

Multimedia SoC Design

Homework of Lab 3

April 8, 2016

In the previous two homework, you have practiced:

1. *Refine a FIR filter example from blocking process to non-blocking process*
2. *Refine the non-blocking FIR filter to pipelined FIR filter structure.*
3. *Follow the predefined interface to design a hierarchical channel and use the access functions through ports for Transaction-Level Modeling (TLM). The communications between hardware blocks are modeled at Untimed (UT) level or Approximated-timed (AT) level.*
4. *Refine the communication from UT or AT level function calls to Pin-Cycle Accurate (PCA).*

1 and 2 are focused on the usage of SystemC inside a hardware block. 3 and 4 are focused on the TLM interface and communication refinement.

For a multimedia SoC, there are usually tens of hardware component within the system. These IPs are connected to a shared bus, and they follow the bus protocol to communicate with each other. However, in the early stage of system modeling, usually we don't want to model the communication as some specific bus protocol. We define higher level interfaces for simplicity, flexibility, and better simulation speed. The flexibility of communication refinement allows us to choose a target bus protocol after we complete the functional and architecture modeling and verification. The developed hardware models based on the interfaces are generic virtual components which can be adapted to other system modeling with a little wrapping work.

The goal of homework 3 is to introduce the bus-interface wrapping and to let you practice how to deal with different TLM-bus library. For code reuse, the hardware components we choose are from homework 2: *the interchange unit and the 2D memory*. For simplicity, the TLM-bus we use in homework 3 is the *SimpleBus* example, which was introduced in Lab. 3.

0.

Review the SimpleBus example first. Start from Simple_Bus.pdf and code/simple_bus/README. Here we briefly describe the SimpleBus project hierarchy for you. SimpleBus defines two types of interface for master: `simple_bus_master_blocking_if`, `simple_bus_master_nonblocking_if`. One for slave: `simple_bus_slave_if`. One for arbiter: `simple_bus_arbiter_if`. And one for direct access to memory-mapped slave: `simple_bus_direct_if`. A hierarchical channel: `simple_bus` is constructed and implements the master interfaces and the direct interface. The slave interface is implemented in the memory. The arbiter interface is implemented in the arbiter. The whole project applies indirect implementation style – i.e. separate module/channel declaration and implementation.

1.

Assume you get a Untimed (UT) or Approximated-timed (AT) interchange unit model and a 2D memory model (Actually you use homework 2 models). You are asked to port the model to SimpleBus. You shall not modify your original source code of the interchange unit and the memory **except adding wait()**. Wrappers are required to convert high-level interface transaction to/from specific bus protocol, which is SimpleBus here.

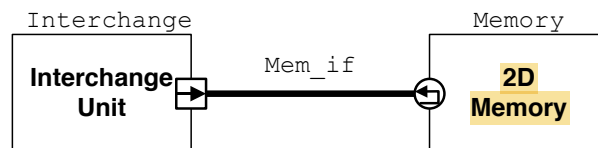


Fig.1 Original UT/AT model

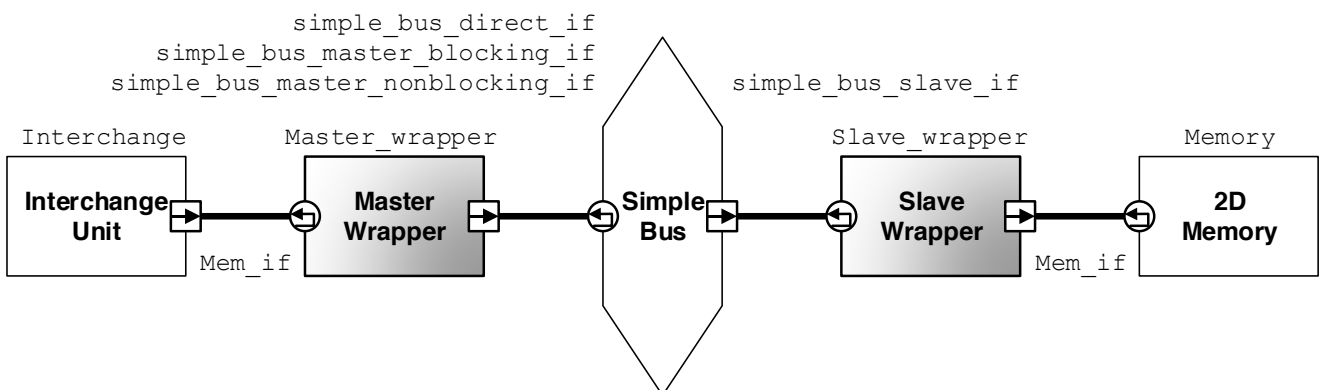


Fig.2 Port UT/AT IP model to SimpleBus

Since the 2D memory use two address to access data, a 2D-to-1D address

conversion has to be done in Master Wrapper and 1D-to-2D address conversion has to be done in Slave Wrapper. The simple bus uses a unified 32-bit byte-address linear space. A slave has to be defined its start address and end address in the space, and so does the slave wrapper.

Requirement : simple_bus_blocking_if

2.

Assume you get a Pin-Cycle-Accurate level interchange unit model and a 2D memory model (Actually you use homework 2 models). You are asked to port the model to SimpleBus. You shall not modify your original source code of the interchange unit and the memory **except adding wait()**. A pin cycle accurate master wrapper is used to connect `sc_signal` and `sc_signal_rv` typed channels and convert them to a specific bus protocol. A pin cycle accurate slave wrapper is used to convert SimpleBus protocol to ports which connect `sc_signal` and `sc_signal_rv` typed channels.

Requirement : simple_bus_non_blocking_if.h

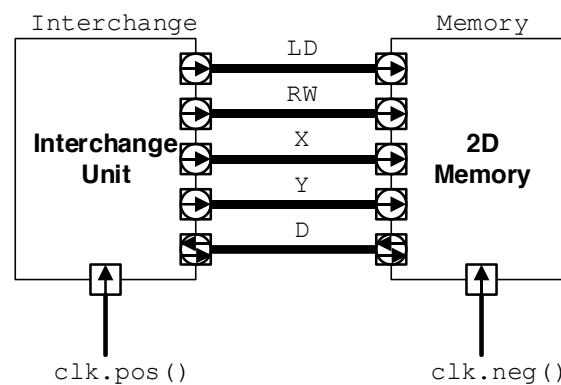


Fig.3 Original PCA model

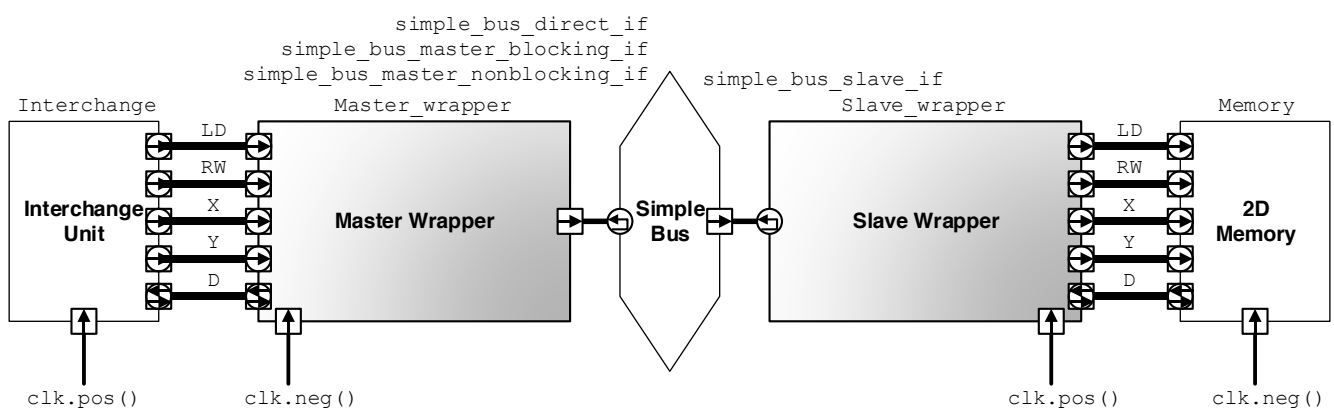



Fig.4 Port PCA IP model to SimpleBus

 All the files need to be compressed as a single ZIP or RAR file.
Send this file to TA via email: ntumsoc@gmail.com

Examples of email title:

[\[MSOC\] HW3_R04901001](#)

Examples of filename:

[MSOC_HW3_R04901001.zip](#)

Due date: 2016/04/22 Before Class (2 weeks)



繳交規定

source code (包括 lab 和作業)、report in PDF

請將這些檔案放入一個資料夾內如下排放

/HW3/	放report
/HW3/Lab3/	為Lab3的專案資料夾
/HW3/problem_1/	為1的專案資料夾
/HW3/problem_2/	為2的專案資料夾

請注意，專案資料夾內必須包含 [.vcxproj\[.*\]](#) 檔和 [.sln](#) 檔，將SystemC必須預先設定以下五項專案參數先設定好

1. VC++目錄的 Include目錄路徑 & 程式庫目錄路徑
2. [systemc.lib](#)
3. [_CRT_SECURE_NO_WARNINGS](#)
4. Multi-threaded Debug (/Mtd)
5. [/vmg](#)

➤ 為避免整份檔案過大，請在專案完成後，刪除所有的[Debug資料夾](#)，以及[ipch資料夾](#)、[.sdf檔](#)、[.suo檔](#)...等等。