary.	Information	When multi clock condition
I004	"*filename_tmp*" was generated because "*filename*" has already existed	Information	prevention to re-write an existing file
I005	skip invalid command [*commandname*]	Information	When there is an invalid command in **`include** file or **−include** file.
I006	ignored "-range_check" option because not supporting range check for array variable	Information	When specifying **−range_check** to array of **{u|s}outN, {u|s}regN, {u|s}varN, [u]char, [u]short, [u]int, {u|s}evN**
I007	ignored "-range_check" option because the width > 64	Information	When specifying **−range_check** to variable , whose width is more than 64, of **{u|s}outN, {u|s}regN, {u|s}varN, {u|s}evN**
I008	ignored "-range_check" option because the variable is a constant variable	Information	When specifying **−range_check** to variable with **const** identifier of **{u|s}varN, [u]char, [u]short, [u]int**
I009	-slec option is ignored for hierarchy generation mode	Information	-slec option is specified when a hierarchy definition file is input

*1 : Reserved words checked by E501 are shown in the following table.

Reserved words
int, long, short, signed, unsigned, float, double, bool, true, false, char, wchar_t, void, class, struct, union, enum, const, volatile, auto, extern, register, static, mutable, friend, typedef, explicit, inline, virtual, public, protected, private, operator, this, if, else, for, while, do, switch, case, default, break, continue, goto, return, try, catch, new, delete, dynamic_cast, static_cast, const_cast, reinterpret_cast, sizeof, typeid, throw, template, typename, export, namespace, using, and, and_eq, bitand, bitor, compl, not, not_eq, or, or_eq, xor, xor_eq, asm, sc_*, SC_*, systemc, reset_signal_is, async_reset_signal_is, dont_initialize, sensitive, read, write, pos, neg, posedge, negedge, posedge_event, negedge_event, wait, next_trigger, halt, always, assign, buf, bufif0, bufif1, casex, casez, cmos, deassign, defparam, disable, edge, endattribute, endcase, endfunction, endmodule, endprimitive, endspecify, endtable, endtask, event, force, forever, fork, highz0, highz1, ifnone, initial, input, join, large, macromodule, medium, module, nmos, notif0, notif1, output, parameter, pmos, primitive, pull0, pull1, pulldown, pullup, rcmos, reg, release, repeat, rnmos, rpmos, rtran, rtranif0, rtranif1, scalared, small, specify, specparam, strength, strong0, strong1, supply0, supply1, table, task, time, tran, tranif0, tranif1, tri, tri0, tri1, triand, trior, trireg, vectored, wand, weak0, weak1, wire, wor, attribute, begin, end, function, inout, nand, nor, package, use, xnor, TESTBENCH, _OSCI, _MEM_MODEL, _MODE_RTL, __CTOS__, CALYPTO_SYSC, T_MAX, HIRE_MAX, MEM_*, ssgen_*, tf, mem_rw1, mem_r1w1, mem_r1w1_2clk, CtoS_MAIN_LOOP, _CTOS_TOP, SSGEN_ASSERT, _COVERAGE, _SLEC_BBOX

*2-1 : Naming rules checked by E502 are shown in the following table (except **#define**).

No	Naming rules
1	Characters which can be used are alphanumeric character and underscore. However, brackets for array form(‘[‘ and ‘]’) can be used.
2	A name is not distinguished in a capital letter and a small letter.
3	The head character should be alphabetic character or underscore. However, the head character of modulename, port name and instance name must be alphabetic character.
4	The tail character should be alphanumeric character.
5	The length of name should be 511 or less.

*2-2 : Naming rules checked by E502 are shown in the following table (for **#define**).

No	Naming rules
1	Characters which can be used are alphanumeric character and underscore.
2	A name is not distinguished in a capital letter and a small letter.
3	The head character must be alphabetic character.
4	The length of name should be 511 or less.

Contact

SIDA S.Imamura shintaro.imamura.ry@renesas.com

Revision history

Rev. No	Classification	Effective date	Content	Approval by	Checked by	Written by
1.0	Established	Approved date	New created	FEDT Fujii 11.09.30	FEDT Oshima 11.09.30	FEDT Imamura 11.09.30
1.1	Modified	Approved date	Chapter 1 ・Changing the formal name of ssgen to “Synthesizable SystemC code Generator” Chapter 2 ・Adding information that it is possible to change the environment and the version arbitrarily of EDA tools by specifying the input to ssgen Chapter 3 ・Updating output files of ssgen ・Updating specification of prevention to re-write an existing file ・Updating specification of signal connections of hierarchy generation mode ・ Deleting “High-speed simulation” from accessing memory array mode. ・Adding chip select port to 2 port memory Chapter 4 ・Updating method of CtoS execution Chapter 5 ・Adding −only_script and −notb to command line option Chapter 6 ・Updating specification of module generation mode command - Adding space_indent, env_systemc, env_vcs, env_ies, env_ctos, env_vcs_gcc - Updating specification of option for soft_reset、uoutN、soutN、uregN、sregN、uvarN、svarN、char、uchar、short、ushort、int、uint、umem、smem、cthread ・ Updating specification of hierarchy module generation mode - Adding space_indent, env_systemc, env_vcs, env_ies, env_ctos, env_vcs_gcc, bind - Updating specification of option for tap Chapter 7 ・Updating example of output file of ssgen Chapter 8 ・Updating specification of macro function for memory access Chapter 9 ・Updating output messages	FEDT Fujii 12.03.28	FEDT Oshima 12.03.28	FEDT Imamura 12.03.28
1.1.1	Modified	Approved date	Chapter 5 ・Adding -include option Chapter 6 ・Adding mem_suffix command ・ Adding `include command to hierarchy generation mode Chapter 9 ・Updating output messages	FEDT Fujii 12.04.25	FEDT Oshima 12.04.23	FEDT Imamura 12.04.23

Rev. No	Classification	Effective date	Content	Approval by	Checked by	Written by
1.1.4	Modified	Approved date	Chapter 4 ・Delete (Note) of (7) Chapter 6 ・Adding style_alloc command ・Adding “-nowd” option to {u|s}mem command. Chapter 7 ・Adding “_CTOS_TOP” macro ・Adding “*_NOWD” macro to macro function of memory access Chapter 8 ・Adding “*_NOWD” macro to macro function of memory access Chapter 9 ・Updating output messages	FEDT Fujii 12.09.26	FEDT Oshima 12.09.26	FEDT Imamura 12.09.26
1.2	Modified	Approved date	All ・Adding chapter 7 Chapter 2 ・Updating default version of EDA tools Chapter 6 ・Adding “-clk_edge” option to cthread command Chapter 8 ・Updating output description of soft_reset command Chapter 10 ・Updating output messages -Changing `define command of E202 -Delete E204 and E604 -Adding E209 and E210	FEDT Fujii 12.11.30	FEDT Oshima 12.11.30	FEDT Imamura 12.10.31
1.3	Modified	Approved date	Chapter 2 ・Updating version of ActivePerl Chapter 3 ・Adding memory interface module ・Adding memory access pattern (5)(6)(7) Chapter 6 ・Adding “rw1_wa”, “rw1_ra” and “rw1_re” to mem_suffix command ・Adding “_SLEC_BBOX” to #ifdef command ・Adding “-range_check” option to {u|s}outN, {u|s}regN, {u|s}varN, [u]char, [u]short and [u]int command ・Adding “rw1:r” and “rw1:w” to {u|s}mem command ・Adding “-rtl” option to sub command Chapter 7 ・Adding initial value, maximum value and minimum value Chapter 8 ・Adding an example of “-range_check” option ・Adding the explanation of “SSGEN_ASSERT” ・Adding an example of memory interface module ・Adding macro for “rw1:r” and “rw1:w” Chapter 10 ・Updating output messages -Changing sub command of E202, E308, E311, E312, E401, E402 -Adding E023, E414, I006, I007 and I008 ・Adding “SSGEN_ASSERT”, “_COVERAGE” and “_SLEC_BBOX” to reserved word.	FEDT Fujii 13.1.23	FEDT Oshima 13.1.23	FEDT Imamura 13.1.22

Rev. No	Classification	Effective date	Content	Approval by	Checked by	Written by
1.4	Modified	Approved date	Chapter 3 ・Describing recommendation that memory port access description should be used in SystemC simulation Chapter 6 ・Adding vcd_trace command ・Adding {u|s}evN command ・Changing "_DEBUG_SIM" to "_DEBUG*" in #ifdef command ・ Adding "-var2reg", "-debug_trace" and "-debug_macro" option to {u|s}varN, [u]char, [u]short and [u]int command ・Supporting three-dimension array with const ・Adding "-noad" option to {u|s}mem command ・Supporting memory in tap command ・Supporting floating port in bind command Chapter 7 ・Adding initial value and maximum/minimum value in range check Chapter 8 ・Adding "_NOAD" and "_NOAW" Chapter 9 ・Adding "_NOAD" and "_NOAW" Chapter 10 ・Updating output messages -Changing specification E001, E002, E003, E006, E016, E021, E101, E202, E203, E206, E207, E209, E305, E306, E308, E312, E329, E501, E502, E601, I006, I007 -Adding E339, E340, E341, E342, E343, E408 ・Deleting "_DEBUG_SIM" from reserved word.	FEDT Fujii 13.5.13	FEDT Oshima 13.5.9	FEDT Imamura 13.5.9
1.5	Modified	Approved date	Chapter 2 ・Updating information of related tools Chapter 3 ・Updating the figure of ssgen input/output ・Adding that memory models have function storing previous value of read data port. ・Not supporting memory array description from this version ・Supporting dual port memory Chapter 5 ・Adding –checker, -slec, -subdir option Chapter 6 ・Adding env_ssgen, env_1team, env_sschecker, env_slec ・Adding dual port memory to mem_suffix ・Adding dual port memory to {u|s}mem ・Adding –pipe_macro and –pipe_max option to cthread ・Adding –port_pfx option to sub ・Supporting sub and bind in #ifdef area Chapter 8 ・Updating descriptions of memory model, CtoS script and simulation script ・Adding descriptions of SLEC script and checker script Chapter 10 ・Updating output messages -Changing specification E401 -Adding E114, E344, E415, E416 and I009			SIDA Imamura 13.9.5