# MSoC HW3 Report

R04943031 電子所一年級 林展翔

**[Problem 1]**

將 HW2 problem1 的架構(如 Fig.1 所示)中間插入 Simple Bus channel，並且加上 Master/Slave Wrappers，使得 Interchange Unit 和 Memory Unit 不需要變動。

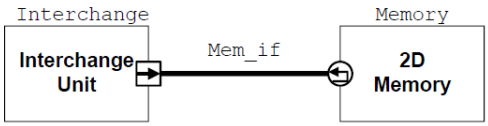


Fig.1 Original UT/AT model

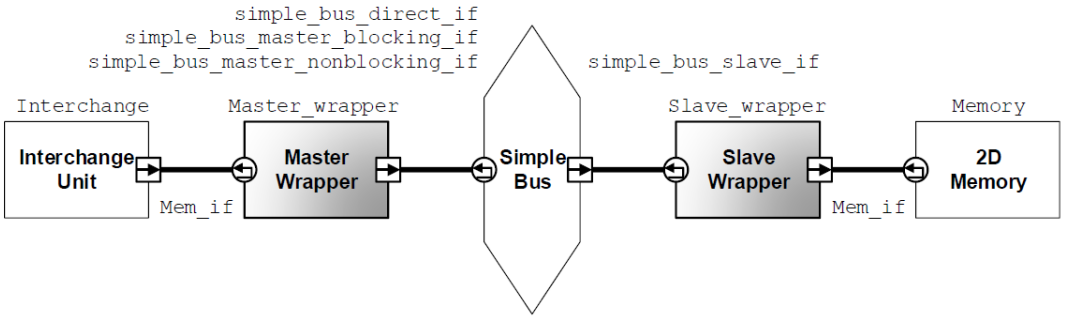


Fig.2 Port UT/AT IP model to SimpleBus

Master Wrapper 必須實作 *Mem_if* 介面，使得 Interchange Unit 能夠藉由介面函式對其送出/要求資料，同時它也要以 Simple Bus 提供的介面(此題要求用 *simple_bus_blocking_if*)與 Simple Bus 溝通。

```
void master_wrapper::word_read(unsigned x, unsigned y, int& d)
{
    unsigned int addr = 4 * (x * size + y);
    master_wrapper_out->burst_read(0, &d, addr);

}

void master_wrapper::word_write(unsigned x, unsigned y, int d)
{

    unsigned int addr = 4 * (x * size + y);
    master_wrapper_out->burst_write(0, &d, addr);
}
```

同樣地，Slave Wrapper 也必須實作 *simple_bus_slave_if* 介面，使用 *mem_if* 介面，以和前後 channel 溝通。

```
simple_bus_status slave_wrapper::read(int *data, unsigned int address)
{
    // retrieve data from size * size array, where each element is of size 4 bytes
    slave_wrapper_out->word_read(((address - m_start_address) / 4) / size,
        ((address - m_start_address) / 4) % size, *data);
    return SIMPLE_BUS_OK;
}

simple_bus_status slave_wrapper::write(int *data, unsigned int address)
{
    slave_wrapper_out->word_write(((address - m_start_address) / 4) / size,
        ((address - m_start_address) / 4) % size, *data);
    return SIMPLE_BUS_OK;
}

inline unsigned int slave_wrapper::start_address() const
{
    return m_start_address;
}

inline unsigned int slave_wrapper::end_address() const
{
    return m_end_address;
}
```

模擬輸出結果如下所示:

```
Original memory content
[ 33 45 16 91 99 29 80 89 ]
[  6 68 18 87 41  7 63 42 ]
[ 49 16 48 26  3 41 76 78 ]
[ 77 73 84 92 63  1 59 23 ]
[ 31 86 98 95 85 78 20 67 ]
[ 77  9 28 87 78 77 58 42 ]
[ 78 48 36 63 65 40 59 72 ]
[ 89 56 36 13 63 31 64 82 ]
Reading memory content
Writing memory content
Mirrored memory content
[ 82 72 42 67 23 78 42 89 ]
[ 64 59 58 20 59 76 63 80 ]
[ 31 40 77 78  1 41  7 29 ]
[ 63 65 78 85 63  3 41 99 ]
[ 13 63 87 95 92 26 87 91 ]
[ 36 36 28 98 84 48 18 16 ]
[ 56 48  9 86 73 16 68 45 ]
[ 89 78 77 31 77 49  6 33 ]
```

**[Problem 2]**

將 HW2 problem2 的架構(如 Fig.3 所示)中間插入 Simple Bus channel，並且加上 Master/Slave Wrappers，使得 Interchange Unit 和 Memory Unit 不需要變動。

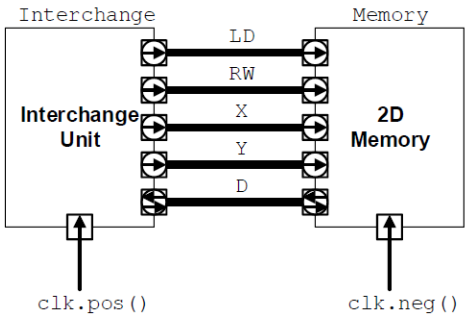


Fig.3 Original PCA model

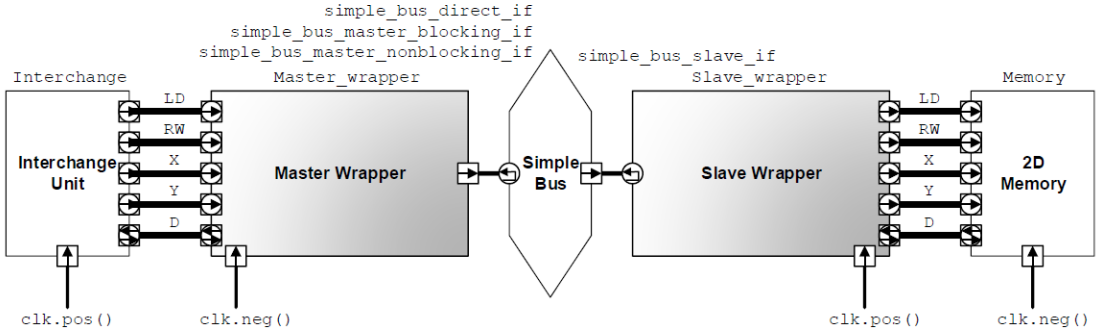


Fig.4 Port PCA IP model to SimpleBus

Master Wrapper 在此題中是以 input ports(LD、RW、X、Y、D)和 Interchange Unit 溝通，然而概念上還是跟用抽象介面時相同: 在讀取/寫入時呼叫 Simple Bus 介面達成任務。藉由輸入訊號的值可得知 Interchange Unit 想要進行的操作，再依此將 data port 的值傳給 Simple Bus 或將 Simple Bus 傳回的值輸出至 data port。

Slave Wrapper 還是以 *simple_bus_slave_if* 介面和 Simple Bus 溝通，不過與 2D Memory 的互動模式已經改為 pin/cycle accurate，必須將原本 mem_if 的讀寫函式以 RTL 的形式改寫。

這題很重要的部分是 Simple Bus 的 master 端(包括 Master Wrapper 和 Interchange Unit)必須等待 bus 已成功將資料讀取/寫入(會回傳 SIMPLE_BUS_OK)才能進行下個操作，否則會使用到錯誤的資料。

模擬輸出結果如下所示:

```
Original memory content
[ 33 45 16 91 99 29 80 89 ]
[  6 68 18 87 41  7 63 42 ]
[ 49 16 48 26  3 41 76 78 ]
[ 77 73 84 92 63  1 59 23 ]
[ 31 86 98 95 85 78 20 67 ]
[ 77  9 28 87 78 77 58 42 ]
[ 78 48 36 63 65 40 59 72 ]
[ 89 56 36 13 63 31 64 82 ]
Reading memory content
Writing memory content
Mirrored memory content
[ 82 72 42 67 23 78 42 89 ]
[ 64 59 58 20 59 76 63 80 ]
[ 31 40 77 78  1 41  7 29 ]
[ 63 65 78 85 63  3 41 99 ]
[ 13 63 87 95 92 26 87 91 ]
[ 36 36 28 98 84 48 18 16 ]
[ 56 48  9 86 73 16 68 45 ]
[ 89 78 77 31 77 49  6 33 ]
```