Development of a demonstration system for Sony technology fair

Hiroyuki Yagi

Purpose of this document

- This document will report the trial and give some feedbacks
  - The design flow we used after the code generation to make the codes synthesizable
  - What can be done to improve the performance
- Discussion
  - Points to be improved for Sony internal review (DR2, middle of January)
  - Deliverable for DR2
    - Tools (Mentor Graphics)
      - Will be discussed
    - Documents (Sony)
      - Setup guide (launched)
        - Includes SystemC, GCC3.2.2, PthreadLib
      - User's guide (launched)
      - Examples (done, Dec-4)

Contents

- The codes generated by BP-MC
- Modifications to synthesize the codes
  - Modifications of the header files
  - Modifications of the C++ source files
  - Modifications to use the Modular IFs
  - Modifications to use the In-house IFs
  - Modifications for the behavioral synthesis
- Summary
- Discussion

The codes generated by BP-MC

- Data types
  - Specified by data types in "Data Type Package Diagram"
    - User defined data type
    - Structured data type
    - Enumeration data type
    - Constant data type
  - Easy to add/modify
  - Can have a default value

  Note: Preferred to have a default value for each variable
- Interfaces
  - Specified by interface in "Interface package Diagram"
  - can contain the signal direction such as "server to client"
  - can have a signal/operation which can have (array) parameters
  - signal name will be used as the channel name

  Note: We need to handle custom interfaces like modular interfaces (Explanation will be shown in next slide)
- Ports
  - Specified with a rule, Provided-in and Required-out
    - sc_in
    - sc_out
  - Also have the port name section on the "Component Diagram"

  Note: Need to clean up the semantics

A work around to use custom interfaces

■We added an extra unnamed parameter to use our custom IFs

CLUT.h

```
#ifndef CLUT_PORTS_H
#define CLUT_PORTS_H

#include "sys_types.h"

#include <[redacted]>
#include "memlib.h"
#include "[redacted]_event.h"
#include "[redacted]para.h"

SC_MODULE( CLUT ) {
typedef [redacted]<sc_int<8>, [redacted]>::out data_required_port;
typedef SRAM[redacted]::port<[redacted]> mem2_required_port;
typedef [redacted]<sc_int<8>, [redacted]>::in din_provided_port;
typedef [redacted]_event_in<[redacted]> start_provided_port;
typedef [redacted]para_slave_port parameter_provided_port;

  /*----- Port Declaration ------*/
    sc_in< bool > clk;
    sc_in< bool > rst_X;
    data_required_port  CLUT_Port1_data;
    mem2_required_port  CLUT_Port3_mem2;
    din_provided_port  CLUT_Port2_din;
    start_provided_port CLUT_Port4_start;
    parameter_provided_port  CLUT_Port5_parameter;

  /*----- Instantiation of Submodules ------*/
```

A rule check tool generates ERROR for disagreement of the macro name with filename (needs CLUT_H)

hidden for security purpose

sysc_sets.cpp contains some uninitialized variables which also result generating errors when a rule check tool is used

2. We need to define the unknown port types, while we have the freedom to use our custom interfaces!

1. MC generates the declaration of ports with unknown port types

■Direct use of our custom IFs is required. The default reset/steady state action for the specific IFs will be a nice option to have.

Modifications of the header files

FILTER.h

```
#define FILTER_PORTS_H

#include "sys_types.h"

#include "HFILTER.h"
#include "CLUT.h"

SC_MODULE( FILTER ) {
// FILTER ports
typedef SRAM[redacted]::port<[redacted]> mem2_required_port;
typedef [redacted]_event_out<[redacted]> done_required_port;
typedef [redacted]<sc_int<8>, [redacted]>::base_out dout_required_port;
typedef [redacted]<sc_int<8>, [redacted]>::base_in din_provided_port;
typedef [redacted]_event_in<[redacted]> start_provided_port;
typedef [redacted]_port parameter_provided_port;

// FILTER channels
typedef [redacted]<sc_int<8>, isConfig> data;
typedef [redacted]<[redacted]> start;
typedef [redacted] parameter;
typedef [redacted]<sc_int<8>, [redacted]> din;
typedef [redacted]<[redacted]> start;

  /*----- Port Declaration ------*/
    sc_in< bool > clk;
    sc_in< bool > rst_X;
    mem2_required_port  FILTER_Port1_mem2;
    done_required_port FILTER_Port3_done;
    dout_required_port  FILTER_Port6_dout;
    din_provided_port  FILTER_Port2_din;
    start_provided_port FILTER_Port5_start;
    parameter_provided_port  FILTER_Port4_parameter;


  /*----- Instantiation of Submodules ------*/
    HFILTER  iHFILTER;
    CLUT  iCLUT;


  /*----- Internal Variables ------*/


  /*----- State Machine Internal Variables ------*/


  /*----- Signal and Event Declaration ------*/
    data HFILTER_Port1_data ;
    start HFILTER_Port4_start ;
    parameter HFILTER_Port5_parameter ;
    din CLUT_Port2_din ;
    start CLUT_Port4_start ;
    parameter CLUT_Port5_parameter ;
```

1. Define types used for the custom interfaces and the custom channels

2. Add the declarations of the user functions/behaviors

3. Register the functions/behaviors as SC_CTHREAD or SC_METHOD

Prefer to have these converged in a file

Modifications of the C++ source files

Rewrite the Default behavior

CLUT.cpp

```
/*-----------------------------------------------------------------------
 * File:  CLUT_ports.cpp
 *
 * UML Component Port Messages
 * Component Name:                        CLUT
 * Domain Name (if modeled internally):   CLUT
 *
 * (C) Copyright 1998-2009 Mentor Graphics Corporation.  All rights reserved.
 *-----------------------------------------------------------------------*/

#include "sys_types.h"
#include "CLUT.h"

void CLUT::component_main(void)
{

   CLUT_Port1_data.write(0);
   CLUT_Port3_mem2.write(0);

    wait();
    while(true){
       wait();
    }
}
```

1. Default Reset action: ports initializers

2. Delete the Default action: do nothing but wait/dispatch the process

3. Insert the new codes or call your function from here.

4. Use State machine specified in the “Instance State machine” inside the Class Diagram N/A so far

Modifications to use Modular IF

- Stream IF and Memory IF
  - use typedef statement
  - required to bind clk and res_X to ports on DUT
  - required to call reset () on Reset for ports on DUT
- Parameter IF
  - use typedef statement

Behavior in thread need to modify to use these IFs

Modifications to use In-house IF

- In-house event IF
  - use typedef statement
  - required to call reset () on Reset for ports on senders (output ports)
- In-house stream IF
  - use typedef statement
  - rewrite the pair of "SC_CTHREAD and reset_signal_is description" to use SONY_THREAD

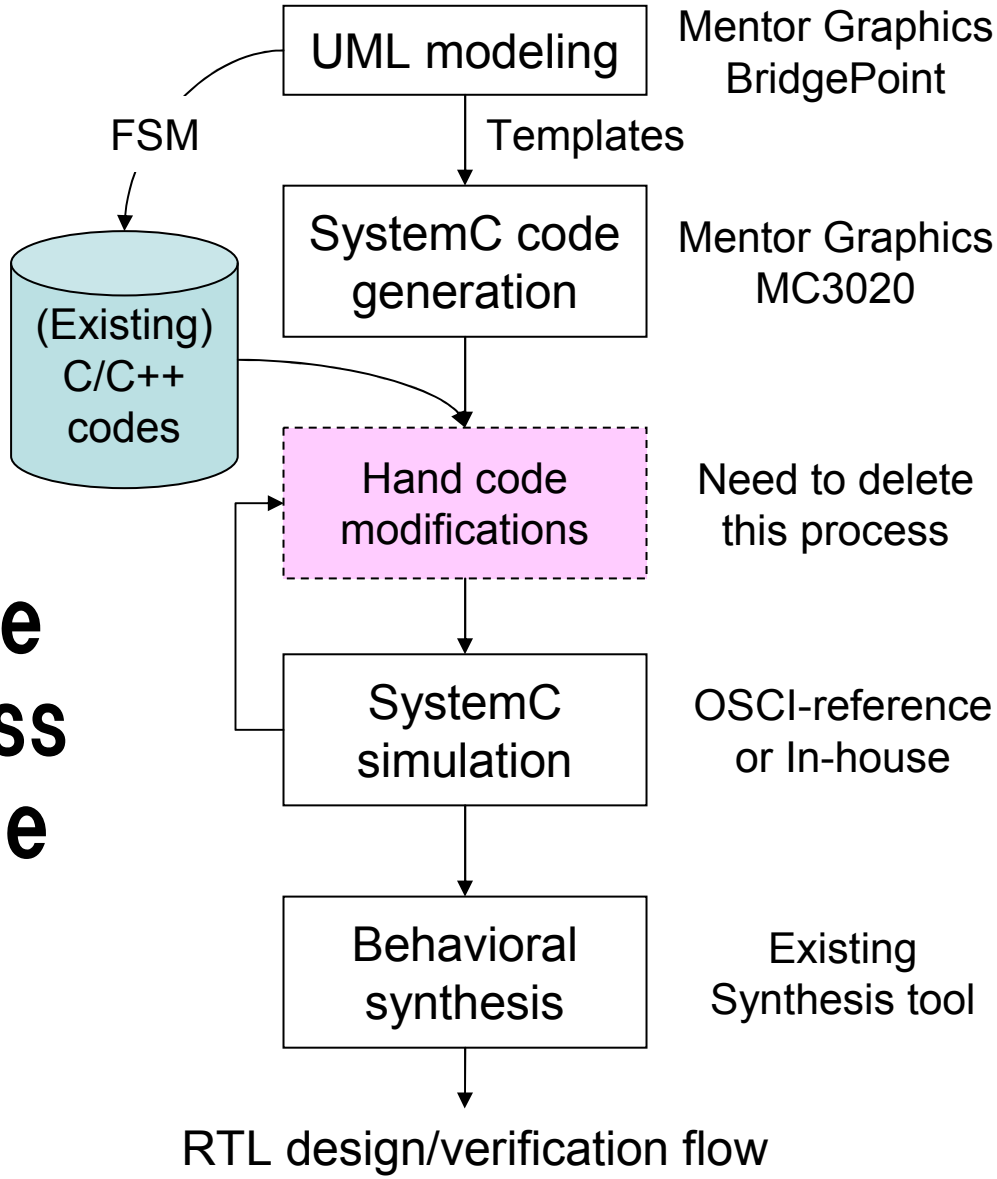
Behavior in thread need to modify to use these IFs

Modifications for behavioral synthesis

- Common modifications
  - Remove statement using sdtio
- Tool-dependent modifications
  - Simulation control descriptions
    - Modify to use In-house_elaborate and In-house_cleanup
    - Change file name of the header (_wrapper.h)
    - Change the instance name of the module (_wrapper)
  - Constraints for the behavioral synthesis
    - Insert the right constraint to the right place!

Summary

- A new design flow bridging the gap between the UML and synthesizable SystemC is stated
- Want to remove the hand coding process as many as possible



Discussion

- Required
  - User defined Interfaces
- Want to have/use
  - Default value for each variable
  - Switch options for generating codes
    - Switch of default Reset/steady state action
    - Switch of code (C/C++/SystemC) for state machine/action diagram
  - Registration of the user functions (SC_CTHREAD/SC_METHOD/SONY_THREAD)
- Want to define
  - Interfaces
    - Provided==Server
      - In for P2P (non-protocol aware) communication
      - bus for bus IF (protocol aware) communication
    - Requested==Client
      - Out for P2P (non-protocol aware) communication
      - Peripherals for bus IF (protocol aware) communication
  - Ports (signal direction)
    - In
      - Provider for "Client to Server" (Requester to Provider)
      - Requester for "Server to Client" (Provider to Requester)
    - Out
      - Requester for "Client to Server" (Requester to Provider)
      - Provider for "Server to Client" (Provider to Requester)