WEB-00528072-01.00E
ZSG-F31-11-0039-01

SC-HEAP user modeling environment users manual

Renesas Electronics Corporation

2011/5/11 Rev. 1.00

Introduction

We have received the requirements for the simulation environment connected with a variety of peripheral IPs from many fields. However, SC-HEAP could not meet to the requirements because SC-HEAP did not have enough peripheral IP. Moreover, it is also impossible that the SC-HEAP developer prepares all peripheral IP peculiar to the user. From such a background, the User Modeling Environment aims to give the environment which is able to connect the IP created by user, by preparing of standard interface to SC-HEAP.

- Refer to bibliography 2 for a necessary settings for execution. (important)
- The configuration same as OSCI SC-HEAP can be used for the created simulator. Refer to bibliography 2 for details.
- It is preferable not to connect IP that synchronizes with the clock signal because it leads to the simulation speed decrease though the clock port has been prepared.

- This manual is described on the assumption that user have already understood the specification of OSCI TLM2.0.

Contents

Feature of user modeling environment

- Feature
  - The adaptor is put between IOB in SC-HEAP and peripheral IP. The connection of peripheral IP is realized by converting the interface from the local bus interface.
  - Interface: OSCI TLM2.0 is used.
    tlm::tlm_fw_transport_if<32,tlm_base_protocol_type,0>、
    tlm::tlm_bw_transport_if<32,tlm_base_protocol_type,0>
  - Socket: The socket which derives from
    tlm_target_socket<32,tlm_base_protocol_type,0> and
    tlm_initiator_socket<32,tlm_base_protocol_type,0> of OSCI TLM2.0 is prepared.
    TlmTargetSocket、TlmInitiatorSocket
  - Communication:
    Normal access: Approximately-timed using the return path is used.
    Debug access: Debug transport interface is used.
  - Payload:generic payload is used.
    (The lock signal etc. are enhanced. tlm_extension class is used.)
  - Connection of plural SLAVE IPs
  - Connection of plural MASTER IPs
  - Connection to interrupt port
  - Master IP connection of DMA
  - Available on Linux and Windows.

Change point in SC-HEAP V3.10

This sheet is not necessary to disclose to user

- Change point in V3.10
  - The connection with target IP which has pure target socket of OSCI TLM2.0 I/F
    tlm_target_socket<32,tlm_base_protocol_type,0> is supported.
    However, the tlm_target_socket<32,tlm_base_protocol_type,0> must be directly bound to the interface.(not via other sockets.)
    About initiator IP connection, The IP which has TlmInitiatorSocket is permitted to connect.

Environment (for Linux)

The following environments are necessary to use the user modeling environment.

- Machine environment
  - Red Hat Enterprise Linux WS release 4 Update 8
  - HP Proliant DL365 etc.
    Note) It seems to be executed normally, if Intel-affiliated machine is used. However, the execution cannot be guaranteed.
- SystemC library
  - OSCI-SystemC2.20
- Compiler of simulator
  - gcc 3.4.6
- Debugger of target
  - GHS Multi 5.0.5 pr1650 Linux version
  - The debugging server is rteserv2(MULTI v5.1.6C V800).
    (note) Please obtain rteserv2, the manual, and the license from GHS.

- Additionally, refer to bibliography 2 for a necessary setting in execution.

Environment (for Windows)

The following environments are necessary to use the user modeling environment.

- Machine environment
  - Windows XP Professional SP3
  - NEC PC98-NX Mate MY28A/E-5

  Note) It seems to be executed normally, if another Windows PC is used. However, the execution cannot be guaranteed.
- SystemC library
  - OSCI-SystemC2.20

  - The SystemC library should be built by msvc80(VisualStudio2005).

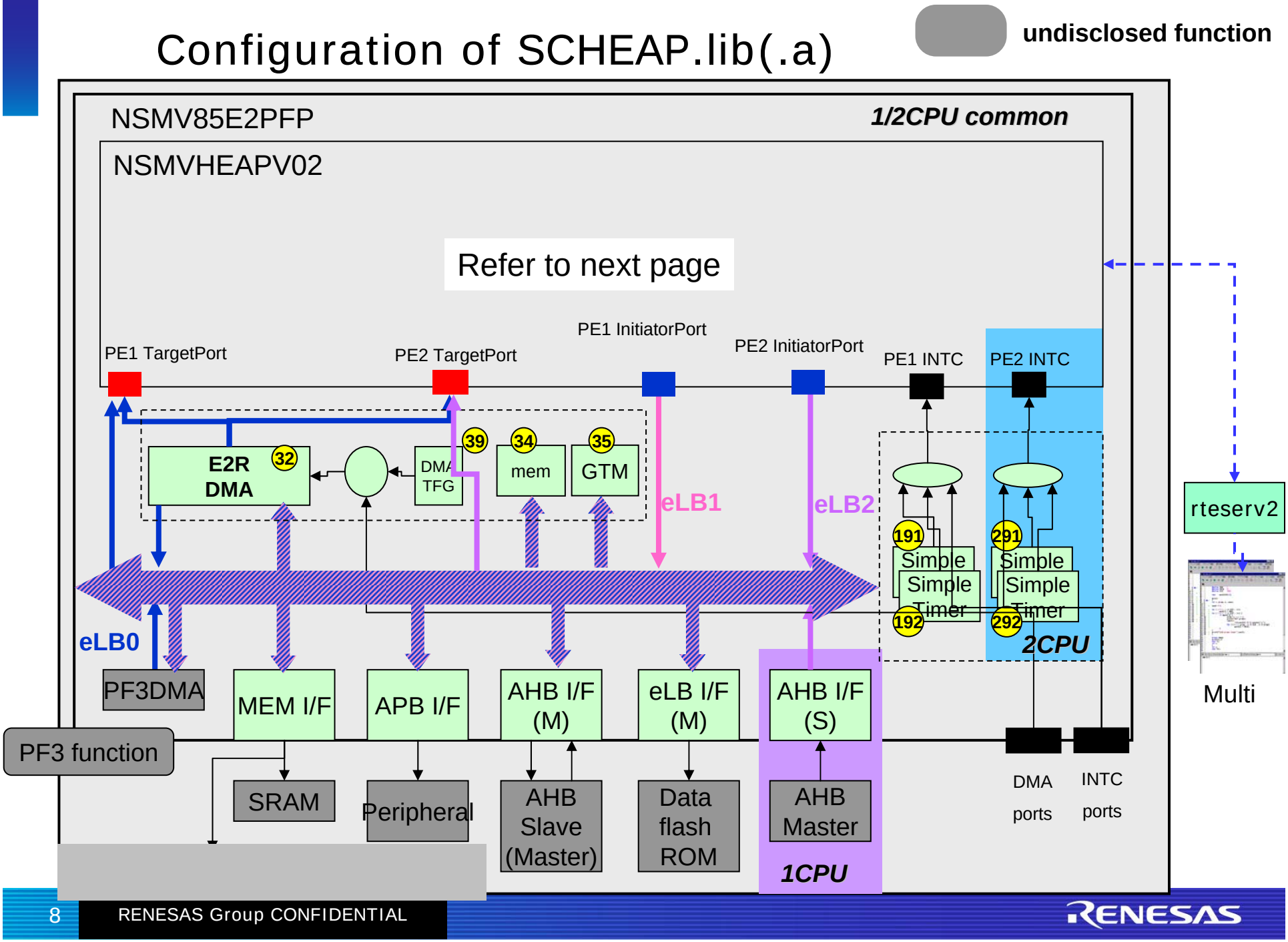
    Please go according to the following procedures.
    1. The project of msvc71(VisualStudio2003) is changed and opening -> is changed into msvc80 with msvc80 once in closing.
    2. The project file is opened again.
    3. Please open the property page by project -> property, and do the following setting changes.
       - Configuration property -> C/C++ -> preprocessor : _CRT_SECURE_NO_DEPRECATE is added.
       - Configuration property -> C/C++ -> details: 4996 is added.
    4. The build is done, and the library is made.

  - And SystemC library should be built with /MD option. (not /MT)
- Compiler of simulator
  - VisualStudio 2005
- Debugger of target
  - GHS Multi 5.0.5 pr1650 Windows version
  - The debugging server is rteserv2(MULTI v5.1.6C V800).

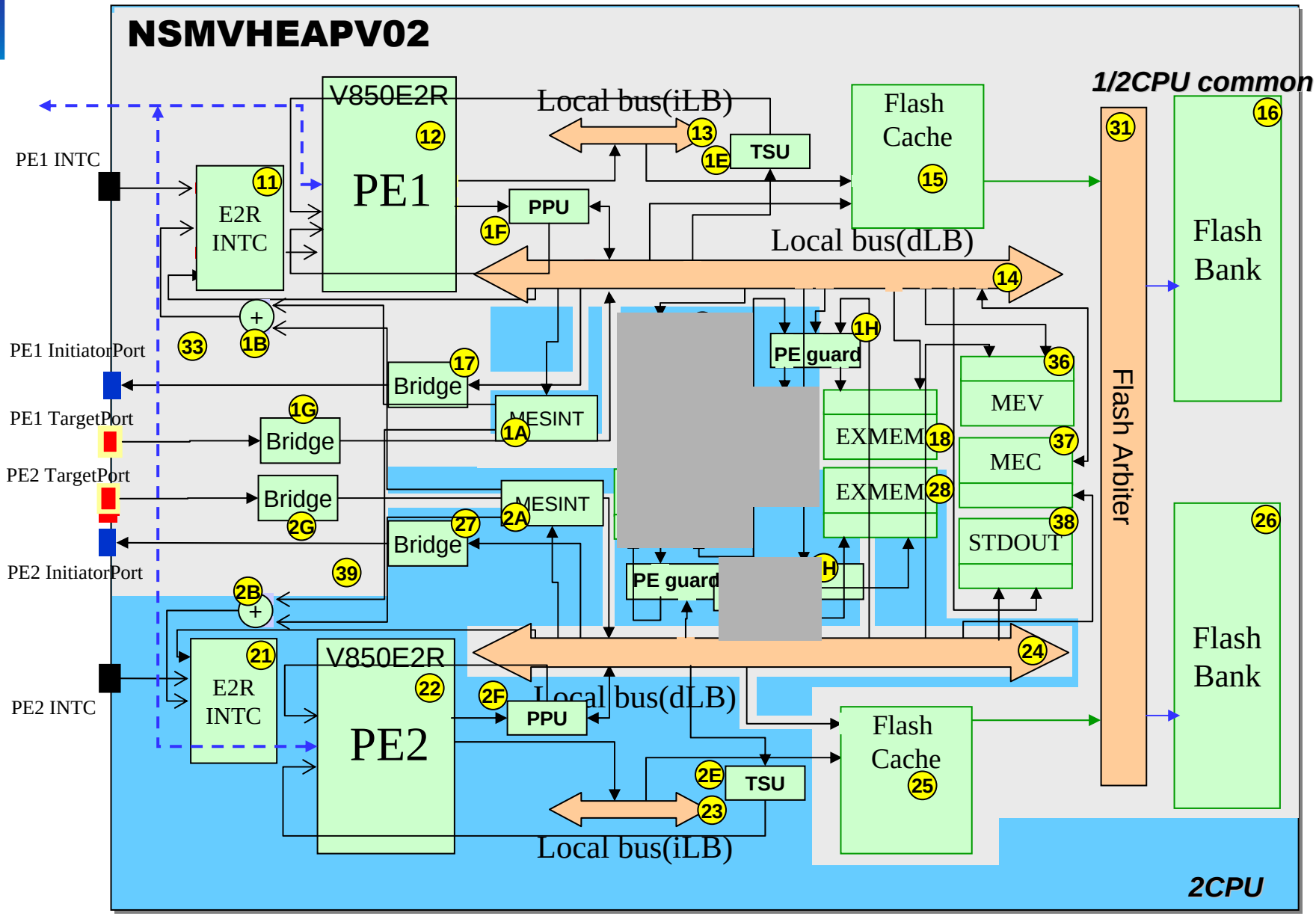
    (note) Please obtain rteserv2, the manual, and the license from GHS.

- Additionally, refer to bibliography 2 for a necessary setting in execution.

Configuration of SCHEAP.lib(.a)

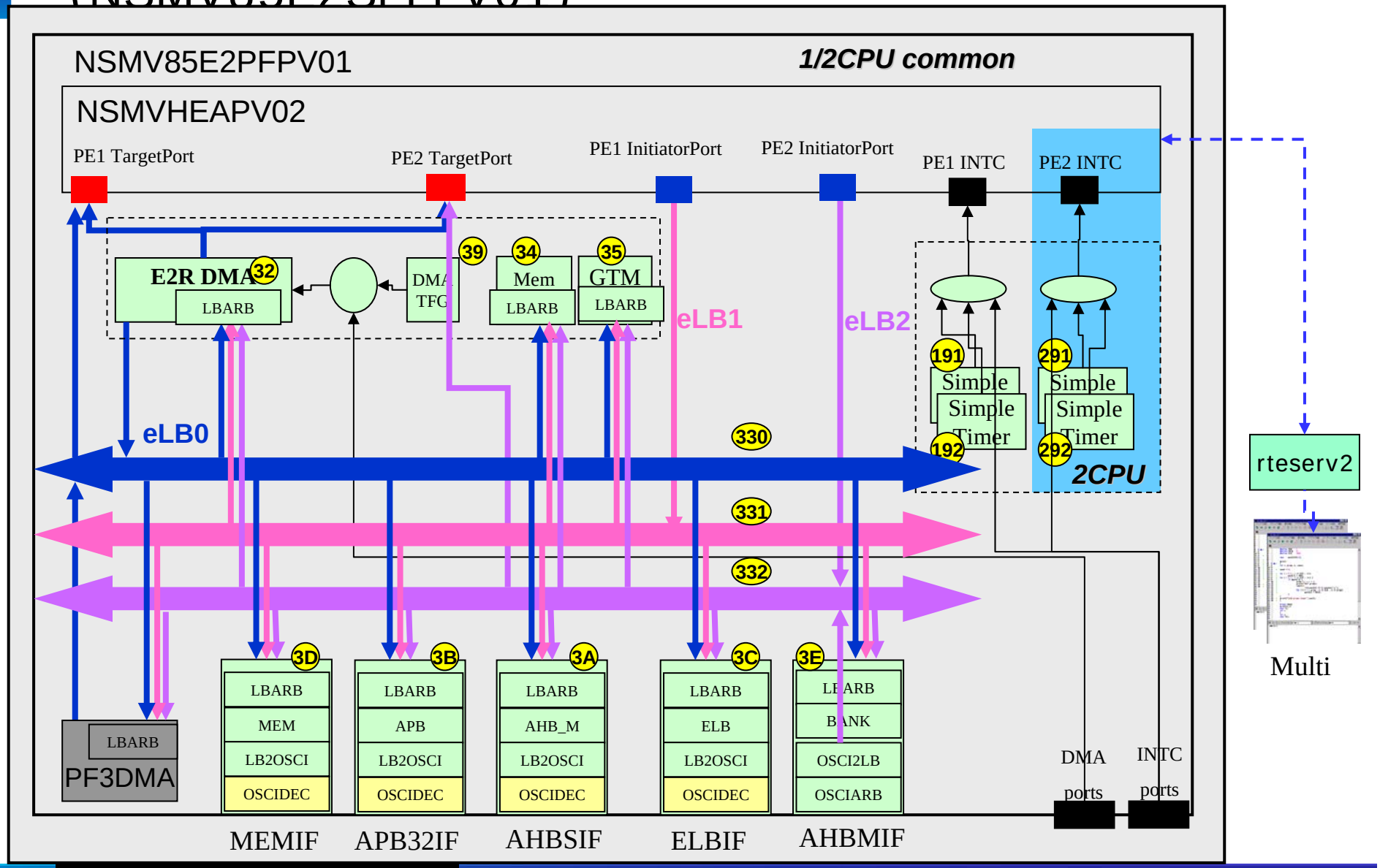


HEAP hierarchy (NSMVHEAPV02)

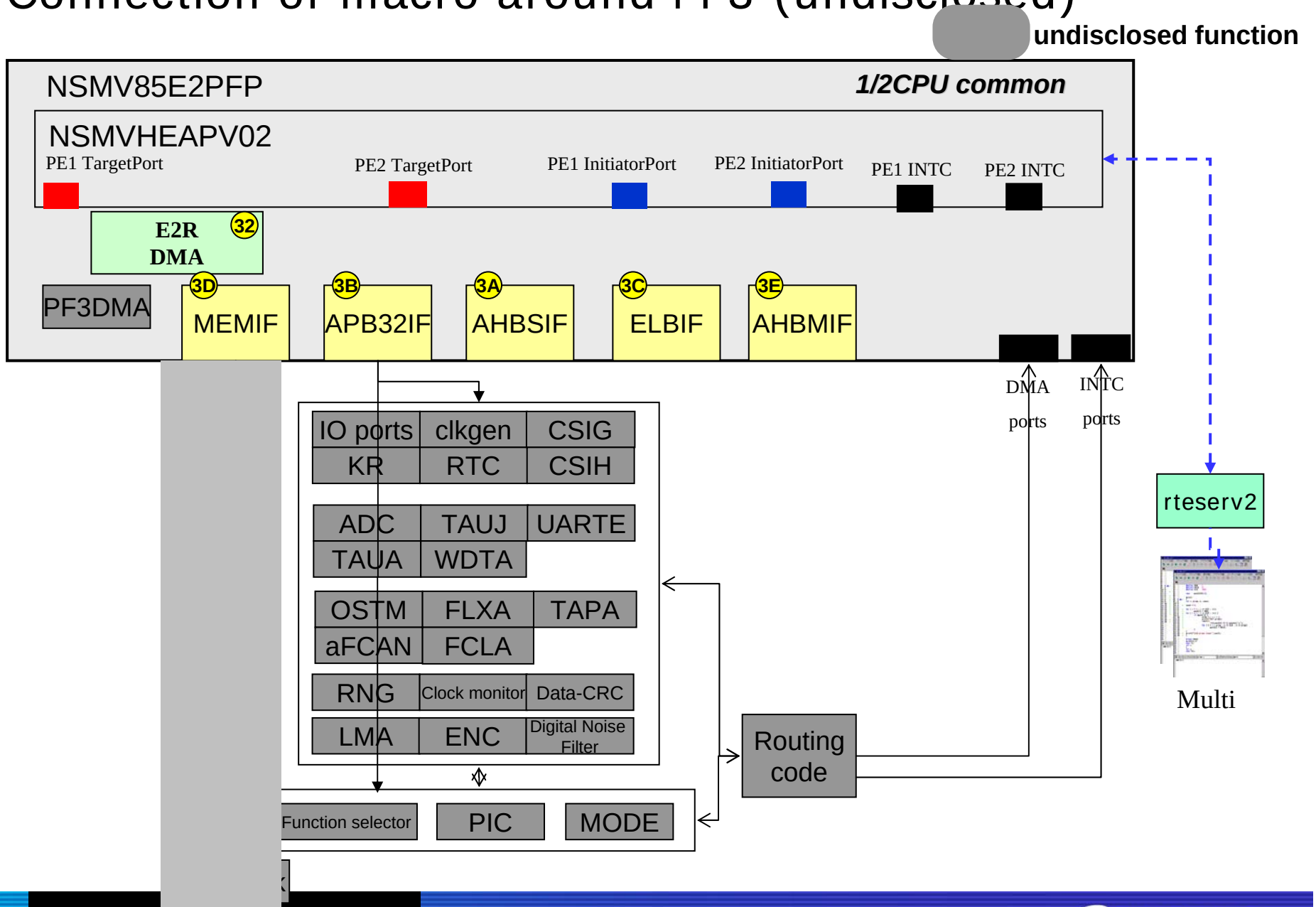


E2SPFP hierarchy (NSMV85E2SPFPV01)

undisclosed function



Connection of macro around PF3 (undisclosed)



Socket used in user modeling environment

- TlmTargetSocket and TlmInitiatorSocket which derives from tlm_target_socket<32,tlm_base_protocol_type,0> and tlm_initiator_socket<32,tlm_base_protocol_type,0> of OSCI TLM2.0, are used for the connection between peripheral IP and SC-HEAP.

  - The connection with slave IP which has pure tlm_target_socket<32,tlm_base_protocol_type,0> is supported from SC-HEAPV3.10. However, the tlm_target_socket<32,tlm_base_protocol_type,0> must be directly connected to the interface.(not via other sockets.) About master IP connection, the IP which has the TlmInitiatorSocket is permitted to connect.

- TlmTargetSocket and TlmInitiatorSocket have added the following functions to tlm_target_socket<32,tlm_base_protocol_type,0> and tlm_initiator_socket<32,tlm_base_protocol_type,0>.

  > The base address and the size are transmitted from SC-HEAP to SLAVE.

  Refer to Realization setBaseAddressSize() of base address size set…

  (To locate SLAVE according to the address map.)

  > SC-HEAP memorizes the pointer of the target socket of SLAVE.

  (To realize the address decoding of SC-HEAP.)

Payload used in the user modeling environment

- Generic payload of OSCI TLM2.0 is used for payload between peripheral IP and SC-HEAP.

- Moreover, payload is enhanced by using the tlm::tlm_extension class.

  The enhanced data member is shown as follows.

  > Lock signal
  > signal of DMA transfer cycle: DMACH (signal that DMA for PF3 outputs)
  > signal of the DMA final forwarding cycle: DMALAST (signal that DMA for PF3 outputs)
  > Pointer of initiator socket

  (To realize plural master IP connections. Use it only at the master IP connection.)

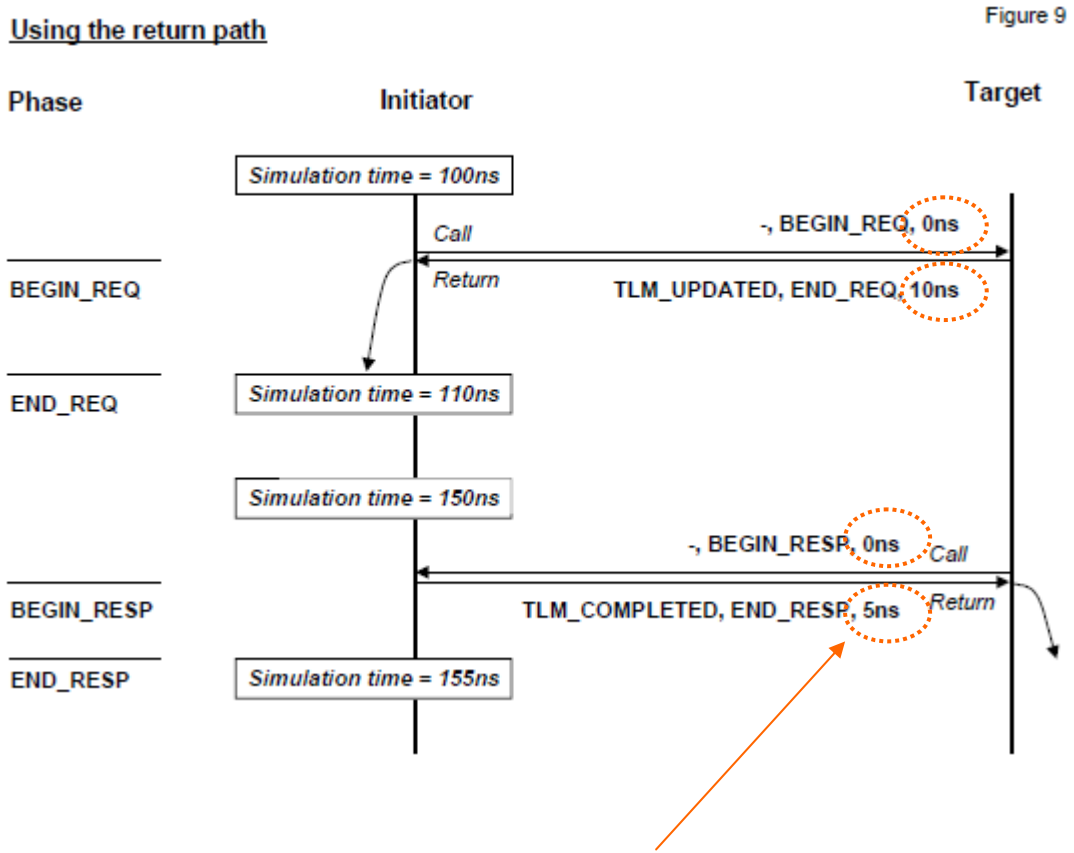
SC-HEAP <-> user module communication

Normal communication
Communicate with Approximately-timed using the return path of OSCI TLM2.0.

Communication at debug access
Communicate by the debug transaction interface of OSCI TLM2.0.

Refer to "OSCI TLM2 USER MANUAL" for details (bibliography 1).

Approximately-timed using the return path



However, the third argument (sc_core::sc_time&) of nb_transport_fw and nb_transport_bw is always used as 0 in this user modeling environment. There is no influence in the simulation even if the values other than 0 are set. (Local time cannot be passed.)

Configuration parameters

undisclosed function

- Configuration parameters for user modeling
  - Specified in heap.cfg.

Meaning	Specification method	Default value
Peripheral IP connection (undisclosed)	[PERIPHERAL]=NONE/PF3/SMPILS/PF3&SMPILS NONE: Only the user module PF3: User module + PF3 peripheral IP*1 SMPILS: User module + SMPILS*2 PF3&SMPILS: User module + PF3 peripheral IP*1 + SMPILS*2	NONE
DMA master	[DMA_MASTER_TYPE]=INTERNAL/EXTERNAL To connect external DMA master, EXTERNAL is specified.	INTERNAL

The parameters same as OSCI SC-HEAP can be used for others.
Refer to "SC-HEAP V3.10 users manual" for details. (bibliography 2)

*1: The simulation which includes the PF3 peripheral macro is supported since SC-HEAPV3.00. The PF3 peripheral macro is undisclosed.

*2: The connection with SMPILS(Simulink) is supported since SC-HEAPV3.00 unofficially. The SMPILS connect function is undisclosed.

Sample of environment

Method of user platform creation

Connection of functions pltfrm() *1, pltfrmFC() *2 and instantiation of NSMV85E2SPFPV01. The description that calls pltfrm() and pltfrmFC() is prepared from sc_main(). Moreover, function pltfrmDelete() and pltfrmFCDelete() that does delete also prepare the instance made substance with pltfrm() and pltfrmFC().

Platform

```
int
sc_main( int argc, char **argv )
{
 E2SPFP = new NSMV85E2SPFPV01(
       "E2SPFP", config_file, clock_period, clock_time_unit, tf);

 // connection
 pltfrmFC(E2SPFP,config_file);
 pltfrmPF3(E2SPFP,config_file);
 pf3 = PF3_0; // Pointer of instance of PF3
 pltfrm(E2SPFP,pf3,config_file);
  pltfrmSmpils(E2SPFP,pf3,config_file);
 pltfrmPF3GND(E2SPFP,pf3,config_file);

 sc_start( cycle_number * clock_period, clock_time_unit );

 // delete
 pltfrmFCDelete();
 pltfrmPF3Delete();
 pf3 = PF3_0; // Pointer of instance of PF3
 pltfrmDelete();
  pltfrmSmpilsDelete();
 pltfrmPF3GNDDelete();

 if((unsigned long long)org_sc_simulation_time() >= cycle_number *
       clock_period){
 sc_stop();
 }
 return 0;
}
```

In SC-HEAP.a

Function call

Function call

pltfrm.cpp

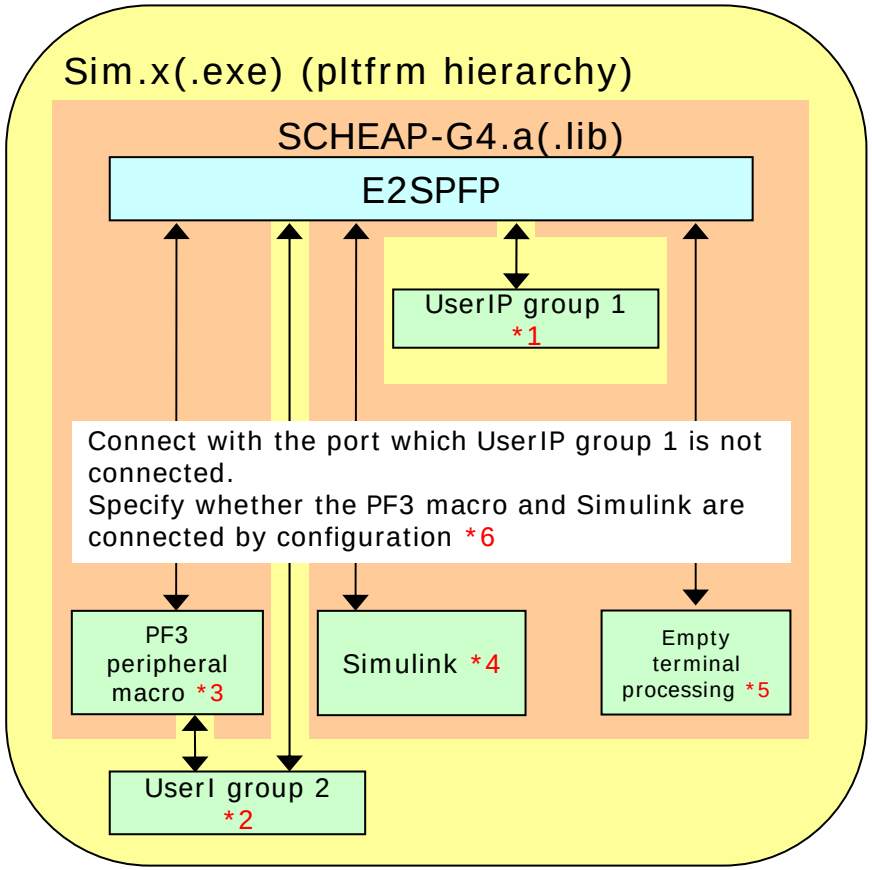
```
#include “pltfrm.h"

void
pltfrm(NSMV85E2SPFPV01 *E2SPFP, PF3 *PF3,
 const char *config_file)
{
The user describes the connection with NSMV85E2SPFPV01.
}

void
pltfrmFC(NSMV85E2SPFPV01 *E2SPFP,
 const char *config_file)
{
The user describes the connection with NSMV85E2SPFPV01.
}

void
pltfrmFCDelete(void)
{
The user describes the connection with NSMV85E2SPFPV01.
}

void
pltfrmFCDelete(void)
{
The user describes the connection with NSMV85E2SPFPV01.
}
```

***1: Function that describes connection of user IP to be made to give priority more than connection of PF3 macro**

***2: Function that does description that connects user IP with port that PF3 macro did not connect (Here is used usually).**

User connection descriptive function pltfrm()/pltfrmFC()

User usually writes connection between E2SPFP and the user module in pltfrm().
User writes connection to pltfrmFC() only when the user wants to connect user macro with precedence to the port which PF3 macro connects.



*1: User module higher-priority than PF3 peripheral macros.
User describes the connection with them in pltfrmFC().

*2: User module connected with port not used by PF3 peripheral macro.
User describes the connection in pltfrm().

*3: It is described in pltfrmPF3().
It is included in SCHEAP-G4.a(.lib).

*4: It is described in pltfrmSmpils().
It is included in SCHEAP-G4.a(.lib).

*5: Finally, all empty ports are treated. It is described in pltfrmPF3GND().
It is included in SCHEAP-G4.a(.lib).

*6: Refer to the configuration.

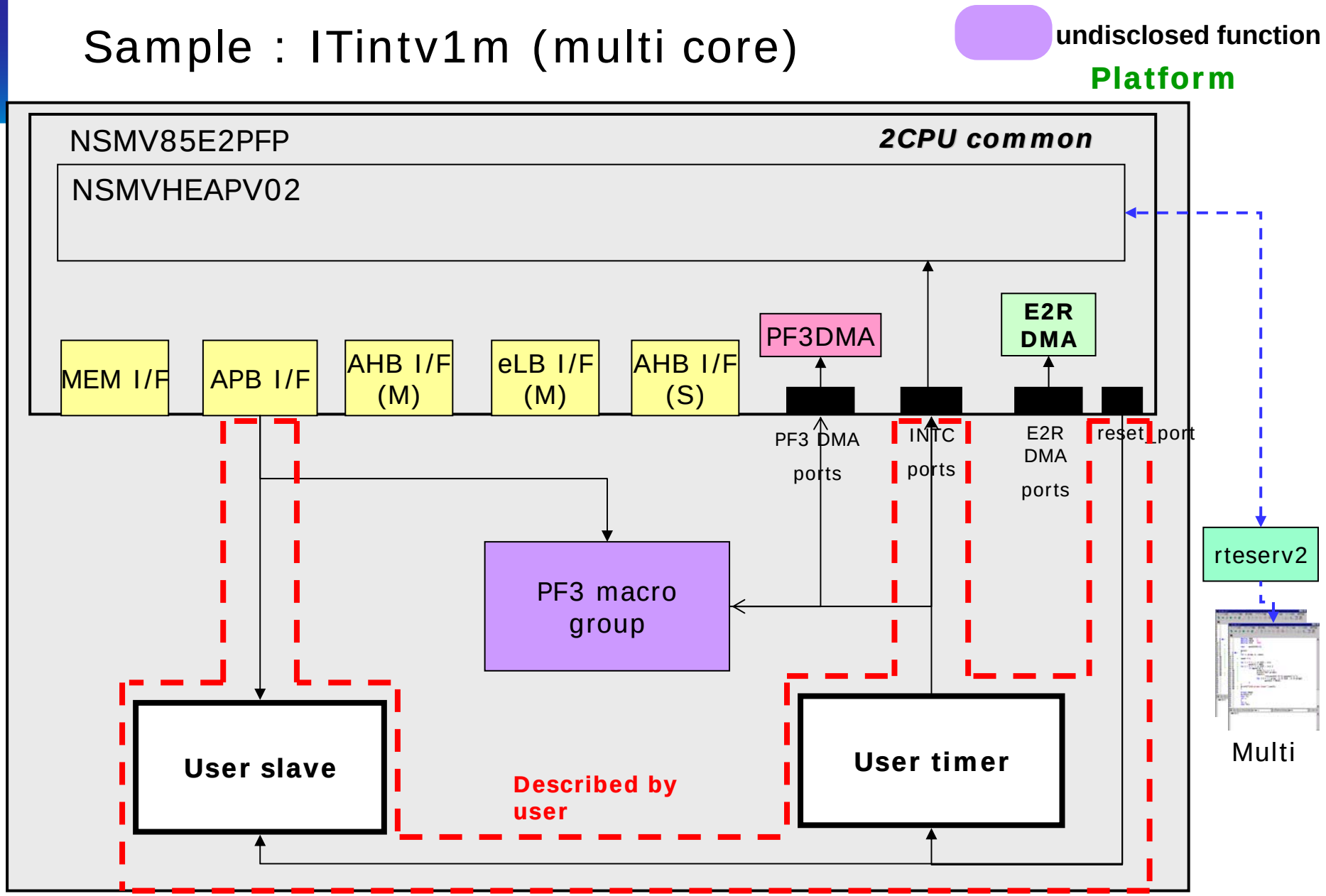
SC-HEAP-G4.a(.lib) port list

Platform

Port name	Type	Variable identifier	Instance name
AHB slave connection I/F port	TlmInitiatorSocket **	AHBSIF_i_socket	AHBSIF_3A
APB32 I/F port	TlmInitiatorSocket **	APB32IF_i_socket	APB32IF_3B
ELB I/F port	TlmInitiatorSocket **	ELBIF_i_socket	ELBIF_3C
External memory I/F port	TlmInitiatorSocket **	MEMIF_i_socket	MEMIF_3D
AHB master connection I/F port	TlmTargetSocket **	AHBMIF_t_socket	AHBMIF_3E
Clock output	sc_in<bool>	clock_port	clock_port
Reset output	sc_in<bool>	reset_port	reset_port
Low clamping signal	sc_out<bool>	lowconst_port	lowconst_port
PE1 feint interrupt port	sc_in<bool>	pe1_feint_port	pe1_feint_port
PE2 feint interrupt port	sc_in<bool>	pe2_feint_port	pe2_feint_port
PE1 fenmi interrupt port	sc_in<bool>	pe1_fenmi_port	pe1_fenmi_port
PE2 fenmi interrupt port	sc_in<bool>	pe2_fenmi_port	pe2_fenmi_port
PE1 eiint interrupt port	sc_in<bool> [256]	pe1_eiint_port	pe1_eiint_portx (x is a serial number).
PE2 eiint interrupt port	sc_in<bool> [256]	pe2_eiint_port	pe2_eiint_portx (x is a serial number).
E2R DMA dmarq port	sc_in<bool> ***	dmarq_port	dmarq_portx (x is a serial number).
E2R DMA intio port	sc_in<bool> ***	intio_port	intio_portx (x is a serial number).
E2R DMA dmaak port	sc_out<bool> ***	dmaak_port	dmaak_portx (x is a serial number).
E2R DMA dmach port	sc_out<bool> ***	dmach_port	dmach_portx (x is a serial number).
E2R DMA dmalast port	sc_out<bool> ***	dmalast_port	dmalast_portx (x is a serial number).
E2R DMA dmanmi port	sc_out<bool> *	dmanmi_port	dmanmi_portx (x is a serial number).
Reset input port for PF3 DMA	sc_in<bool>	reset	reset
Operation frequency input port for PF3 DMA	sc_in<uint64>	freq	freq
PF3 DMA DMACTRG port	sc_in<bool> [128]	dmactrg	dmactrgx (x is a serial number).
PF3 DMA DTSTRG port	sc_in<bool> [128]	dtstrg	dtstrgx (x is a serial number).
PF3 DMA INTCT port	sc_out<bool> [16]	intct	intctx (x is a serial number).
PF3 DMA INTDMA port	sc_out<bool> [16]	intdma	intdmax (x is a serial number).
PF3 DMA IRQ port	sc_out<bool> [128]	irq	irqx (x is a serial number).

Refer to the PIN I/F connection for the connection method.

Sample : ITintv1m (multi core)



The feature: The user timer is connected with the interruption port.

Sample : ITintv1m (Windows)

Platform

The red-letter part: User writing part when user newly make the environment.

Directory configuration (for Windows)

```
pltfrmCompile
 |
 |- build
 |   |-- ITintv1m                Simulator building environment      pltfrm.cpp, pltfrm.h
 |                               Simulator which connected user memory and user timer.
 |                               ITintv1m execution environment      heap.cfg, *.map, *.bat
 |
 |- include                      Header files
 |
 |- lib                          Archive of SC-HEAP       SCHEAP-G4.lib
 |
 |- multi                        Debugging server storage directory
 |
 |- soft                         Sample program
 |  |- intv1m_v4_heap            Interrupt (intv1m_v4_heap) software
 |
 |- models
    |- TIMER                     User timer
    |- ATSLAVE                   User slave
    |- tlm                       OSCI TLM 2.0
```

Sample : ITintv1m (Linux)

Platform

The red-letter part: User writing part when user newly make the environment.

Directory configuration (for Linux)

```
pltfrmCompile
 |
 |- build
 |   |-- ITintv1m                Simulator building environment     pltfrm.cpp, pltfrm.h
 |                               Simulator which connected user memory and user timer.
 |                               ITintv1m execution environment     heap.cfg, *.map, *.csh
 |
 |- include                      Header file
 |
 |- lib                          Archive of SC-HEAP      SCHEAP-G4.a
 |
 |- lib-modelsO3                 Objectfile (*.o) storage place
 |
 |- multi                        Debugging server storage directory
 |
 |- soft                         Sample program
 |  |- intv1m_v4_heap            Interrupt (intv1m_v4_heap) software
 |
 |- models
   |- TIMER                      User timer
   |- ATSLAVE                    User memory
   |- tlm                        OSCI TLM 2.0
```

Sample : ITintv1m

User creation part

Platform

- Platform connection

  build/ITintv1m/pltfrm.cpp, pltfrm.h

- User IP

  models/TIMER
  models/ATSLAVE (memory)

Sample : ITintv1m

Platform

Simulator executable file building procedure

[Windows]

- Build in pltfrmCompile/build/ITintv1m/msvc80.
  - Execute scheap.sln.
    - Use Release as the solution configuration.
    - Executable file (sim.exe) is created in pltfrmCompile/build/ITintv1m/msvc80.

[Linux]

- Build in pltfrmCompile/build/ITintv1m.
  - Execute make.
    - pltfrm.tb.mk is called from pltfrm.mk and pltfrm.mk from Makefile
    - Executable file (sim.x) is created in pltfrmCompile/build/ITintv1m.

Sample : ITintv1m

User code

pltfrm.cpp

```
//************** User include header **************//
#include "pltfrm.h"

//************** User include header **************//
void pltfrm(NSMV85E2SPFPV01 *E2SPFP, PF3 *PF3,
 const char *config_file=NULL)
{
 //************** User code ************
 //----- instance definition
 ATSLAVE *ATSLAVE_1;
 TIMER *TIMER_1;

 //----- instantiation
 ATSLAVE_1 = new ATSLAVE("ATSLAVE_1",...);
 TIMER_1 = new TIMER("TIMER_1",...);

 //----- connection to INTC
 E2SPFP->pe1_eiint_port[10](intcmd_sig);
 E2SPFP->pe2_eiint_port[11](intcmd_sig);

 //----- connection
 E2SPFP->APB32IF_i_socket.bind(ATSLAVE_1->target_socket);

 //----- for timer
 TIMER_1->reset(E2SPFP->reset_port);
 TIMER_1->intcmd(intcmd_sig);

 ATSLAVE_1->reset_port(E2SPFP->reset_port);
 //************** User code **************//
}
```

The pointer of the PF3 macro group is passed.
The user is made to be able to do connected description of the PF3 macro in the future with this pointer.

The timer and the INTC input are connected.

It connects it through OSCI TLM.

The clock and reset are direct connections of the port.

Platform

```
void pltfrmFC(NSMV85E2SPFPV01 *E2SPFP,
 const char *config_file=NULL)
{
}

void pltfrmFCDelete()
{
}

void pltfrmDelete()
{
}
```

The connection is described in pltfrm() for the connection to the port where PF3 peripheral macro is not connected. To connect user macro higher-priority than PF3 peripheral macro, user describes it in pltfrmFC().
Moreover, after the simulation, if user wants to delete explicitly the instance which is instanciated in pltfrm() and pltfrmFC(), user can delete it in pltfrmDelete() and pltfrmFCDelete().

Sample : ITintv1m

Platform

User code

pltfrm.h

```
#ifndef PLTFRM_H
#define PLTFRM_H

#include “NSMV85E2SPFPV01.h"
//************** User include header **************//
#include “ATSLAVE.h"
#include “TIMER.h"
#include “PF3.h"
//************** User include header **************//

// signal defined
sc_signal<bool> intcmd_sig;                                    // Signal for timer

//************** User code **************//
typedef unsigned int ADDRESS_TYPE;
typedef unsigned char DATA_TYPE;
//************** User code **************//

#endif // PLTFRM_H
```

Sample : ITintv1m (Windows)

Platform

Creation of scheap.sln (for Windows)

Refer to pltfrmCompile/build/ITintv1m/msvc80/scheap.sln.

Sample : ITintv1m (Linux)

Creation of makefile (for Linux)

Platform

The red-letter part: User writing part when user newly make the environment.

plrfrmCompile/ITintv1m/pltfrm.mk

```
# Target selection variables
MODE ?= release
KERNEL ?= systemc
LIBTYPE ?= static

# Name
MODEL                   =
TARGET                  = sim.x

# Models
MODEL_ATSLAVE           = ATSLAVE
MODEL_TIMER             = TIMER

# Location
PROJ_HOME               ?= $(shell pwd)/../..
LIBPATH_ROOT?= $(PROJ_HOME)/lib-modelsO3
MODELPATH_ROOT          = $(PROJ_HOME)/models
INCLUDE_HEADER          = $(PROJ_HOME)/include
ifdef TLM_INC_DIR
TLM_INCLUDE_HEADER      ?= ${TLM_INC_DIR}
else
TLM_INCLUDE_HEADER ?= $(PROJ_HOME)/../pltfrmCompile/models/tlm
endif
```

User module

```
# SystemC Location and architecture
ifeq "$(shell uname -n)" "sdlpc567"
 SYSTEMC_HOME           ?= /eda_tools/systemc
else
 SYSTEMC_HOME ?=
                /home/product/systemc/tools/systemc
endif
ifeq "$(shell uname -s)
 TARGET_ARCH =
else
 TARGET_ARCH = linux
endif
SYSTEMC_INCPATH         = $(SYSTEMC_HOME)/include
SYSTEMC_LIBPATH = $(SYSTEMC_HOME)/lib-$(TARGET_ARCH)
SYSTEMC_LIB             ?= systemc

# Make command
MAKE                    ?= /usr/bin/gmake
override MAKEFLAGS      += --no-print-directory

# RM command
RM                      = rm
RM_OPT                  = -f
```

Path of OSCI SystemC library

Sample : ITintv1m (Linux)

Platform

pltfrmCompile/ITintv1m/pltfrm.mk (continuation)

```
# Linux RedHat7.3 - gcc2.96
CXX     ?= /usr/bin/g++
DEFFLAG = -D__SYSTEMC_21__ -DSC_USE_SC_STRING_OLD ¥
        -DSC_INCLUDE_DYNAMIC_PROCESSES              ¥
        -DNSMVINTC711_DEF -DCM_REPORT_OUT           ¥
        -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 ¥
        -DLINUX -DPLTFRM_DEF
ifeq ($(MODE),debug)
OPTFLAG         = -O0 -m32
DBGFLAG         = -ggdb3 -DDEBUG
else
OPTFLAG         = -O3 -m32
DBGFLAG         = -DNDEBUG
endif
INCPATH = -I$(SYSTEMC_INCPATH) -I. -L$(SYSTEMC_LIBPATH) ¥
        -I$(INCLUDE_HEADER) -I$(TLM_INCLUDE_HEADE

# build rules
all:
        if test ! -d $(LIBPATH_ROOT); then mkdir $(LIBPATH_ROOT); fi;
        (cd $(MODELPATH_ROOT)/$(MODEL_ATSLAVE);     ¥
         $(MAKE) -f Makefile                        ¥
         MODEL="$(MODEL_ATSLAVE)"                   ¥
         PROJ_HOME="$(PROJ_HOME)"                   ¥
         LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATSLAVE)“ ¥
         SYSTEMC_HOME="$(SYSTEMC_HOME)"             ¥
         TARGET_ARCH="$(TARGET_ARCH)"               ¥
         MAKE="$(MAKE)"                             ¥
         CXX="$(CXX)"
         OPTFLAG="$(OPTFLAG)“
         DEFFLAG="$(DEFFLAG)"                       ¥
         DBGFLAG="$(DBGFLAG)"                       ¥
         INCPATH="$(INCPATH)"                       ¥
        )
        (cd $(MODELPATH_ROOT)/$(MODEL_TIMER);       ¥
         $(MAKE) -f Makefile                        ¥
         MODEL="$(MODEL_TIMER)"                     ¥
         PROJ_HOME="$(PROJ_HOME)“                   ¥
         LIBPATH="$(LIBPATH_ROOT)/$(MODEL_TIMER)“ ¥
         SYSTEMC_HOME="$(SYSTEMC_HOME)"             ¥
         TARGET_ARCH="$(TARGET_ARCH)“               ¥
         MAKE="$(MAKE)"                             ¥
         CXX="$(CXX)"
         OPTFLAG="$(OPTFLAG)"
         DEFFLAG="$(DEFFLAG)"
         DBGFLAG="$(DBGFLAG)"                       ¥
         INCPATH="$(INCPATH)"                       ¥
        )
        ($(MAKE) -f pltfrm.tb.mk                    ¥
         TARGET="$(TARGET)"                         ¥
         PROJ_HOME="$(PROJ_HOME)"                   ¥
         LIBPATH="$(LIBPATH_ROOT)“                  ¥
         SYSTEMC_HOME="$(SYSTEMC_HOME)"             ¥
         TARGET_ARCH="$(TARGET_ARCH)"               ¥
         SYSTEMC_LIB="$(SYSTEMC_LIB)"               ¥
         MAKE="$(MAKE)"                             ¥
         CXX="$(CXX)"                               ¥
         OPTFLAG="$(OPTFLAG)"                       ¥
         DEFFLAG="$(DEFFLAG)"
         DBGFLAG="$(DBGFLAG)"
         INCPATH="$(INCPATH)"
        )
```

Call the Makefile for each IP

Compilation of ATSLAVE

Compilation of Timer

Compile with pltfrm.tb.mk (Explain one after another on the page).

Sample : ITintv1m (Linux)

Platform

pltfrmCompile/ITintv1m/pltfrm.mk (continuation)

```
clean:
        (cd $(MODELPATH_ROOT)/$(MODEL_ATSLAVE);         ¥
         $(MAKE) -f Makefile                            ¥
         LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATSLAVE)"     ¥
        clean )
        (cd $(MODELPATH_ROOT)/$(MODEL_TIMER);           ¥
         $(MAKE) -f Makefile                            ¥
         LIBPATH="$(LIBPATH_ROOT)/$(MODEL_TIMER)“       ¥
        clean )
        ($(MAKE) -f pltfrm.tb.mk                        ¥
         TARGET="$(TARGET)"                             ¥
         PROJ_HOME="$(PROJ_HOME)"                       ¥
         LIBPATH="$(LIBPATH_ROOT)"                      ¥
         SYSTEMC_HOME="$(SYSTEMC_HOME)"                 ¥
         TARGET_ARCH="$(TARGET_ARCH)"                   ¥
         SYSTEMC_LIB="$(SYSTEMC_LIB)"                   ¥
         MAKE="$(MAKE)"                                 ¥
         CXX="$(CXX)"                                   ¥
         OPTFLAG="$(OPTFLAG)"                           ¥
         DEFFLAG="$(DEFFLAG)"                           ¥
         DBGFLAG="$(DBGFLAG)"                           ¥
         INCPATH="$(INCPATH)"                           ¥
        clean )
#       @($(RM) $(RM_OPT) $(OBJS) $(TARGET) core*);
#       @(if [ -d $(LIBPATH_ROOT) ] ; then $(RM) -r $(RM_OPT)
        $(LIBPATH_ROOT) ; fi;)
```

```
version:
        @echo "#### build machine and OS version"
        @uname -a
        @echo " "
        @echo "#### g++ version"
        @$(CXX) --version
        @echo " "
        @echo "#### gmake version"
        @$(MAKE) --version
        @echo " "
        @echo "#### OSCI version"
        @(strings $(SYSTEMC_LIBPATH)/lib$(SYSTEMC_LIB).a |
        grep "SystemC" )
        @echo " "
        @echo "#### IP component version"
        @(cvs status | egrep -e "=========" -e "File:" -e
        "revision" )
        @(cd $(MODELPATH_ROOT); cvs status | egrep -e
        "=========" -e "File:" -e "revision" )
```

Sample : ITintv1m (Linux)

Platform

Continuation of creation of makefile (for Linux)

The red-letter part: User writing part when user newly make the environment.

pltfrmCompile/ITintv1m/pltfrm.tb.mk

```
# Name(overwrite from the top make)
TARGET                  ?= sim.x

# Location(overwrite from the top make)
PROJ_HOME               ?= $(shell pwd)/../..
LIBPATH                 ?= $(PROJ_HOME)/lib-models
SIM_KERNEL_POSTFIX      ?=

# Location(for local)
MODEL_HOME              = $(PROJ_HOME)/models
MODEL_ATSLAVE_PATH      = $(MODEL_HOME)/ATSLAVE
MODEL_TIMER_PATH        = $(MODEL_HOME)/TIMER
#
SCHEAP_A = $(PROJ_HOME)/lib/SCHEAP-G4$(SIM_KERNEL_POSTFIX).a
MODEL_ATSLAVE_A         = $(LIBPATH)/ATSLAVE/ATSLAVE.a
MODEL_TIMER_A           = $(LIBPATH)/TIMER/TIMER.a

# SystemC location and architecture(overwrite from the top make)
ifeq "$(shell uname -n)" "sdlpc567"
 SYSTEMC_HOME           ?= /eda_tools/systemc
else
 SYSTEMC_HOME           ?= /home/product/systemc/tools/systemc
endif
ifeq "$(shell uname -s)" "SunOS"
 TARGET_ARCH = gccsparcOS5
else
 TARGET_ARCH ?= linux
endif
SYSTEMC_LIB             = systemc
```

Path of user module

User module Archive

```
# SystemC location and (for local)
SYSTEMC_INCPATH         = $(SYSTEMC_HOME)/include
SYSTEMC_LIBPATH = $(SYSTEMC_HOME)/lib-$(TARGET_ARCH)

# Make command(overwrite from the top make)
MAKE                    ?= gmake

# other command(for local)
RM                      = rm
                        = -f

# Linux RedHat7.3 - gcc2.96(overwrite from the top make)
CXX             ?= /usr/bin/g++
OPTFLAG         = -O3 -m32
                  __ -DSC_USE_SC_STRING_OLD ¥
                  NAMIC_PROCESSES ¥
                  -DNSMVINTC711_DEF -DCM_REPORT_OUT ¥
                -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 ¥
                -DLINUX
DBGFLAG         = -Wno-deprecated
INCPATH         = -I$(SYSTEMC_INCPATH) -I.
```

Sample : ITintv1m (Linux)

Platform

pltfrmCompile/ITintv1m/pltfrm.tb.mk (continuation)

```
# Linux RedHat7.3 - gcc2.96(for local)
DEPFLAG                         = -MM
DEFFLAG_MINE     = -D_V850E2R_LOCAL_BUS_ -DLINUX -DV850E2 ¥
                  -D__FLASH_CACHE_INTERNAL__
DBGFLAG_MINE =
INCPATH_MINE = -I$(MODEL_ATSLAVE_PATH) ¥
               -I$(MODEL_TIMER_PATH)
CXXFLAGS       = $(OPTFLAG) $(DEFFLAG) $(DEFFLA          ) ¥
         $(DBGFLAG) $(DBGFLAG_MINE) $(INCPA
#LFLAGS = -ldl -L$(SYSTEMC_LIBPATH) -l$(SYST
LFLAGS          ?= -L$(SYSTEMC_LIBPATH) -l$(SYSTEMC_LIB)

# Files(for local)
OBJECTS                         = pltfrm$(SIM_KERNEL_POSTFIX).o
LIBS                            = $(SCHEAP_A) ¥
                                  $(MODEL_ATSLAVE_A) ¥
                                  $(MODEL_TIMER_A)

# Build rules
.PHONY: all clean
all:      $(TARGET)

$(TARGET): $(OBJECTS) $(LIBS)
        $(CXX) $(CXXFLAGS) -o $@ -rdynamic $(OBJECTS) -ldl ¥
        ?lpthread -W1,-whole-archive $(LIBS) $(LFLAGS) -W1, ¥
        -no-whole-archive
        @echo "Done"

pltfrm$(SIM_KERNEL_POSTFIX).o: pltfrm.cpp
        $(CXX) -c $(CXXFLAGS) -o pltfrm$(SIM_KERNEL_POSTFIX).o ¥
        pltfrm.cpp
```

```
clean:
        @($(RM) $(RM_OPT) $(OBJECTS) $(TARGET) core*);
#       @(if [ -d $(LIBPATH) ] ; then $(RM) -r $(RM_OPT)
        $(LIBPATH) ; fi;)

####### Dependencies
./pltfrm.o: ¥
        $(MODEL_ATSLAVE_PATH)/ATSLAVE.h ¥
        $(MODEL_TIMER_PATH)/TIMER.h
```

Path of user module

Header file of user IP

User module
Archive

Sample : ITintv1m (Windows)

Platform

When SystemC model is executed with Multi debugger

- Execution of MULTI and SystemC model
  - % run_multi_win.bat
  - Execute run_multi_win.bat which is located in pltfrmCompile/build/ITintv1m/ from command prompt.
  - After MULTI logo is displayed, simulator is invoked and connected to MULTI debugger.
- Operation of MULTI debugger
  - Execute the program till last by clicking GO button after break point setting.
  - Close the windows for MULTI when simulation is finished.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITintv1m (Windows)

Platform

When SystemC model is executed without debugger

- Execute SystemC model
  - % run_core_win.bat
  - Execute run_core_win.bat which is located in pltfrmCompile/build/ITintv1m/ from command prompt.
    Do the execution until the execution cycle number specified in this script.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITintv1m (Linux)

Platform

- When SystemC model is executed with Multi debugger

- Execution of MULTI and SystemC model
  - % run_multi.csh
  - Execute run_multi.csh which is located in pltfrmCompile/build/ITintv1m/ from command prompt.
  - After MULTI logo is displayed, simulator is invoked and connected to MULTI debugger.
- Operation of MULTI debugger
  - Execute the program till last by clicking GO button after break point setting.
  - Close the windows for MULTI when simulation is finished.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITintv1m (Linux)

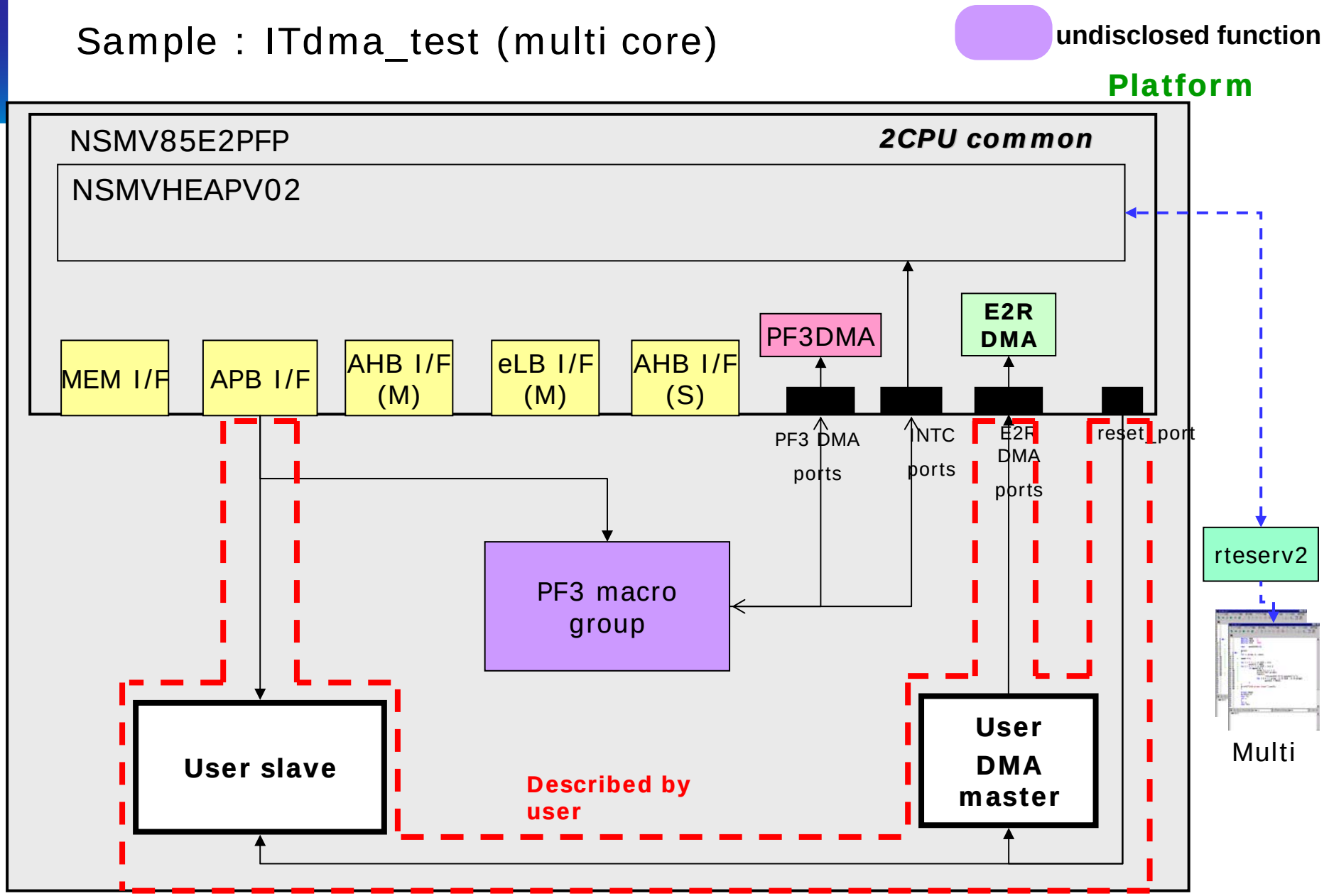
Platform

When SystemC model is executed without debugger

- Execute SystemC model
  - % run_core.csh
  - Execute run_core.csh which is located in pltfrmCompile/build/ITintv1m/ from command prompt. Do the execution until the execution cycle number specified in this script.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITdma_test (multi core)



The feature: The DMA master is connected.

Sample : ITdma_test (Windows)

Platform

Directory configuration (for Windows)

The red-letter part: User writing part when user newly make the environment.

```
pltfrmCompile
 |
 |- build
 |   |-- ITdma_test          Simulator building environment      pltfrm.cpp, pltfrm.h
 |                           Simulator which connected memory IP and DMA master.
 |                           ITdma_test execution environment heap.cfg, *.map, *.bat
 |
 |- include                  Header file
 |
 |- lib                      Archive of SC-HEAP      SCHEAP-G4.lib
 |
 |- multi                    Debugging server storage directory
 |
 |- soft                     Sample program
 |  |- dma_test              DMA transfer software
 |
 |- models
    |- DMA_MASTER            User DMA master
    |- ATSLAVE               User memory
    |- tlm                   OSCI TLM 2.0
```

Sample : ITdma_test (Linux)

Platform

The red-letter part: User writing part when user newly make the environment.

Directory configuration (for Linux)

```
pltfrmCompile
 |
 |- build
 |   |-- ITdma_test          Simulator making environment      pltfrm.cpp, pltfrm.h
 |                           Simulator which connected memory IP with DMA master.
 |                           ITdma_test execution environment heap.cfg, *.map, *.csh
 |
 |- include                  Header file
 |
 |- lib                      Archive of SC-HEAP      SCHEAP-G4.a
 |
 |- lib-modelsO3             Storage place of objectfile (*.o)
 |
 |- multi                    Debugging server storage directory
 |
 |- soft                     Sample program
 |  |- dma_test              DMA transfer software
 |
 |- models
    |- DMA_MASTER            User DMA master
    |- ATSLAVE               User slave
    |- tlm                   OSCI TLM 2.0
```

Sample : ITdma_test

Platform

User-written part

- Platform connection
  build/ITdma_test/pltfrm.cpp, pltfrm.h
- User IP
  models/DMA_MASTER
  models/ATSLAVE (memory)

Sample : ITdma_test

Platform

Simulator executable file building procedure

[Windows]

- Build in pltfrmCompile/build/ITdma_test/msvc80.
  - Execute scheap.sln.
    - Use Release as the solution configuration.
    - Executable file (sim.exe) is created in pltfrmCompile/build/ITdma_test/msvc80.

[Linux]

- Build in pltfrmCompile/build/ITdma_test.
  - Execute make.
    - pltfrm.tb.mk is called from pltfrm.mk and pltfrm.mk from Makefile.
    - Executable file (sim.x) is created in pltfrmCompile/build/ITdma_test.

Sample : ITdma_test

User code

pltfrm.cpp

```
//************** User include header **************//
#include "pltfrm.h“

//************** User include header **************//
void pltfrm(NSMV85E2SPFPV01 *E2SPFP, PF3 *PF3,
 const char *config_file=NULL)
{
//************** User code **************//
//----- instance definition
 ATSLAVE *ATSLAVE_1, *ATSLAVE_2;
 DMA_MASTER *DMA_MASTER_1, *DM
//----- instantiation
 ATSLAVE_1 = new ATSLAVE(“ATSLAVE
 ATSLAVE_2 = new ATSLAVE(“ATSLAVE
 DMA_MASTER_1 = new DMA_MASTER("DMA_MASTER_1",...);
  DMA_MASTER_2 = new DMA_MASTER("DMA_MASTER_2“,...);
//----- connection for E2SPFP
  E2SPFP->lowconst_port(lowconst_sig);
  E2SPFP->APB32IF_i_socket.bind(ATSLAVE_1->target_socket);
  E2SPFP->APB32IF_i_socket.bind(ATSLAVE_2->target_socket);
//----- connection for DMA of E2SPFP
  (*(E2SPFP->dmanmi))(lowconst_sig);
 (*((E2SPFP->dmarq)[0][0]))(dmarq_sig[0]);
 (*((E2SPFP->intio)[0][0]))(lowconst_sig[0]);
  (*((E2SPFP->dmaak)[0][0]))(dmaak_sig[0]);
 (*((E2SPFP->dmach)[0][0]))(dmach_sig[0]);
 (*((E2SPFP->dmalast)[0][0]))(dmalast_sig[0]);
 (*((E2SPFP->dmarq)[0][1]))(dmarq_sig[1]);
 (*((E2SPFP->intio)[0][1]))(lowconst_sig[1]);
 (*((E2SPFP->dmaak)[0][1]))(dmaak_sig[1]);
 (*((E2SPFP->dmach)[0][1]))(dmach_sig[1]);
 (*((E2SPFP->dmalast)[0][1]))(dmalast_sig[1]);
 for(char i=2; i<DMA_CHANNEL_NMB; i++){
 (*((E2SPFP->dmarq)[0][i]))(dmarq_sig[i]);
 (*((E2SPFP->intio)[0][i]))(lowconst_sig[i]);
 (*((E2SPFP->dmaak)[0][i]))(dmaak_sig[i]);
 (*((E2SPFP->dmach)[0][i]))(dmach_sig[i]);
 (*((E2SPFP->dmalast)[0][i]))(dmalast_sig[i]);
 }
```

The pointer of the PF3 macro group is passed.
The user is made to be able to do connected description of the PF3 macro in the future with this pointer.

It connects it through OSCI TLM.

Platform

```
//----- connection for DMA_MSTR
 DMA_MSTR_1->reset(E2SPFP->reset_port);
 DMA_MSTR_1->dmarq0(dmarq_sig[0]);
 DMA_MSTR_1->dmaak0(dmaak_sig[0]);
 DMA_MSTR_1->dmach0(dmach_sig[0]);
 DMA_MSTR_1->dmalast0(dmalast_sig[0]);
 DMA_MSTR_2->reset(E2SPFP->reset_port);
 DMA_MSTR_2->dmarq1(dmarq_sig[1]);
 DMA_MSTR_2->dmaak1(dmaak_sig[1]);
 DMA_MSTR_2->dmach1(dmach_sig[1]);
 DMA_MSTR_2->dmalast1(dmalast_sig[1]);
 ----- connection for ATSLAVE_1
 TSLAVE_1->reset_port(E2SPFP->reset_port);
 ----- connection for ATSLAVE_2
 ATSLAVE_2->reset_port(E2SPFP->reset_port);
//************** User code **************//
}

void pltfrmFC(NSMV85E2SPFPV01 *E2SPFP,
 const char *config_file=NULL)
{

}

void pltfrmFCDelete()
{
}

void pltfrmDelete()
{
}
```

The clock and reset are direct connections of the port.

The DMA master and the DMA control terminal are connected.

The connection is described in pltfrm() for the connection to the port where PF3 peripheral macro is not connected. To connect user macro higher-priority than PF3 peripheral macro, user describes it in pltfrmFC().
Moreover, after the simulation, if user wants to delete explicitly the instance which is instantiated in pltfrm() and pltfrmFC(), user can delete it in pltfrmDelete() and pltfrmFCDelete().

Sample : ITdma_test

Platform

User code

pltfrm.h

```
#ifndef PLTFRM_H
#define PLTFRM_H

#include “NSMV85E2SPFPV01.h"
//************* User include header *************//
#include “DMA_MASTER.h"
#include “ATSLAVE.h"
#include “PF3.h"
//************* User include header *************//

// User definition
#define DMA_CHANNEL_NMB 8
// signal defined
sc_signal<bool> lowconst_sig;          // Signal for low clamping
sc_signal<bool> dmarq_sig[2];          // Signal for DMA
sc_signal<bool> dmaak_sig[2];          //
sc_signal<bool> dmach_sig[2];          //
sc_signal<bool> dmalast_sig[2];        //

//************* User code *************//
typedef unsigned int ADDRESS_TYPE;
typedef unsigned char DATA_TYPE;
//************* User code *************//

#endif // PLTFRM_H
```

Sample : ITdma_test (Windows)

Creation of scheap.sln (for Windows)

Refer to pltfrmCompile/build/ITdma_test/msvc80/scheap.sln.

Sample : ITdma_test (Linux)

Platform

The red-letter part: User writing part when user newly make the environment.

Creation of makefile (for Linux)

pltfrmCompile/ITdma_test/pltfrm.mk

```
# Target selection variables
MODE ?= release
KERNEL ?= systemc
LIBTYPE ?= static

# Name
MODEL                   =
TARGET                  = sim.x

# Models
MODEL_ATSLAVE       = ATSLAVE
MODEL_DMA_MASTER    = DMA_MASTER

# Location
PROJ_HOME               ?= $(shell pwd)/../..
MODELPATH_ROOT      = $(PROJ_HOME)/models
INCLUDE_HEADER      = $(PROJ_HOME)/include
LIBPATH_ROOT        ?= $(PROJ_HOME)/lib-modelsO3
ifdef TLM_INC_DIR
TLM_INCLUDE_HEADER ?= ${TLM_INC_DIR}
else
TLM_INCLUDE_HEADER  ?= $(PROJ_HOME)/../pltfrmCompile/models/tlm
endif
```

User module

```
# SystemC Location and architecture
ifeq "$(shell uname -n)" "sdlpc567"
 SYSTEMC_HOME       ?= /eda_tools/systemc
else
 SYSTEMC_HOME ?=
                    /home/product/systemc/tools/systemc
endif
ifeq "$(shell uname -s)"
 TARGET_ARCH
else
 TARGET_ARCH        = linux
endif
SYSTEMC_INCPATH     = $(SYSTEMC_HOME)/include
SYSTEMC_LIBPATH = $(SYSTEMC_HOME)/lib-$(TARGET_ARCH)
SYSTEMC_LIB         ?= systemc

# Make command
MAKE                ?= /usr/bin/gmake
override MAKEFLAGS  += --no-print-directory

# RM command
RM                  = rm
RM_OPT              = -f
```

Path of OSCI SystemC library

Sample : ITdma_test (Linux)

Platform

pltfrmCompile/ITdma_test/pltfrm.mk (continuation)

```
# Linux RedHat7.3 - gcc2.96
CXX ?= /usr/bin/g++
DEFFLAG = -D__SYSTEMC_21__ -DSC_USE_SC_STRING_OLD ¥
          -DSC_INCLUDE_DYNAMIC_PROCESSES ¥
          -DNSMVINTC711_DEF -DCM_REPORT_OUT ¥
          -D_LARGEFILE64_SOURCE ¥
          -D_FILE_OFFSET_BITS=64 -DLINUX -DPLTFRM_DEF
ifeq ($(MODE),debug)
OPTFLAG      = -O0 -m32
DBGFLAG      = -ggdb3 -DDEBUG
else
OPTFLAG      = -O3 -m32
DBGFLAG      = -DNDEBUG
endif
INCPATH = -I$(SYSTEMC_INCPATH) -I. -L$(SYSTEMC_LIBPATH) ¥
          -I$(INCLUDE_HEADER) -I$(TLM_INCLUDE_HEADER)

# build rules
all:
    if test ! -d $(LIBPATH_ROOT); then mkdir $(LIBPATH
    (cd $(MODELPATH_ROOT)/$(MODEL_ATSLAVE);
     $(MAKE) -f Makefile                          ¥
     MODEL="$(MODEL_ATSLAVE)"                     ¥
     PROJ_HOME="$(PROJ_HOME)"                     ¥
     LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATSLAVE)“ ¥
     SYSTEMC_HOME="$(SYSTEMC_HOME)"               ¥
     TARGET_ARCH="$(TARGET_ARCH)"                 ¥
     MAKE="$(MAKE)"                               ¥
     CXX="$(CXX)"                                 ¥
     OPTFLAG="$(OPTFLAG)"
     DEFFLAG="$(DEFFLAG)"
     DBGFLAG="$(DBGFLAG)"                         ¥
     INCPATH="$(INCPATH)“                         ¥
    )
    (cd $(MODELPATH_ROOT)/$(MODEL_DMA_MASTER); ¥
     $(MAKE) -f Makefile                          ¥
     MODEL="$(MODEL_DMA_MASTER)"                  ¥
     PROJ_HOME="$(PROJ_HOME)"                     ¥
     LIBPATH="$(LIBPATH_ROOT)/$(MODEL_DMA_MASTER)“ ¥
     SYSTEMC_HOME="$(SYSTEMC_HOME)"               ¥
     TARGET_ARCH="$(TARGET_ARCH)"                 ¥
     MAKE="$(MAKE)"
     CXX="$(CXX)"
     OPTFLAG="$(OPTFLAG)"                         ¥
     DEFFLAG="$(DEFFLAG)"                         ¥
     DBGFLAG="$(DBGFLAG)"                         ¥
     INCPATH="$(INCPATH)“                         ¥
    )
    $(MAKE) -f pltfrm.tb.mk                       ¥
    TARGET="$(TARGET)"                            ¥
    PROJ_HOME="$(PROJ_HOME)"                      ¥
    LIBPATH="$(LIBPATH_ROOT)"                     ¥
    SYSTEMC_HOME="$(SYSTEMC_HOME)"                ¥
    TARGET_ARCH="$(TARGET_ARCH)"                  ¥
    SYSTEMC_LIB="$(SYSTEMC_LIB)"                  ¥
    MAKE="$(MAKE)"                                ¥
    CXX="$(CXX)"                                  ¥
    OPTFLAG="$(OPTFLAG)"                          ¥
    DEFFLAG="$(DEFFLAG)"                          ¥
    DBGFLAG="$(DBGFLAG)"                          ¥
    INCPATH="$(INCPATH)"
    )
```

Call Makefile for each IP

Compilation of DMA_MASTER

Compilation of ATSLAVE

Compiles with pltfrm.tb.mk (Explain one after another on the page).

Sample : ITdma_test (Linux)

Platform

pltfrmCompile/ITdma_test/pltfrm.mk (continuation)

```
clean:
	(cd $(MODELPATH_ROOT)/$(MODEL_ATSLAVE);          ¥
	 $(MAKE) -f Makefile                             ¥
	 LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATSLAVE)"      ¥
	clean )
	(cd $(MODELPATH_ROOT)/$(MODEL_DMA_MASTER);       ¥
	 $(MAKE) -f Makefile                             ¥
	 LIBPATH="$(LIBPATH_ROOT)/$(MODEL_DMA_MASTER)"   ¥
	clean )
	($(MAKE) -f pltfrm.tb.mk                         ¥
	 TARGET="$(TARGET)"                              ¥
	 PROJ_HOME="$(PROJ_HOME)"                        ¥
	 LIBPATH="$(LIBPATH_ROOT)"                       ¥
	 SYSTEMC_HOME="$(SYSTEMC_HOME)"                  ¥
	 TARGET_ARCH="$(TARGET_ARCH)"                    ¥
	 SYSTEMC_LIB="$(SYSTEMC_LIB)"                    ¥
	 MAKE="$(MAKE)"                                  ¥
	 CXX="$(CXX)"                                    ¥
	 OPTFLAG="$(OPTFLAG)"                            ¥
	 DEFFLAG="$(DEFFLAG)"                            ¥
	 DBGFLAG="$(DBGFLAG)"                            ¥
	 INCPATH="$(INCPATH)"                            ¥
	clean )
#	@($(RM) $(RM_OPT) $(OBJS) $(TARGET) core*);
#	@(if [ -d $(LIBPATH_ROOT) ] ; then $(RM) -r $(RM_OPT)
	$(LIBPATH_ROOT) ; fi;)
```

```
version:
	@echo "#### build machine and OS version"
	@uname -a
	@echo " "
	@echo "#### g++ version"
	@$(CXX) --version
	@echo " "
	@echo "#### gmake version"
	@$(MAKE) --version
	@echo " "
	@echo "#### OSCI version"
	@(strings $(SYSTEMC_LIBPATH)/lib$(SYSTEMC_LIB).a | grep
	"SystemC" )
	@echo " "
	@echo "#### IP component version"
	@(cvs status | egrep -e "=========" -e "File:" -e "revision" )
	@(cd $(MODELPATH_ROOT); cvs status | egrep -e
	"=========" -e "File:" -e "revision" )
```

Sample : ITdma_test (Linux)

Platform

Continuation of creation of makefile (for Linux)

The red-letter part: User writing part when user newly make the environment.

pltfrmCompile/ITdma_test/pltfrm.tb.mk

```
# Name(overwrite from the top make)
TARGET                                  ?= sim.x

# Location(overwrite from the top make)
PROJ_HOME                       ?= $(shell pwd)/../..
LIBPATH                         ?= $(PROJ_HOME)/lib-models
SIM_KERNEL_POSTFIX              ?=

# Location(for local)
MODEL_HOME                      = $(PROJ_HOME)/models
MODEL_ATSLAVE_PATH      = $(MODEL_HOME)/ATSLAVE
MODEL_DMA_MASTER_PATH  = $(MODEL_HOME)/DMA_MASTER
#
SCHEAP_A    = $(PROJ_HOME)/lib/SCHEAP-G4$(SIM_KERNEL_POSTFIX).a
MODEL_ATSLAVE_A         = $(LIBPATH)/ATSLAVE/ATSLAVE.a
MODEL_DMA_MASTER_A  =
                $(LIBPATH)/DMA_MASTER/DMA_MASTER.a

# SystemC location and architecture(overwrite from the top make)
ifeq "$(shell uname -n)" "sdlpc567"
 SYSTEMC_HOME                   ?= /eda_tools/systemc
else
 SYSTEMC_HOME                   ?= /home/product/systemc/tools/systemc
endif
ifeq "$(shell uname -s)" "SunOS"
 TARGET_ARCH = gccsparcOS5
else
 TARGET_ARCH ?= linux
endif
SYSTEMC_LIB                     = systemc
```

Path of user module

User module Archive

```
# SystemC location and (for local)
SYSTEMC_INCPATH                 = $(SYSTEMC_HOME)/include
SYSTEMC_LIBPATH    = $(SYSTEMC_HOME)/lib-$(TARGET_ARCH)

# Make command(overwrite from the top make)
MAKE            ?= gmake

# other command(for local)
RM              = rm
RM_OPT          = -f

                                        96(overwrite from the top make)
CXX         ?= /usr/bin/g++
OPTFLAG    = -O3 -m32
DEFFLAG = -D__SYSTEMC_21__ -DSC_USE_SC_STRING_OLD ¥
          -DSC_INCLUDE_DYNAMIC_PROCESSES ¥
          -DNSMVINTC711_DEF -DCM_REPORT_OUT ¥
          -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 ¥
           -DLINUX
                                        Wno-deprecated
                                        $(SYSTEMC_INCPATH) -I.
```

Sample : ITdma_test (Linux)

Platform

PltfrmCompile/ITdma_test/pltfrm.tb.mk (continuation)

```
# Linux RedHat7.3 - gcc2.96(for local)
DEPFLAG         = -MM
DEFFLAG_MINE = -D_V850E2R_LOCAL_BUS_ -DLINUX ¥
               -DV850E2 -D__FLASH_CACHE_INTERNAL__
DBGFLAG_MINE                    =
INCPATH_MINE = -I$(MODEL_ATSLAVE_PATH) ¥
               -I$(MODEL_DMA_MASTER_PATH)
CXXFLAGS = $(OPTFLAG) $(DEFFLAG) $(DEFFLAG_MINE) ¥
           $(DBGFLAG) $(DBGFLAG_MINE) $(INCPATH) ¥
           $(INCPATH_MINE)
#LFLAGS = -ldl -L$(SYSTEMC_LIBPATH) -l$(SYSTEMC_LIB) ¥
           -dynamic
LFLAGS      ?= -L$(SYSTEMC_LIBPATH) -l$(SYSTEMC_LIB)

# Files(for local)
OBJECTS         = pltfrm$(SIM_KERNEL_POSTFIX).o
LIBS                            = $(SCHEAP_A) ¥
                                  $(MODEL_ATSLAVE_A) ¥
                                  $(MODEL_DMA_MASTER_A)

# Build rules
.PHONY: all clean
all:            $(TARGET)

$(TARGET): $(OBJECTS) $(LIBS)
           $(CXX) $(CXXFLAGS) -o $@ -rdynamic $(OBJECTS) -ldl
           -lpthread -W1,-whole-archive $(LIBS) $(LFLAGS) -W1,-
           no-whole-archive
           @echo "Done"
```

Path of user module

User module Archive

```
pltfrm$(SIM_KERNEL_POSTFIX).o: pltfrm.cpp
                $(CXX) -c $(CXXFLAGS) -o
pltfrm$(SIM_KERNEL_POSTFIX).o pltfrm.cpp

clean:
                @($(RM) $(RM_OPT) $(OBJECTS) $(TARGET) core*);
#               @(if [ -d $(LIBPATH) ] ; then $(RM) -r $(RM_OPT)
$(LIBPATH) ; fi;)

####### Dependencies
./pltfrm.o: ¥
                $(MODEL_ATSLAVE_PATH)/ATSLAVE.h ¥
                $(MODEL_DMA_MASTER_PATH)/DMA_MASTER.h
```

Header file of user IP

Sample : ITdma_test (Windows)

Platform

- When SystemC model is executed with Multi debugger

- Execution of MULTI and SystemC model
  - % run_multi_win.bat
  - Execute run_multi_win.bat which is located in pltfrmCompile/build/ITdma_test/ from command prompt.
  - After MULTI logo is displayed, simulator is invoked and connected to MULTI debugger.
- Operation of MULTI debugger
  - Execute the program till last by clicking GO button after break point setting.
  - Close the windows for MULTI when simulation is finished.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITdma_test (Windows)

Platform

When SystemC model is executed without debugger

- Execute SystemC model
  - % run_core_win.bat
  - Execute run_core_win.bat which is located in pltfrmCompile/build/ITdma_test/ from command prompt. Do the execution until the execution cycle number specified in this script.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITdma_test (Linux)

Platform

- When SystemC model is executed with Multi debugger

- Execution of MULTI and SystemC model
  - % run_multi.csh
  - Execute run_multi.csh which is located in pltfrmCompile/build/ITdma_test/ from command prompt.
  - After MULTI logo is displayed, simulator is invoked and connected to MULTI debugger.
- Operation of MULTI debugger
  - Execute the program till last by clicking GO button after break point setting.
  - Close the windows for MULTI when simulation is finished.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITdma_test (Linux)

Platform

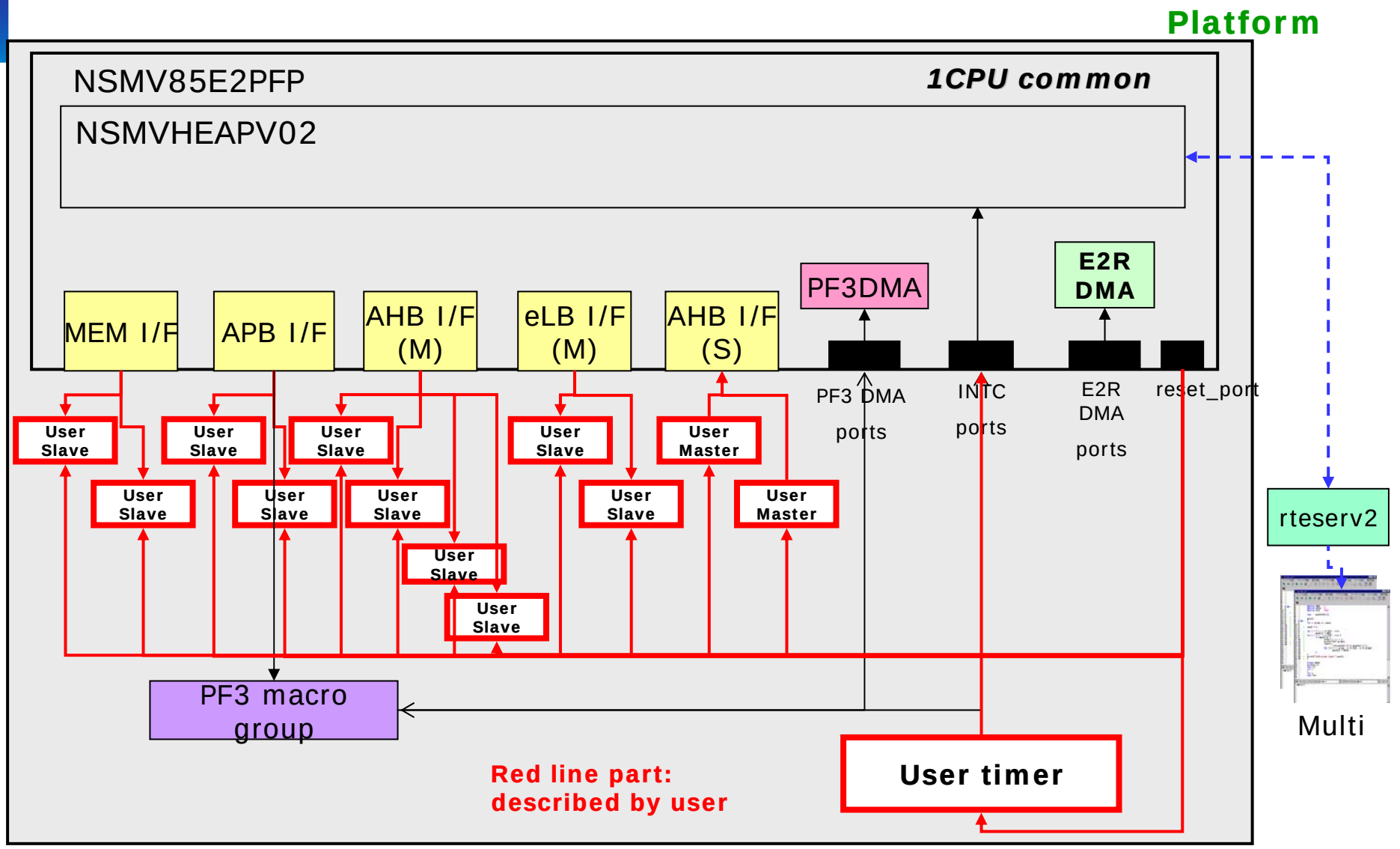
When SystemC model is executed without debugger

- Execute SystemC model
  - % run_core.csh
  - Execute run_core.csh which is located in pltfrmCompile/build/ITdma_test/ from command prompt. Do the execution until the execution cycle number specified in this script.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITextAccess1cpu (single core)

undisclosed function



The feature: The user master is connected.

Sample : ITextAccess1cpu (Windows)

Platform

The red-letter part: User writing part when user newly make the environment.

Directory configuration (for Windows)

```
pltfrmCompile
 |
 |- build
 |   |-- ITextAccess1cpu          Simulator building environment      pltfrm.cpp, pltfrm.h
 |                                Simulator which connected memory, timer and master IP.
 |                                ITextAccess1cpu execution environment heap.cfg, *.map, and *.bat
 |
 |- include                 Header file
 |
 |- lib                     Archive of SC-HEAP      SCHEAP-G4.lib
 |
 |- multi                   Debugging server storage directory
 |
 |- soft                    Sample program
 |   |- external_access     Use software
 |
 |- models
    |- TIMER                User timer
    |- ATSLAVE              User slave
    |- ATMASTER             User master
    |- tlm                  OSCI TLM 2.0
```

Sample : ITextAccess1cpu (Linux)

Platform

Directory configuration (for Linux)

The red-letter part: User writing part when user newly make the environment.

```
pltfrmCompile
 |
 |- build
 |   |-- ITextAccess1cpu          Simulator making environment      pltfrm.cpp, pltfrm.h
 |                                Simulator that connected memory slave IP with user timer.
 |                                ITextAccess1cpu execution environment heap.cfg, *.map, and *.csh
 |
 |- include                 Header file
 |
 |- lib                     Archive of SC-HEAP      SCHEAP-G4.a
 |
 |- lib-modelsO3            Objectfile (*.o) storage place
 |
 |- multi                   Debugging server storage directory
 |
 |- soft                    Sample program
 |  |- external_access      Use software
 |
 |- models
    |- TIMER                User timer
    |- ATSLAVE              User slave
    |- ATMASTER             User master
    |- tlm                  OSCI TLM 2.0
```

Sample : ITextAccess1cpu

Platform

User-written part

- Platform connection

  build/ITextAccess1cpu/pltfrm.cpp, pltfrm.h

- User IP

  models/TIMER

  models/ATSLAVE (memory)

  models/ATMASTER (master IP)

Sample : ITextAccess1cpu

Simulator executable file building procedure

[Windows]

- Build in pltfrmCompile/build/ITextAccess1cpu/msvc80.
  - Execute scheap.sln.
    - Use Release as the solution configuration.
    - Executable file (sim.exe) is created in pltfrmCompile/build/ITextAccess1cpu/msvc80.

[Linux]

- Build in pltfrmCompile/build/ITextAccess1cpu.
  - Execute make.
    - pltfrm.tb.mk is called from pltfrm.mk and pltfrm.mk from Makefile.
    - Executable file (sim.x) is created in pltfrmCompile/build/ITextAccess1cpu.

Sample : ITextAccess1cpu

User code

pltfrm.cpp

```
//************** User include header **************//
#include "pltfrm.h"

//************** User include header **************//
void pltfrm(NSMV85E2SPFPV01 *E2SPFP, PF3 *PF3,
 const char *config_file=NULL)
{
 //************** User code **************
 //----- instance definition
 ATSLAVE *ATSLAVE_1, *ATSLAVE_2,
 *ATSLAVE_5, *ATSLAVE_6, *ATSLAVE
 *ATSLAVE_9, *ATSLAVE_10;
 TIMER *TIMER_1;
 ATMASTER *ATMASTER_1, *ATMASTER_2;

 //----- instantiation
 ATSLAVE_1 = new ATSLAVE("ATSLAVE_1",...);
 :
 ATSLAVE_10 = new ATSLAVE("ATSLAVE_10",...);
 TIMER_1 = new TIMER("TIMER_1",...);
 ATMASTER_1 = new ATMASTER("ATMASTER_1",...);
 ATMASTER_2 = new ATMASTER("ATMASTER_2",...);

 //----- connection
 E2SPFP->APB32IF_i_socket.bind(ATSLAVE_1->target_socket);
 E2SPFP->APB32IF_i_socket.bind(ATSLAVE_2->target_socket);
 E2SPFP->AHBSIF_i_socket.bind(ATSLAVE_3->target_socket);
 E2SPFP->AHBSIF_i_socket.bind(ATSLAVE_4->target_socket);
 E2SPFP->AHBSIF_i_socket.bind(ATSLAVE_5->target_socket);
 E2SPFP->AHBSIF_i_socket.bind(ATSLAVE_6->target_socket);
 E2SPFP->ELBIF_i_socket.bind(ATSLAVE_7->target_socket);
 E2SPFP->ELBIF_i_socket.bind(ATSLAVE_8->target_socket);
 E2SPFP->MEMIF_i_socket.bind(ATSLAVE_9->target_socket);
 E2SPFP->MEMIF_i_socket.bind(ATSLAVE_10->target_socket);

 ATMASTER_1->initiator_socket.bind(E2SPFP->AHBMIF_t_socket;
 ATMASTER_2->initiator_socket.bind(E2SPFP->AHBMIF_t_socket;
```

The pointer of the PF3 macro group is passed.
The user is made to be able to do connected description of the PF3 macro in the future with this pointer.

Platform

```
//----- connection to ATSLAVE
ATSLAVE_1->reset_port(E2SPFP->reset_port);
:
ATSLAVE_10->reset_port(E2SPFP->reset_port);

//----- connection to ATMASTER
ATMASTER_1->reset_port(E2SPFP->reset_port);
ATMASTER_2->reset_port(E2SPFP->reset_port);

//----- connection to INTC
E2SPFP->pe1_eiint_port[10](intcmd_sig);
E2SPFP->pe2_eiint_port[11](intcmd_sig);
//---- for timer
TIMER_1->reset(E2SPFP->reset_port);
TIMER_1->intcmd(intcmd_sig);

//************** User code **************//
}

void pltfrmFC(NSMV85E2SPFPV01 *E2SPFP,
 const char *config_file=NULL)
{
}

void pltfrmFCDelete()
{
}

void pltfrmDelete()
{
}
```

To each slave connection I/F
The user slave is connected.

To master I/F
Connection of user master

The connection is described in pltfrm() for the connection to the port where PF3 peripheral macro is not connected. To connect user macro higher-priority than PF3 peripheral macro, user describes it in pltfrmFC().
Moreover, after the simulation, if user wants to delete explicitly the instance which is instanciated in pltfrm() and pltfrmFC(), user can delete it in pltfrmDelete() and pltfrmFCDelete().

Sample : ITextAccess1cpu

Platform

User code

pltfrm.h

```
#ifndef PLTFRM_H
#define PLTFRM_H

#include “NSMV85E2SPFPV01.h"
//************** User include header **************//
#include “ATMASTER.h"
#include “TIMER.h"
#include “ATSLAVE.h"
#include “PF3.h"
//************** User include header **************//

// signal defined
sc_signal<bool> intcmd_sig;                                    // Signal for timer

typedef unsigned int ADDRESS_TYPE;
typedef unsigned char DATA_TYPE;
//************** User code **************//

#endif // PLTFRM_H
```

Sample : ITextAccess1cpu (Windows)

Creation of scheap.sln (for Windows)

Refer to pltfrmCompile/build/ITextAccess1cpu/msvc80/scheap.sln.

Sample : ITextAccess1cpu (Linux)

Creation of makefile (for Linux)

pltfrmCompile/ITextAccess1cpu/pltfrm.mk

Platform

The red-letter part: User writing part when user newly make the environment.

```
# Target selection variables
MODE                    ?= release
KERNEL                  ?= systemc
LIBTYPE                 ?= static

# Name
MODEL                   =
TARGET                  = sim.x

# Models
MODEL_ATSLAVE           = ATSLAVE
MODEL_TIMER             = TIMER
MODEL_ATMASTER          = ATMASTER

# Location
PROJ_HOME               ?= $(shell pwd)/../..
LIBPATH_ROOT            ?= $(PROJ_HOME)/lib-modelsO3
MODELPATH_ROOT          = $(PROJ_HOME)/models
INCLUDE_HEADER = $(PROJ_HOME)/include
ifdef TLM_INC_DIR
TLM_INCLUDE_HEADER ?= ${TLM_INC_DIR}
else
TLM_INCLUDE_HEADER    ?= $(PROJ_HOME)/../pltfrmCompile/models/tlm
endif
```

User module

```
# SystemC Location and architecture
ifeq "$(shell uname -n)" "sdlpc567"
 SYSTEMC_HOME           ?= /eda_tools/systemc
else
 SYSTEMC_HOME ?=
 /home/product/systemc/tools/systemc
endif
ifeq "$(shell uname -s
 TARGET_ARCH
else
ARGET_ARCH              = linux
ndif
SYSTEMC_INCPATH         = $(SYSTEMC_HOME)/include
SYSTEMC_LIBPATH = $(SYSTEMC_HOME)/lib-$(TARGET_ARCH)
SYSTEMC_LIB             ?= systemc

# Make command
MAKE                    ?= /usr/bin/gmake
override MAKEFLAGS      += --no-print-directory

# RM command
RM                      = rm
RM_OPT                  = -f
```

OS"

Path of OSCI SystemC library

Sample : ITextAccess1cpu (Linux)

pltfrmCompile/ITextAccess1cpu/pltfrm.mk (continuation)

```
# Linux RedHat7.3 - gcc2.96
CXX       ?= /usr/bin/g++
DEFFLAG = -D__SYSTEMC_21__ -DSC_USE_SC_STRING_OLD ¥
            -DSC_INCLUDE_DYNAMIC_PROCESSES ¥
            -DNSMVINTC711_DEF -DCM_REPORT_OUT ¥
            -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 ¥
            -DLINUX -DPLTFRM_DEF
ifeq ($(MODE),debug)
OPTFLAG         = -O0 -m32
DBGFLAG         = -ggdb3 -DDEBUG
else
OPTFLAG         = -O3 -m32
DBGFLAG         = -DNDEBUG
endif
INCPATH   = -I$(SYSTEMC_INCPATH) -I. -L$(SYSTEMC_LIBPATH) ¥
          -I$(INCLUDE_HEADER) -I$(TLM_INCLUDE_HEADER)

# build rules
all:
          if test ! -d $(LIBPATH_ROOT); then mkdir $(LIB
          (cd $(MODELPATH_ROOT)/$(MODEL_ATSLAVE);¥
           $(MAKE) -f Makefile                          ¥
           MODEL="$(MODEL_ATSLAVE)"                     ¥
           PROJ_HOME="$(PROJ_HOME)"                     ¥
           LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATSLAVE)"¥
           SYSTEMC_HOME="$(SYSTEMC_HOME)"               ¥
           TARGET_ARCH="$(TARGET_ARCH)"                 ¥
           MAKE="$(MAKE)"                               ¥
           CXX="$(CXX)"                                 ¥
           OPTFLAG="$(OPTFLAG)"
           DEFFLAG="$(DEFFLAG)"
           DBGFLAG="$(DBGFLAG)"                         ¥
           INCPATH="$(INCPATH)"                         ¥
          )
          (cd $(MODELPATH_ROOT)/$(MODEL_TIMER);         ¥
           $(MAKE) -f Makefile                          ¥
           MODEL="$(MODEL_TIMER)"                       ¥
           PROJ_HOME="$(PROJ_HOME)"                     ¥
           LIBPATH="$(LIBPATH_ROOT)/$(MODEL_TIMER)“¥
           SYSTEMC_HOME="$(SYSTEMC_HOME)"               ¥
           TARGET_ARCH="$(TARGET_ARCH)"  ¥
           MAKE="$(MAKE)"                               ¥
           CXX="$(CXX)"                                 ¥
           OPTFLAG="$(OPTFLAG)"
           DEFFLAG="$(DEFFLAG)"
           DBGFLAG="$(DBGFLAG)"                         ¥
           INCPATH="$(INCPATH)"                         ¥
          )
          (cd $(MODELPATH_ROOT)/$(MODEL_ATMASTER);¥
           $(MAKE) -f Makefile                          ¥
           MODEL="$(MODEL_ATMASTER)"                    ¥
           PROJ_HOME="$(PROJ_HOME)"                     ¥
           LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATMASTER)"¥
           SYSTEMC_HOME="$(SYSTEMC_HOME)"               ¥
           TARGET_ARCH="$(TARGET_ARCH)"                 ¥
           MAKE="$(MAKE)"                               ¥
           CXX="$(CXX)"                                 ¥
           OPTFLAG="$(OPTFLAG)"
           DEFFLAG="$(DEFFLAG)"
           DBGFLAG="$(DBGFLAG)"                         ¥
           INCPATH="$(INCPATH)"                         ¥
          )
```

Platform

Call Makefile for each IP

Compilation of ATSLAVE

Compilation of Timer

Compilation of ATMASTER

Sample : ITextAccess1cpu (Linux)

Platform

pltfrmCompile/ITextAccess1cpu/pltfrm.mk (continuation)

```
        ($(MAKE) -f pltfrm.tb.mk                          ¥
         TARGET="$(TARGET)"                               ¥
         PROJ_HOME="$(PROJ_HOME)"                         ¥
         LIBPATH="$(LIBPATH_ROOT)“                        ¥
         SYSTEMC_HOME="$(SYSTEMC_HOME)"                   ¥
         TARGET_ARCH="$(TARGET_ARCH)"                     ¥
         SYSTEMC_LIB="$(SYSTEMC_LIB)"                     ¥
         MAKE="$(MAKE)"                                   ¥
         CXX="$(CXX)"                                     ¥
         OPTFLAG="$(OPTFLAG)"                             ¥
         DEFFLAG="$(DEFFLAG)"                             ¥
         DBGFLAG="$(DBGFLAG)"                             ¥
         INCPATH="$(INCPATH)"                             ¥
        )

clean:
        (cd $(MODELPATH_ROOT)/$(MODEL_ATSLAVE);           ¥
         $(MAKE) -f Makefile                              ¥
         LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATSLAVE)"       ¥
        clean )
        (cd $(MODELPATH_ROOT)/$(MODEL_TIMER);             ¥
         $(MAKE) -f Makefile                              ¥
         LIBPATH="$(LIBPATH_ROOT)/$(MODEL_TIMER)“         ¥
        clean )
        (cd $(MODELPATH_ROOT)/$(MODEL_ATMASTER);          ¥
         $(MAKE) -f Makefile                              ¥
         LIBPATH="$(LIBPATH_ROOT)/$(MODEL_ATMASTER)"¥
        clean )
        ($(MAKE) -f pltfrm.tb.mk                          ¥
         TARGET="$(TARGET)"                               ¥
         PROJ_HOME="$(PROJ_HOME)"                         ¥
         LIBPATH="$(LIBPATH_ROOT)"                        ¥
         SYSTEMC_HOME="$(SYSTEMC_HOME)"                   ¥
         TARGET_ARCH="$(TARGET_ARCH)"                     ¥
         SYSTEMC_LIB="$(SYSTEMC_LIB)"                     ¥
         MAKE="$(MAKE)"                                   ¥
         CXX="$(CXX)"                                     ¥
         OPTFLAG="$(OPTFLAG)"                             ¥
         DEFFLAG="$(DEFFLAG)"                             ¥
         DBGFLAG="$(DBGFLAG)"                             ¥
         INCPATH="$(INCPATH)"                             ¥
        clean )
#       @($(RM) $(RM_OPT) $(OBJS) $(TARGET) core*);
#       @(if [ -d $(LIBPATH_ROOT) ] ; then $(RM) -r $(RM_OPT)
        $(LIBPATH_ROOT) ; fi;)

version:
        @echo "#### build machine and OS version"
        @uname -a
        @echo " "
        @echo "#### g++ version"
        @$(CXX) --version
        @echo " "
        @echo "#### gmake version"
        @$(MAKE) --version
        @echo " "
        @echo "#### OSCI version"
        @(strings $(SYSTEMC_LIBPATH)/lib$(SYSTEMC_LIB).a |
        grep "SystemC" )
        @echo " "
        @echo "#### IP component version"
        @(cvs status | egrep -e "=========" -e "File:" -e
        "revision" )
        @(cd $(MODELPATH_ROOT); cvs status | egrep -e
        "=========" -e "File:" -e "revision" )
```

It compiles with pltfrm.tb.mk (Explain on the next page).

Sample : ITextAccess1cpu (Linux)

Continuation of creation of makefile (for Linux)

Platform

The red-letter part: User writing part when user newly make the environment.

pltfrmCompile/ITextAccess1cpu/pltfrm.tb.mk

```
# Name(overwrite from the top make)
TARGET                  ?= sim.x

# Location(overwrite from the top make)
PROJ_HOME               ?= $(shell pwd)/../..
LIBPATH                 ?= $(PROJ_HOME)/lib-models
SIM_KERNEL_POSTFIX      ?=

# Location(for local)
MODEL_HOME              = $(PROJ_HOME)/models
MODEL_ATSLAVE_PATH      = $(MODEL_HOME)/ATSLAVE
MODEL_TIMER_PATH        = $(MODEL_HOME)/TIMER
MODEL_ATMASTER_PATH     = $(MODEL_HOME)/ATMASTER
#
SCHEAP_A = $(PROJ_HOME)/lib/SCHEAP-G4$(SIM_KERNEL_POSTFIX).a
MODEL_ATSLAVE_A     = $(LIBPATH)/ATSLAVE/ATSLAVE.a
MODEL_TIMER_A       = $(LIBPATH)/TIMER/TIMER.a
MODEL_ATMASTER_A    = $(LIBPATH)/ATMASTER/ATMASTER.a

# SystemC location and architecture(overwrite from          make)
ifeq "$(shell uname -n)" "sdlpc567"
 SYSTEMC_HOME           ?= /eda_tools/systen
else
 SYSTEMC_HOME           ?= /home/product/systemc/tools/systemc
endif
```

Path of user module

User module Archive

```
ifeq "$(shell uname -s)" "SunOS"
 TARGET_ARCH = gccsparcOS5
else
 TARGET_ARCH ?= linux
endif
SYSTEMC_LIB             = systemc

emC location and (for local)
MC_INCPATH          = $(SYSTEMC_HOME)/include
SYSTEMC_LIBPATH = $(SYSTEMC_HOME)/lib-$(TARGET_ARCH)

# Make command(overwrite from the top make)
MAKE                    ?= gmake

# other command(for local)
RM                      = rm
RM_OPT                  = -f

# Linux RedHat7.3 - gcc2.96(overwrite from the top make)
CXX                     ?= /usr/bin/g++
OPTFLAG                 = -O3 -m32
DEFFLAG = -D__SYSTEMC_21__ -DSC_USE_SC_STRING_OLD ¥
          -DSC_INCLUDE_DYNAMIC_PROCESSES ¥
          -DNSMVINTC711_DEF -DCM_REPORT_OUT ¥
          -D_LARGEFILE64_SOURCE -D_FILE_OFFSET_BITS=64 ¥
          -DLINUX
DBGFLAG                 = -Wno-deprecated
INCPATH                 = -I$(SYSTEMC_INCPATH) -I.
```

Sample : ITextAccess1cpu (Linux)

PltfrmCompile/ITextAccess1cpu/pltfrm.tb.mk (continuation)

```
# Linux RedHat7.3 - gcc2.96(for local)
DEPFLAG             = -MM
DEFFLAG_MINE        = -D_V850E2R_LOCAL_BUS_ -DLINUX -DV850E2 ¥
                      -D__FLASH_CACHE_INTERNAL__
DBGFLAG_MINE        =
INCPATH_MINE        = -I$(MODEL_ATSLAVE_PATH) ¥
                      -I$(MODEL_TIMER_PATH) ¥
                      -I$(MODEL_ATMASTER_PATH)
CXXFLAGS            = $(OPTFLAG) $(DEFFLAG) $(DEFFLAG_MINE) ¥
                       $(DBGFLAG) $(DEFFLAG_MINE) $(INCPATH) ¥
                       $(INCPATH_MINE)
#LFLAGS             = -ldl -L$(SYSTEMC_LIB) -l$(SYSTEMC_LIB) ¥
                      -dynamic
LFLAGS              ?= -L$(SYSTEMC_L

# Files(for local)
OBJECTS             = pltfrm$(SIM_KERNEL_POSTFIX).o
LIBS                = $(SCHEAP_A) ¥
                       $(MODEL_ATSLAVE_A) ¥
                       $(MODEL_TIMER_A) ¥
                       $(MODEL_ATMASTER_A)

# Build rules
.PHONY: all clean
all:                $(TARGET)

$(TARGET): $(OBJECTS) $(LIBS)
                    $(CXX) $(CXXFLAGS) -o $@ -rdynamic $(OBJECTS) ¥
                    -ldl -lpthread -W1,-whole-archive $(LIBS) ¥
                    $(LFLAGS) -W1,-no-whole-archive
                    @echo "Done"
```

Path of user module

User module Archive

Platform

```
pltfrm$(SIM_KERNEL_POSTFIX).o: pltfrm.cpp
        $(CXX) -c $(CXXFLAGS) -o
        pltfrm$(SIM_KERNEL_POSTFIX).o pltfrm.cpp

clean:
        @($(RM) $(RM_OPT) $(OBJECTS) $(TARGET) core*);
#       @(if [ -d $(LIBPATH) ] ; then $(RM) -r $(RM_OPT)
        $(LIBPATH) ; fi;)

####### Dependencies
./pltfrm.o: ¥
        $(MODEL_ATSLAVE_PATH)/ATSLAVE.h ¥
        $(MODEL_TIMER_PATH)/TIMER.h ¥
        $(MODEL_ATMASTER_PATH)/ATMASTER.h
```

Header file of user description

Sample : ITextAccess1cpu (Windows)

Platform

- When SystemC model is executed with Multi debugger

- Execution of MULTI and SystemC model
  - % run_multi_win.bat
  - Execute run_multi_win.bat which is located in pltfrmCompile/build/ITextAccess1cpu/ from command prompt.
  - After MULTI logo is displayed, simulator is invoked and connected to MULTI debugger.
- Operation of MULTI debugger
  - Execute the program till last by clicking GO button after break point setting.
  - Close the windows for MULTI when simulation is finished.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITextAccess1cpu (Windows)

Platform

When SystemC model is executed without debugger

- Execute SystemC model
  - % run_core_win.bat
  - Execute run_core_win.bat which is located in pltfrmCompile/build/ITextAccess1cpu/ from command prompt. Do the execution until the execution cycle number specified in this script.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITextAccess1cpu (Linux)

Platform

- When SystemC model is executed with Multi debugger

- Execution of MULTI and SystemC model
  - % run_multi.csh
  - Execute run_multi.csh which is located in pltfrmCompile/build/ITextAccess1cpu/ from command prompt.
  - After MULTI logo is displayed, simulator is invoked and connected to MULTI debugger.
- Operation of MULTI debugger
  - Execute the program till last by clicking GO button after break point setting.
  - Close the windows for MULTI when simulation is finished.

➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Sample : ITextAccess1cpu (Linux)

Platform

When SystemC model is executed without debugger

- Execute SystemC model
  - % run_core.csh
  - Execute run_core.csh which is located in pltfrmCompile/build/ITextAccess1cpu/ from command prompt. Do the execution until the execution cycle number specified in this script.

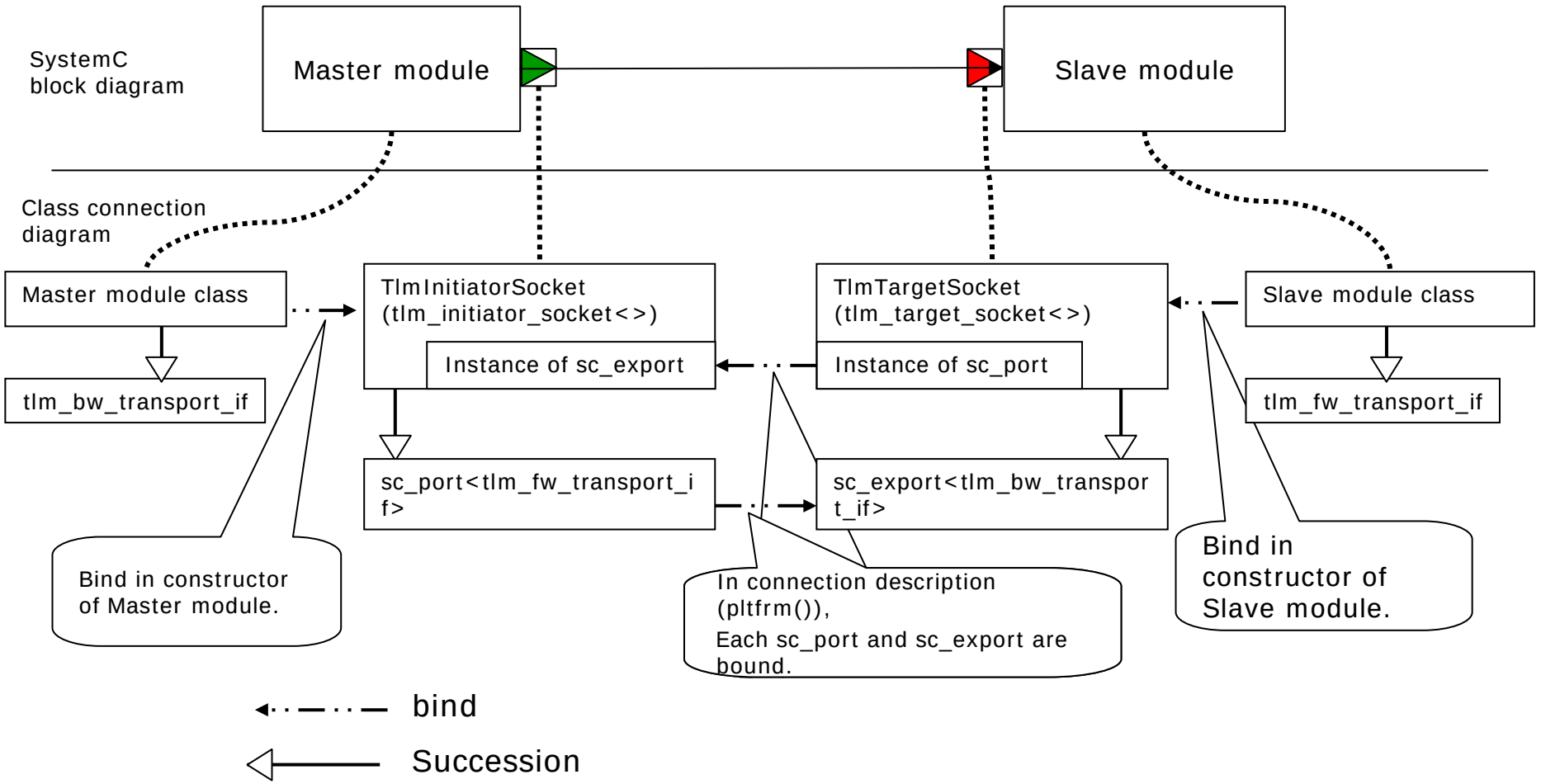
➢ Refer to bibliography 2 for a necessary settings for execution. (important)

Method of user module creation (common part of master / slave)

Connection of master <-> slave

User module

I/F based on OSCI TLM2.0 is used.



Payload used by user module

User module

- **Member variable of generic payload that can be used in user module**

Address	:	sc_dt::uint64	m_address
Command	:	tlm_command	m_command
Data pointer	:	unsigned char*	m_data
Data length	:	unsigned int	m_length *1
Width of stream	:	unsigned int	m_streaming_width *1
Response	:	tlm_response_status	m_response_status

  - Refer to bibliography 1 for get/set of the above-mentioned member variable. Other member variables are always used by the initial value.

- **Member variable of extended payload used in user module**

Lock signal	:	bool	mTargetLock
Pointer of target socket	:	TlmInitiatorSocket	*mpInitiatorSocket

  - Refer to the get/set method of extended payload.
    Set the pointer of the target socket in the master.

***1: The burst transmission is supported. Refer to the Burst transmission for the setting method of data length and the width of the stream in the burst transmission.**

get/set method of extended payload

User module

The tlm_extension class is used for extension of tlm_generic_payload.
(Refer to bibliography 1.)

[Set procedure]

- The storage area of additional payload is created.

```
TlmAdditionalPayload additionalPayload;
```

- The pointer of the storage area is set.

```
trans->set_extension( &additionalPayload );          // Trans is a pointer of tlm_generic_payload.
```

- Set of each signal

```
additionalPayload.setTargetLock( true );
additionalPayload.setInitiatorSocket( pSocket );     // initiator socket of master module
```

[Get procedure]

- Get after confirmation whether additional payload is included in transaction

```
TlmAdditionalPayload *pAdditionalPayload = NULL;
trans->get_extension( pAdditionalPayload ); // Trans is a pointer of tlm_generic_payload.
if( pAdditionalPayload ) {
        bool targetLock = pAdditionalPayload->getTargetLock();
} else {
                                                    // Additional payload is not set.
}
```

Burst transmission

The number of beats in the burst transmission is decided as follows by the variable of generic payload. (Refer to bibliography 1 for the burst transmission.)

```
unsigned int m_length;           // Total bytes number
unsigned int m_streaming_width;  // Streaming width
                                 // Number of bytes transferred on each beat
```

- Case1: 4bytes * 1 (single transfer)

```
m_length = 4;
m_streaming_width = 4;
```

- Case2: 4bytes * 8

```
m_length = 32;
m_streaming_width = 4;
```

- Case3: 1bytes * 4

```
m_length = 4;
m_streaming_width = 1;
```

Attention

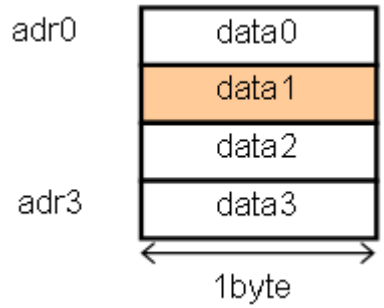
1. Set m_streaming_width in less than the bus width.
2. Set m_streaming_width to the same value as m_length at single transmission.

Data packing

User module

Pack only the specified size from the specified address.

Example: One byte data of address offset 1 in four byte alignment



Packed data in generic payload

Size : 1 byte

Address : adr1

Data : stored in array of unsigned char :
data[0]=data1, [1]=don't care, [2]=don't care, [3]=don't care

Image :

[0]	data1
[1]	
[2]	
[3]	

Makefile of user module (necessary in case of Linux) 1/2

Creation of makefile (for Linux)

User module

It is necessary to prepare following Makefile for the compilation of the module.

It is called from compilation time pltfrmCompile/IT*/Makefile.

pltfrmCompile/models/ATSLAVE/Makefile

The red-letter part: User writing part when user newly make the environment.

```
# Name(overwrite from the top make)
MODEL                           = ATSLAVE

# Location(overwrite from the top make)
PROJ_HOME                       = $(shell pwd)/../..
LIBPATH                         = lib-$(MODEL)
# Location(for local)
MODEL_HOME                      =$(PROJ_HOME)/models
MODEL_COMMON_PATH               =$(MODEL_HOME)/common

# SystemC location and architecture(overwrite from the top make)
SYSTEMC_HOME                    =/home/product/systemc/tools/systemc
TARGET_ARCH                     = linux
SYSTEMC_INCPATH                 =$(SYSTEMC_HOME)/include

# Make command(overwrite from the top make)
MAKE                            = gmake

# other command(for local)
RM                              = rm
RM_OPT                          = -rf
AR                              = ar cqs
# Linux RedHat7.3 - gcc2.96(overwrite from the top make)
CXX                             = /usr/bin/g++
OPTFLAG                         = -g
```

IP name

Makefile of user module (necessity in case of Linux) 2/2

User module

PltfrmCompile/models/ATSLAVE/Makefile (continuation)

The red-letter part: User writing part when user newly make the environment.

```
DEFFLAG                                 =
DBGFLAG                                 = -Wno-deprecated
INCPATH                                 = I$(SYSTEMC_INCPATH) -I. -I$(MODEL_COMMON_PATH)
# Linux RedHat7.3 - gcc2.96(for local)
DEPFLAG                         = -MM
DEFFLAG_MINE = -D_V850E2R_LOCAL_BUS_ -DLINUX -DV850E2
DBGFLAG_MINE                    =
INCPATH_MINE =
CXXFLAGS        = $(OPTFLAG) $(DEFFLAG) $(DEFFLAG_MINE) $(DBGFLAG) $(DBGFLAG_MINE) $(INCPATH) $(INCPATH_MINE)

# Files(for local)
SOURCES                         = ATSLAVE.cpp
OBJECTS                         = $(LIBPATH)/ATSLAVE.o
TARGET                          = $(LIBPATH)/$(MODEL).a

# Implicit rules
%.d:%.cpp
                $(CXX) $(DEPFLAG) $(CXXFLAGS) $< | sed 's!$*.o:!$$(LIBPATH)/&!g' > $@
$(LIBPATH)/%.o:%.cpp
                $(CXX) -c $(CXXFLAGS) -o $@ $<

# Build rules
.PHONY : all compile clean
all:
                if test ! -d $(LIBPATH); then mkdir $(LIBPATH); fi;
                $(MAKE) compile

compile: $(TARGET)

$(TARGET): $(OBJECTS)
                $(AR) $@ $(OBJECTS) 2>&1 | c++filt
                @echo "Done"

$(OBJECTS): $(SOURCES) $(SOURCES:.cpp=.d)

clean: The future omission
```

The source file name to be compiled is enumerated.

Link object file name is enumerated.

Method of user module creation (slave IP)

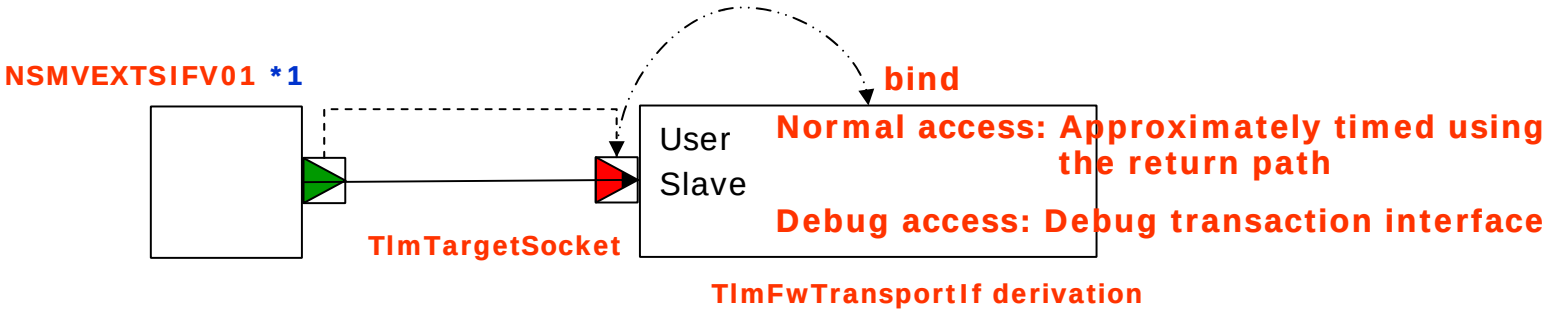
Method of making user slave

Slave

- TlmTargetSocket which derives from OSCI tlm::tlm_target_socket<32,tlm_base_protocol_type,0> is used as a socket.
- The class of the user slave derives from TlmFwTransportIf(tlm::tlm_fw_transport_if<>).
- TlmTargetSocket and the user slave are bound in the user slave constructor.
- It communicates with Approximately-timed using the return path at Normal access.
- It communicates with the debug transaction interface at Debug access.
- The base address and the size of the user slave are set from NSMVEXTSIFV01 through TlmTargetSocket. Use if necessary.
- The burst transmission is supported.

Limitations

- Local time cannot be passed by the third argument of nb_transport_bw() and nb_transport_fw(). (There is no influence in the simulation even if the values other than 0 are set.)
- Do notifying of completion of the transaction (call of nb_transport_bw()) before the rising edge of the clock.

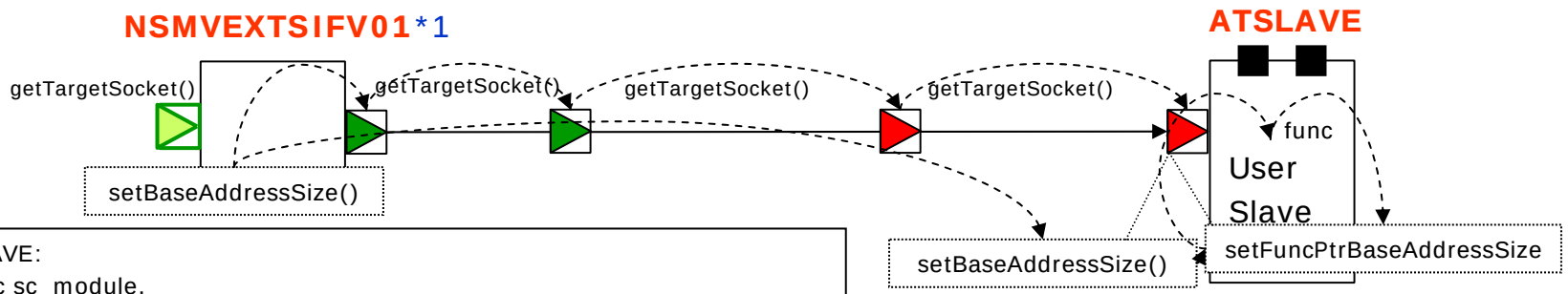


*1: NSMVEXTSIFV01: Slave bus I/F (APB I/F etc)

Realization of base address and size setting of slave setBaseAddressSize()

Slave

- ATSLAVE is identified from NSMVEXTSIFV01*1 through getTargetSocket().
- Base address and size are set with setBaseAddrressSize().
- The function which is called back from setBaseAddressSize() is set in ATSLAVE.




```
ATSLAVE:
 public sc_module,
 virtual public TlmFwTransportIf,
 public OSCI2SLAVEBASE
{
Public:
 TlmTargetSocket target_socket;

public:
 SC_HAS_PROCESS(ATSLAVE);
 ATSLAVE(sc_core::sc_module_name name,
                                  bool invalidate = false) :
 target_socket("target_socket"),
 m_invalidate(invalidate)
 {
   target_socket(*this);
   target_socket.setFuncPtrBaseAddressSize
   (this, &ATSLAVE::setfunc);
 }
};
```

*1: NSMVEXTSIFV01: Slave bus I/F (APB I/F etc)

Function used by user slave

Slave

- **Function which user must implement in the slave**

  Function for access usually

  TlmSyncEnum nb_transport_fw(TlmTransactionType& *2, TlmPhase&, sc_core::sc_time&)

  Function for debugging access

  unsigned int transport_dbg(TlmDebugTransactionType& *3)

  Function for blocking access (It is necessary having [ri] prepared by the dummy though doesn't use).

  TlmSyncEnum b_transport(TlmTransactionType& *2, sc_core::sc_time&)

  Function for direct memory access (It is necessary having [ri] prepared by the dummy though doesn't use).

  bool get_direct_mem_ptr(const sc_dt::uint64&, TlmDmiMode&, tlm::tlm_dmi&)

  void invalidate_dmi_method(void)

  Function that sets size of user slave

  Void setfunc(void)

- **Function can be called from slave**

  Function of tlm::tlm_bw_transport_if and TlmTargetSocket. It uses it like target_socket->nb_transport_bw.

  Function for access usually (for response)

  TlmSyncEnum nb_transport_bw(TlmTransactionType& *2, TlmPhase&, sc_core::sc_time&)

  Function for base address acquisition

  ADDRESS_TYPE getBaseAddress()

  Function for acquisition of size of base

  ADDRESS_TYPE getBaseSize()

  Slave size area setting function pointer setting function

  void setFuncPtrBaseAddressSize()

*2: TlmTransactionType is tlm::tlm_generic_payload.

*3: TlmDebugTransactionType is tlm::tlm_generic_payload.

Payload used by user slave

Slave

- **Member variable of generic payload that can be used in user slave**

Address	:	sc_dt::uint64	m_address
Command	:	tlm_command	m_command
Data pointer	:	unsigned char*	m_data
Data length	:	unsigned int	m_length *1
Width of stream	:	unsigned int	m_streaming_width *1
Response	:	tlm_response_status	m_response_status

  - Refer to bibliography 1 for get/set of the above-mentioned member variable. Other member variables are always used by the initial value.

- **Member variable of extended payload used in user slave**

Lock signal	:	bool	mTargetLock
Pointer of target socket	:	TlmInitiatorSocket *	mpInitiatorSocket

  - Refer to the get/set method of extended payload.

***1: The burst transmission is supported. Refer to the Burst transmission for the setting method of data length and the width of the stream in the burst transmission.**

Example of user slave implementation

Slave

Refer to pltfrmCompile/models/ATSLAVE for the example of the user slave implementation.

Method of user module creation (master IP)

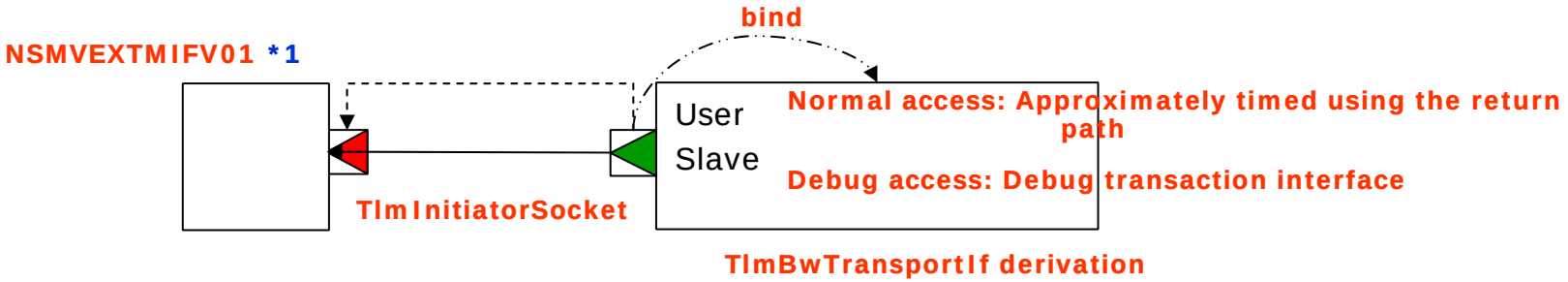
Method of making user master

Master

- TlmInitiatorSocket which derives from OSCI tlm:: tlm_initiator_socket<32,tlm_base_protocol_type,0> is used as a socket.
- The class of the user slave derives from TlmBwTransportIf(tlm::tlm_bw_transport_if<>).
- TlmInitiatorSocket and the user slave are bound in the user slave constructor.
- It communicates with Approximately-timed using the return path at Normal access.
- It communicates with the debug transaction interface at Debug access.
- The burst transmission is supported.

Limitations:

- Local time cannot be passed by the third argument of nb_transport_bw() and nb_transport_fw(). (There is no influence in the simulation even if the values other than 0 are set.)
- Set initiator socket pointer which is member of the extended payload, when the transaction is transmitted. The setting method must be to refer to the get/set method of extended payload.



*1: NSMVEXTMIFV01: Master bus I/F (AHB I/F (M))

Function used by user master

Master

Function which user must implement in the master

Function for access usually

TlmSyncEnum nb_transport_bw(TlmTransactionType& *2, TlmPhase&, sc_core::sc_time&)

Function for direct memory access (It is necessary having prepared by the dummy though doesn't use).

void invalidate_direct_mem_ptr(sc_dt::uint64 start_range, sc_dt::uint64 end_range)

Function can be called from master

Function of tlm::tlm_fw_transport_if and TlmInitiatorSocket.

It uses it like target_socket->nb_transport_fw.

Function for access usually

TlmSyncEnum nb_transport_fw(TlmTransactionType& *2 , TlmPhase&, sc_core::sc_time&)

Function for debugging access

unsigned int transport_dbg(TlmDebugTransactionType& *3)

***2: TlmTransactionType is tlm::tlm_generic_payload.**
***3: TlmDebugTransactionType is tlm::tlm_generic_payload.**

Payload used by user master

Master

- **Member variable of generic payload that can be used in user master**

Address	:	sc_dt::uint64	m_address
Command	:	tlm_command	m_command
Data pointer	:	unsigned char*	m_data
Data length	:	unsigned int	m_length *1
Width of stream	:	unsigned int	m_streaming_width *1
Response	:	tlm_response_status	m_response_status

  - Refer to bibliography 1 for get/set of the above-mentioned member variable. Other member variables are always used by the initial value.

- **Member variable of enhancing payload used in user master**

Lock signal	:	bool	mTargetLock
Pointer of target socket	:	TlmInitiatorSocket *	mpInitiatorSocket

  - Refer to the get/set method of extended payload.
    Set the pointer of the target socket in the master.

***1: The burst transmission is supported. Refer to the Burst transmission for the setting method of data length and the width of the stream in the burst transmission.**

Example of user master implementation

Master

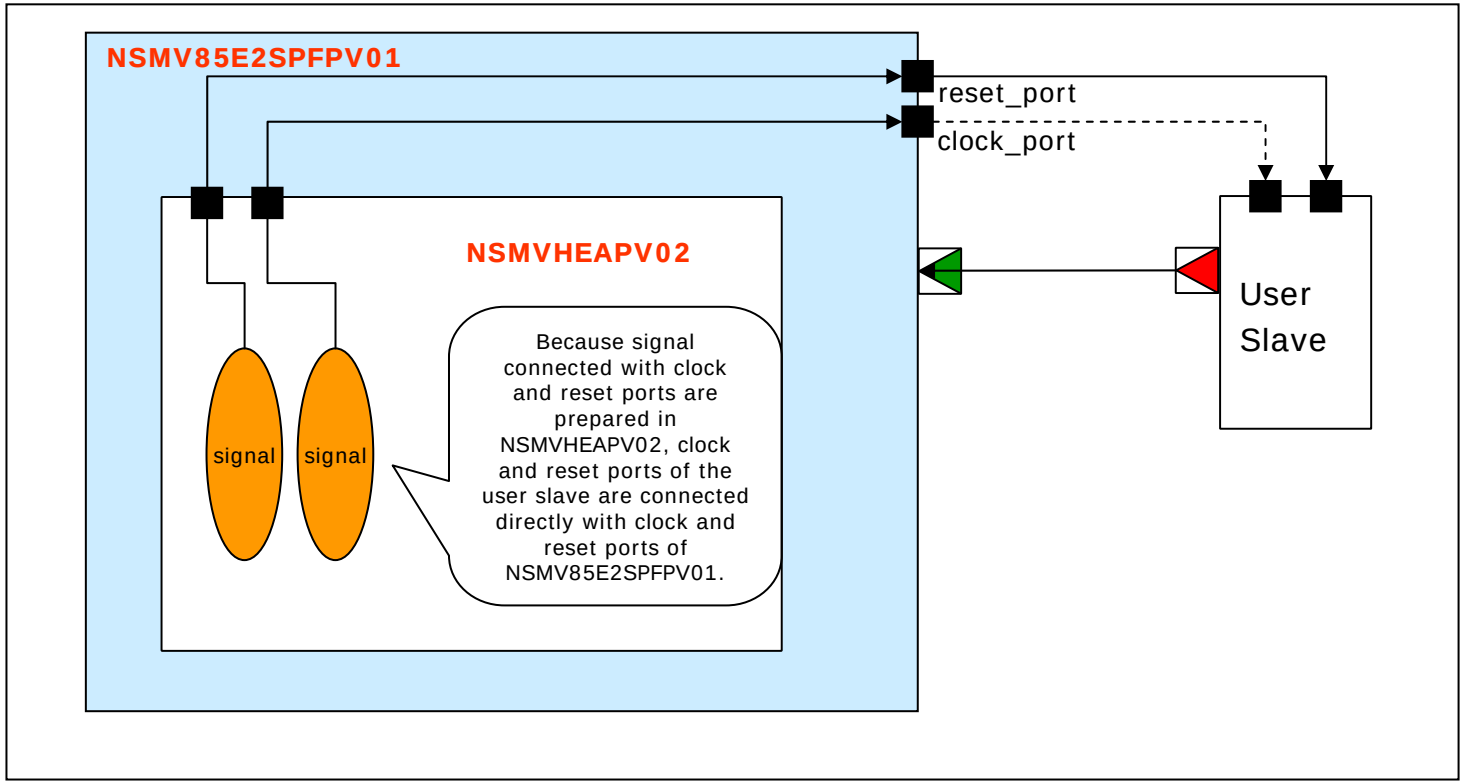
Refer to pltfrmCompile/models/ATMASTER for the example of the user master implementation.

PIN I/F connection of SCHEAP.lib(.a)

PIN I/F connection (clock and reset port)

PIN I/F

The port is connected directly.

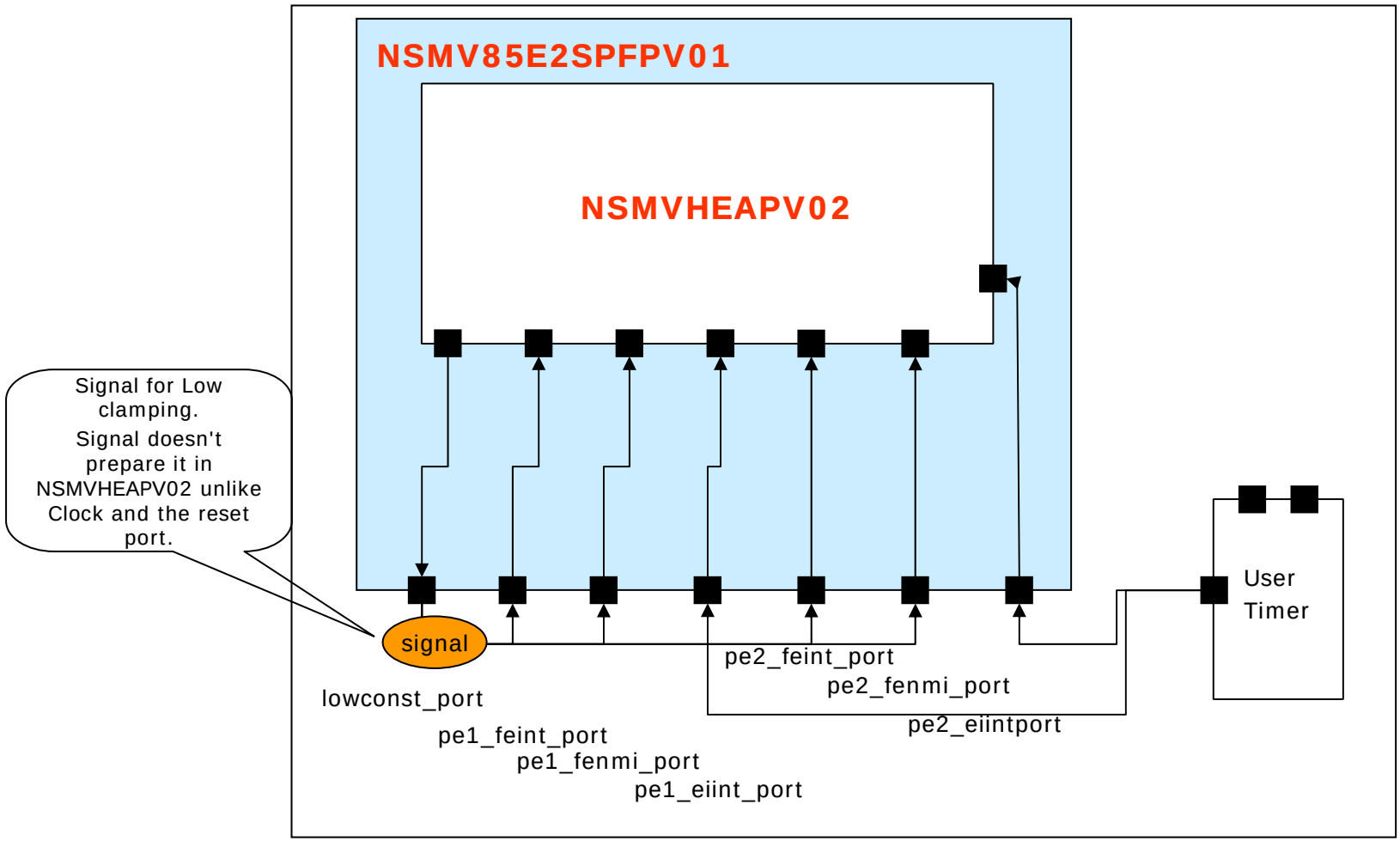


Importance: Clock_port is not connected in the example of the modeling environment in the package.
Clock port is prepared, but it is desirable that the IP synchronized with clock is not connected to lead to a simulation speed decline.

PIN I/F connection (lowconst signal)

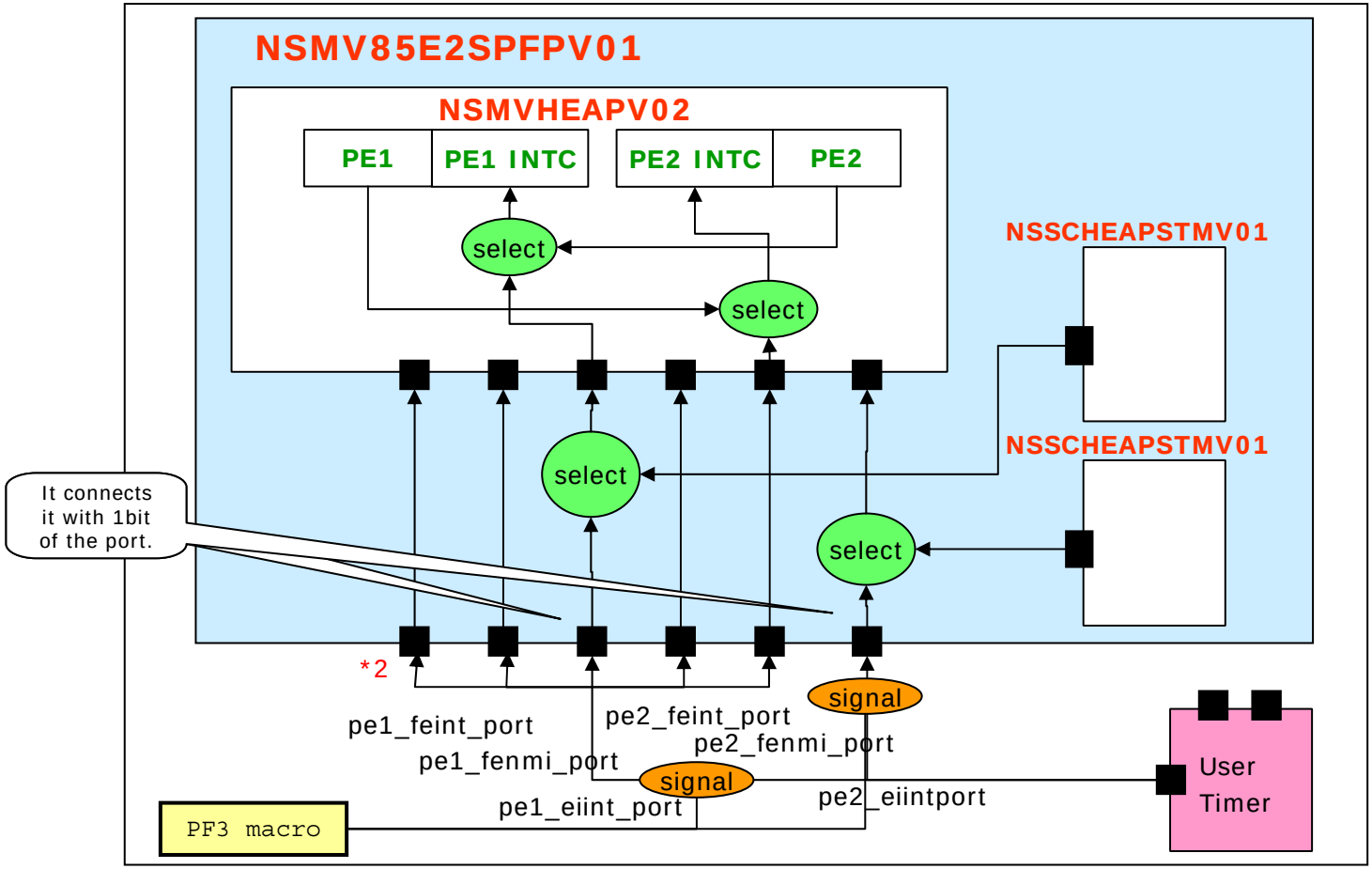
PIN I/F

The Low clamping signal is prepared.



PIN I/F connection (INTC port)

The user timer is connected with the port where interrupt signal wants to be input.
When two or more IPs are connected with the same port, the connection is interpreted by the following preferential orders.
Interrupt between PEs > Internal timer(NSSCHEAPSTMV01) > user timer(descrived in pltfrmFC() *1) > PF3 peripheral macro > user timer(described in pltfrm() *1)

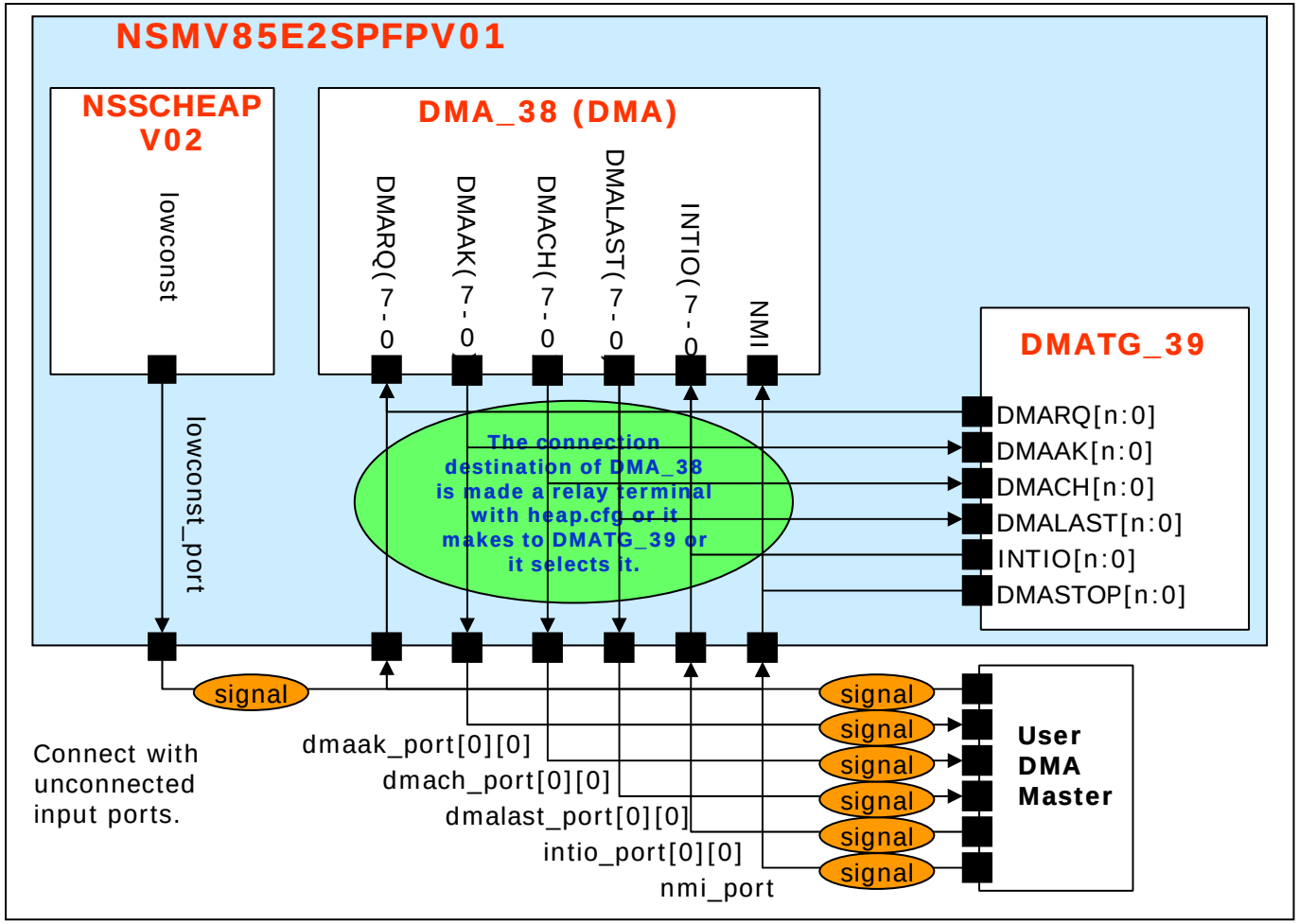


*1: **Refer to User connection descriptive function pltfrm() / pltfrmFC()**

*2: About an unconnected INTC request ports, they are not necessary to be treated by user, because they are connected with the low_const port in SCHEAP-G4.a.

PIN I/F connection (DMA control port)

External DMA master is connected with the control port of DMA.
The port of DMA(DMA_38) in SC-HEAP is connected to external DMA control port or internal DMA master (DMATG_39). (Select it with heap.cfg . Refer to the Configuration.)



- The left figure shows the case that the number of channel of DMA set to 8 and external DMA master is connected with channel 0 of DMA.
  8-40 channels can be set to the number of channel of DMA.
  SC-HEAP instantiates plural DMA_38 which has 8 channels, when 9 channels or more is set.
  The specification method of the DMA control port in that case is as follows.

Example: When the number of channel is set to 40(0～39), and dmarq_port of Ch39 is specified

dmarq_port[4][7]

Module No. (0～4)

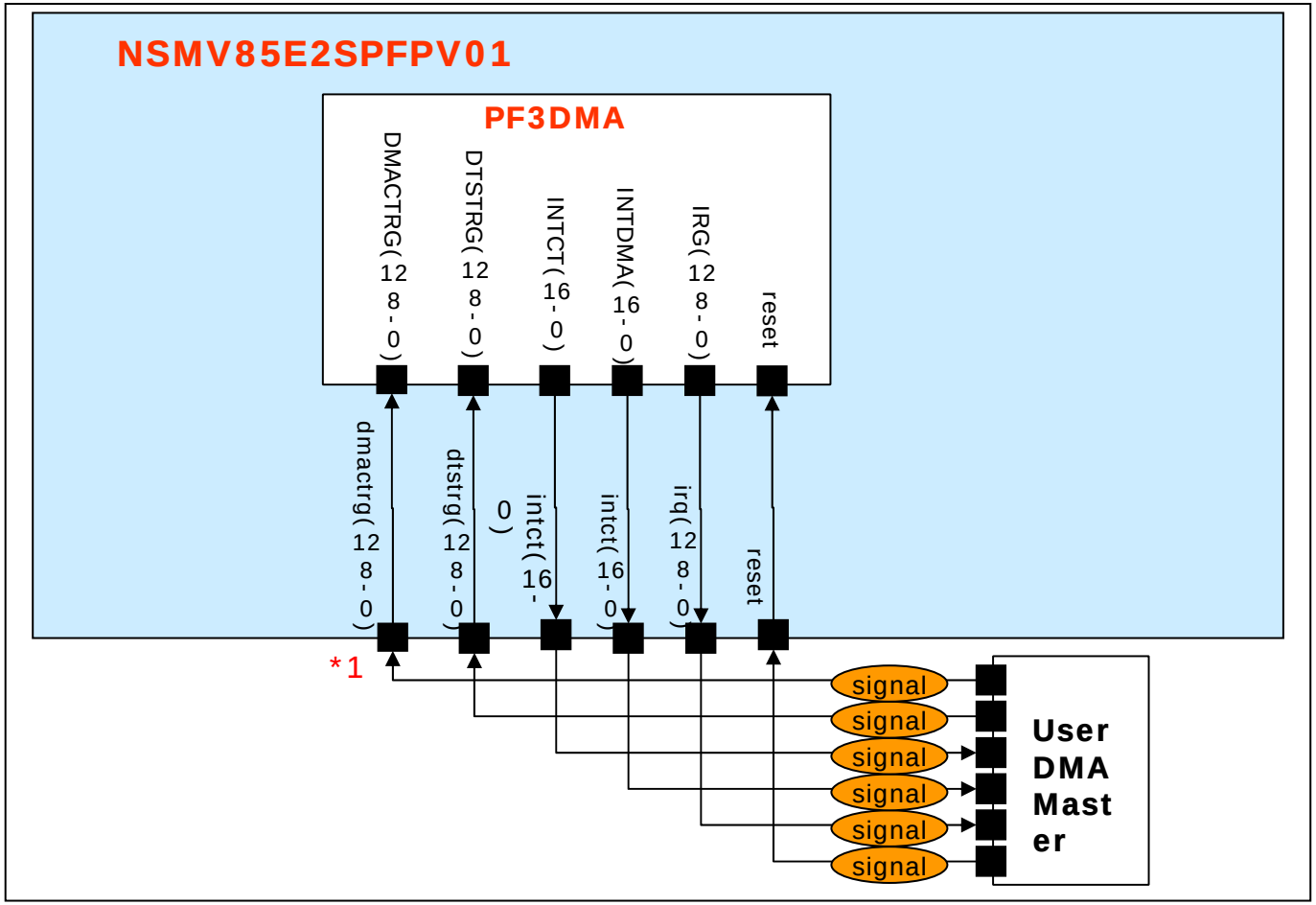
Channel No. (0～7)

PIN I/F connection (PF3 DMA control port)

- There is no sample environment.

This sheet is not necessary to disclose to the user who does not use PF3 IP.

External DMA master can be connected with the control port of PF3DMA.



*1: About an unconnected PF3DMA control ports, they are not necessary to be treated by user, because they are connected with the low_const port in SCHEAP-G4.a.

Bibliography

[1] OSCI TLM2 USER MANUAL http://www.systemc.org/downloads/standards/tlm20/

[2] SC-HEAP V3.11 Users Manual (OSCI version V850E2 HEAP) ZSG-F31-11-0037

Renesas Electronics Corporation