# *Lab 3*

## TLM 2.0 & Simple bus

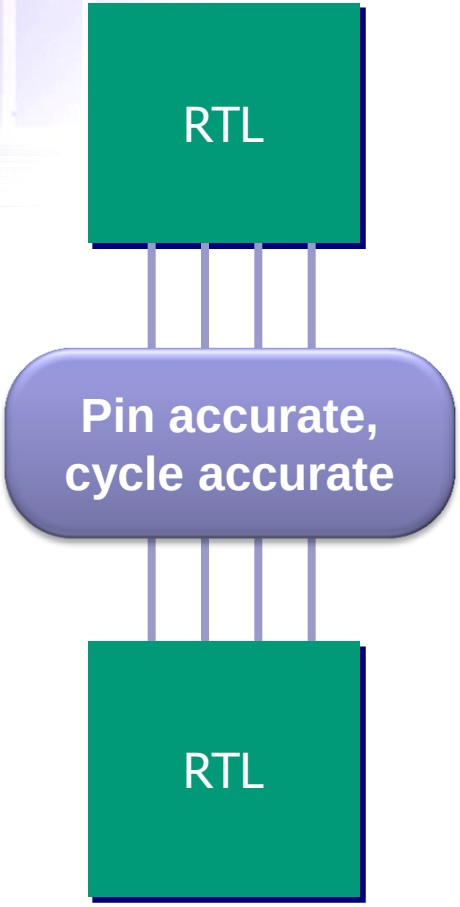
TA: Po-Chen Wu (吳柏辰)

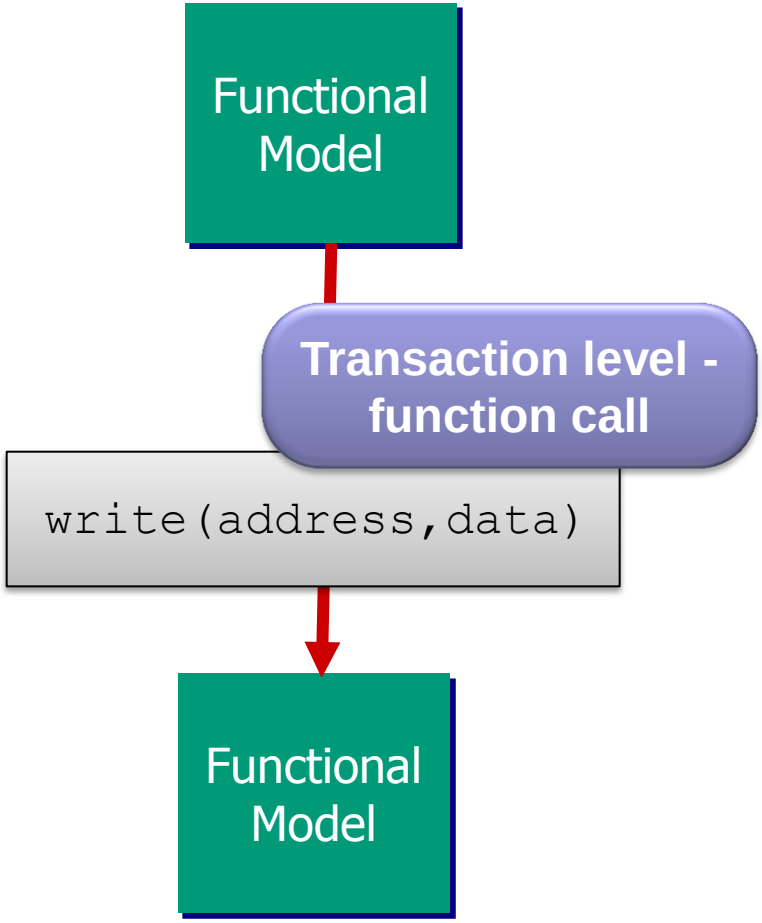
# Outline

- TLM 2.0
- Simple Bus Example
- Lab 3 Practice: Edge Detection

# TLM2.0 Introduction

# Introduction



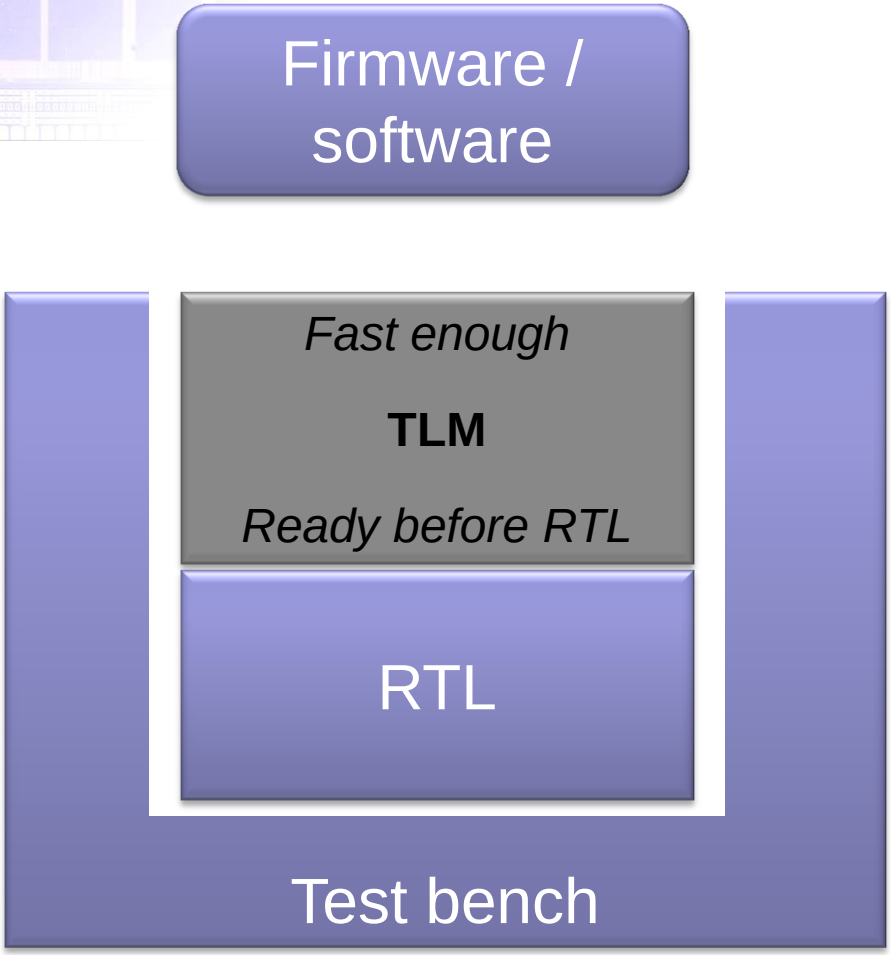



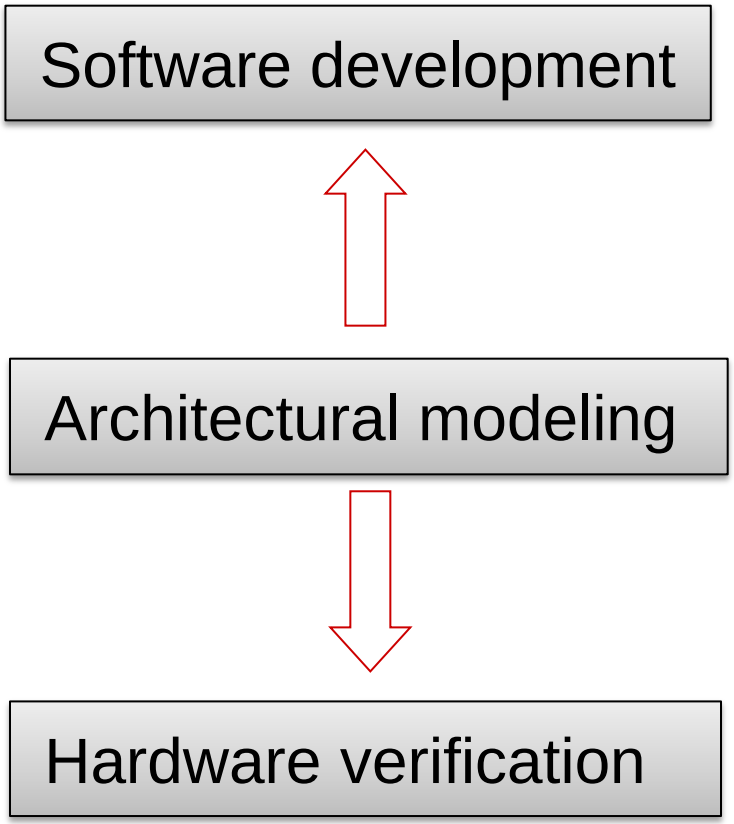

Simulate every event

100-10,000 X faster simulation

# Introduction(Cont'd)

*Accelerates product release schedule*




*TLM = golden model*

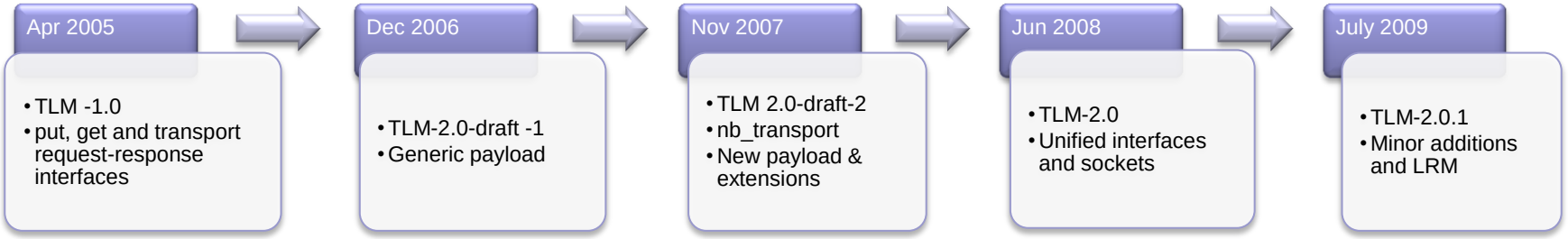
# Why is TLM interesting?

- Fast and compact
- Integrate HW and SW models
- Early platform for SW development, easy to distribute
- Early system exploration and verification
- Verification reuse

Fast!

Early!

# OSCI TLM Development



# TLM-1.0 → TLM-2.0

- TLM-2.0 is the new standard for interoperability between memory mapped bus models
  - Incompatible with TLM-2.0-draft1 and TLM-2.0-draft2
- TLM-1.0 is not deprecated (put, get, nb_put, nb_get, transport)
- TLM-1.0 is included within TLM-2.0
  - Migration path from TLM-1.0 to TLM-2.0

# More about TLM2.0…

- More details can be obtained from TLM_2_0_presentation.ppt

# Building TLM2.0 Libraries

- Similar to building SystemC libraries
  - 開啟code\TLM2.0_Example\include\TLM2.0.vcproj
  - 重建TLM2.0
  - 工具→選項→專案與方案→VC++目錄
    - 顯示目錄→Include檔案→加入 Lab3\TLM2.0_Example\include\tlm
    - 顯示目錄→程式庫檔→加入 Lab3\TLM2.0_Example\include\Debug
- OR, you can try the directory code\TLM2.0_Example\examples\tlm\build-msvc

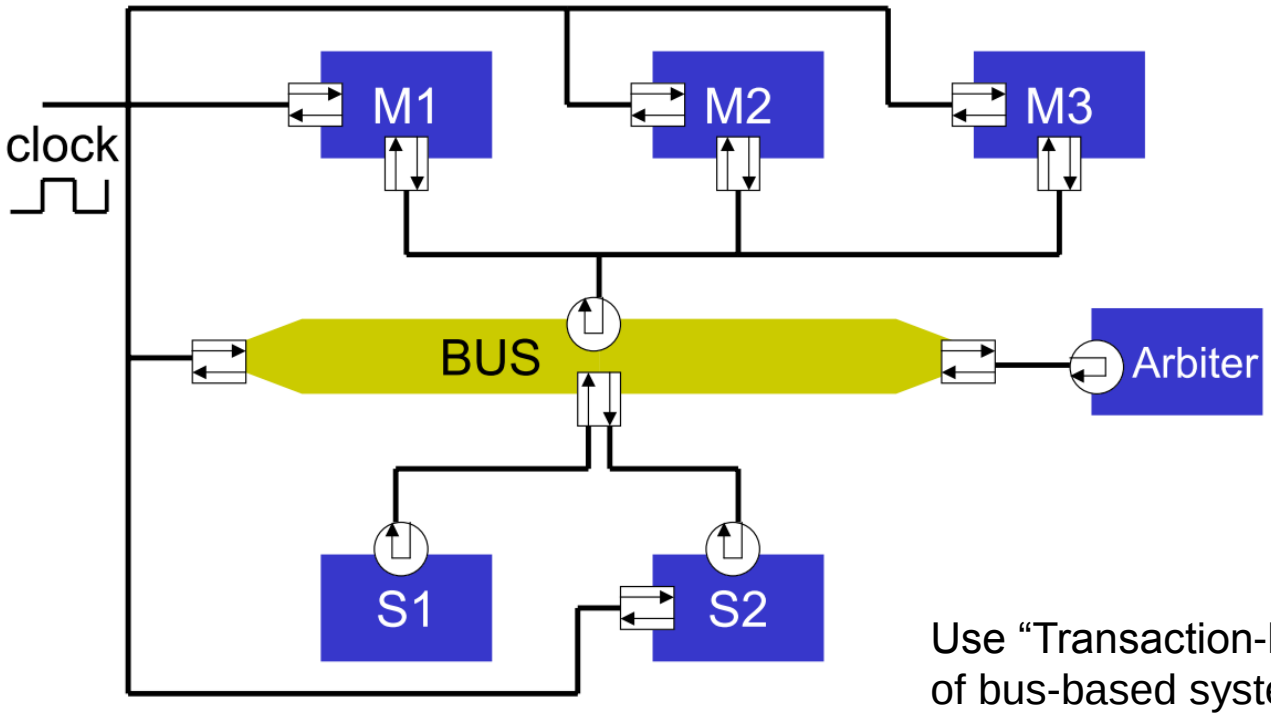
# Build Examples

- You can build and run examples just like building previous Labs and HWs.
- There is a directory build-msvc inside every example directory.
  - Follow the doc in the directory and build
  - The doc gives introduction to the example
- Don’t forget to build TLM2.0 first!

# Simple Bus

# Simple Bus example

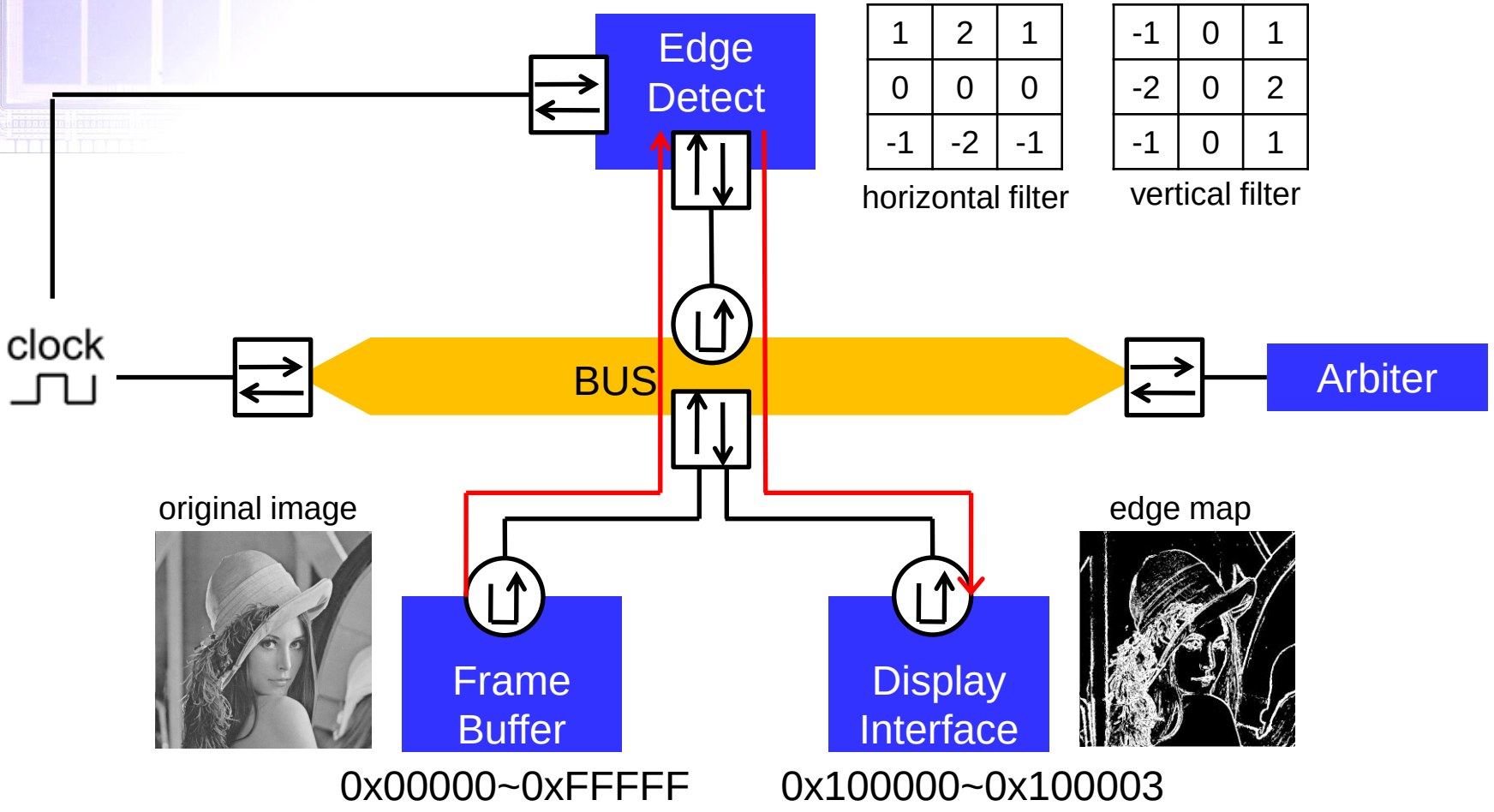
- Try code/simple_bus




Use “Transaction-level modeling of bus-based systems with SystemC 2.0” slide for the following.

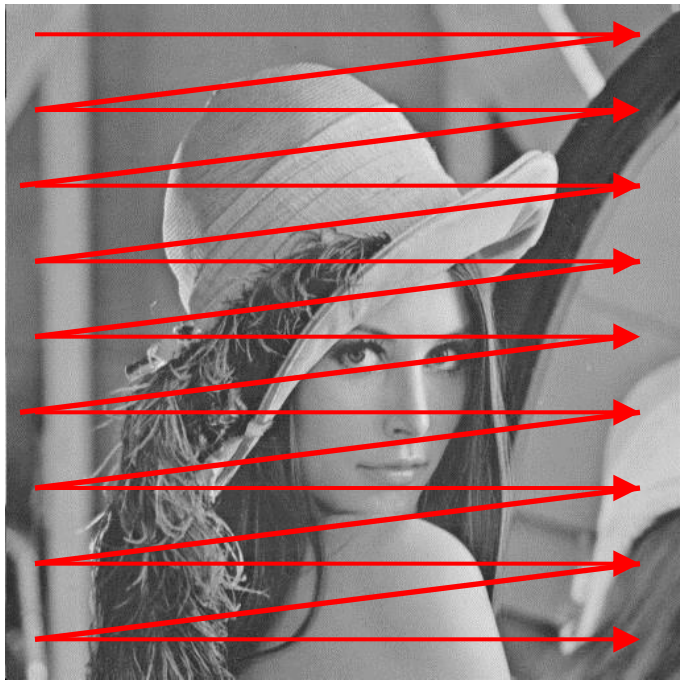
# Lab 3 Practice: Edge Detection

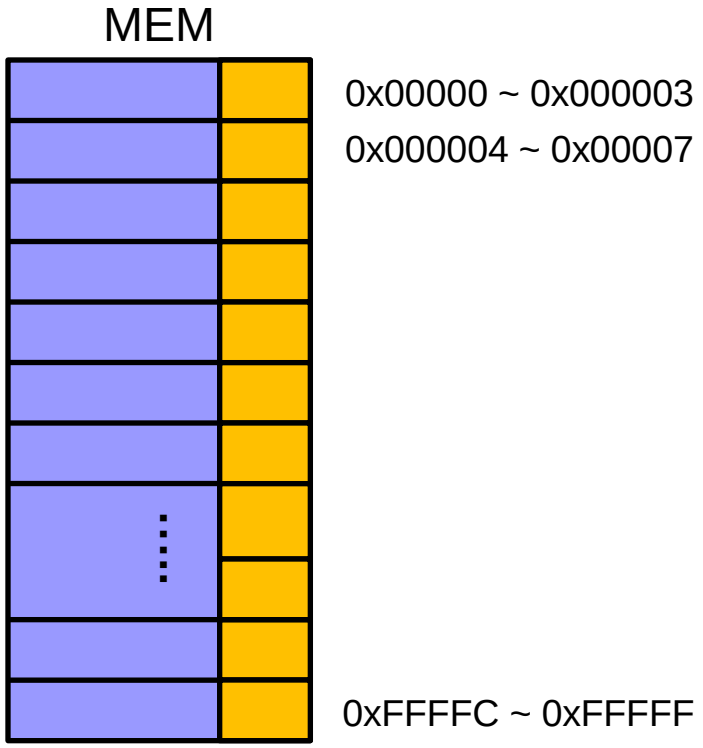
# Edge Detection



# Edge Detection(Cont'd)

- The bus use a unified 32-bit byte-address
- Image size : 512x512
- Arrangement of frame buffer





# Edge Detection(Cont'd)

- Window operation of edge filter

```
int mydata[9];
```

| | | |
|---|---|---|
| mydata[0] | mydata[1] | mydata[2] |
| mydata[3] | mydata[4] | mydata[5] |
| mydata[6] | mydata[7] | mydata[8] |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

horizontal filter

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

vertical filter

```
int edge = edge_filtering(mydata);
```

write the value to display interface !

# Edge Detection(Cont'd)

- Requirement
  - Use direct master interface to R/W
  - Complete the net connection
  - Complete the behavior of edge detect
  - Complete the behavior of frame buffer