Introduction to AMBA Bus System

工研院 / 系統晶片技術中心工程師　吳欣龍

1. 前言

本篇文章主要是介紹ARM Limited公司所推出的AMBA協定（Advanced Micro-controller Bus Architecture）。AMBA協定目前是open且free的，讀者可從ARM的網站（www.arm.com）下載完整的Specification。

這篇文章並沒有打算說明完整的AMBA協定內容，詳細的Spec.還是請讀者閱讀ARM所提供的文件。原本的AMBA協定包含了四大部：AHB、 ASB、 APB、 Test Methodology，限於篇幅的關係，我們挑選較重要的AHB, APB加以基本的介紹，並探討AHB的一些重要的特性。

2. AMBA 概述

ISSN 1609-8633

AMBA協定的目的是為了要推出on-chip bus的規範，一開始AMBA 1.0只有ASB與APB，為了節省面積，所以這時候的bus協定都是tristate的bus，而到後來2.0的AHB為了能更方便設計者（trisate bus要花更多精力去注意timing），因此AMBA bus改用multiplexor的架構，並增加了新的特性。

一個以AMBA架構的SoC，一般來說包含了 high-performance的system bus - AHB與low-power的peripheral bus - APB。 System bus是負責連接例如ARM之類的embedded processor與DMA controller，on-chip memory 和其他 interface，或其他需要high bandwidth的元件。而peripheral bus則是用來連接系統的周邊元件，其protocol相對AHB來講較為簡單，與AHB之間則透過Bridge相連，期望能減少system bus的loading。一個典型的AMBA架構如圖2.1：

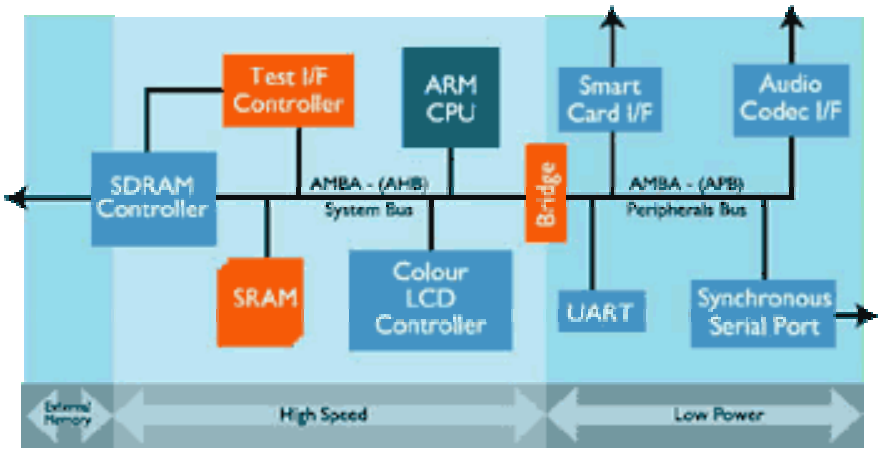

圖2.1

3. AHB簡介

ARM當初訂定AHB（Advanced High-Performance Bus）主要是想讓它能夠用來當作SoC的on-chip system bus，它的一些特性包括：

- single-clock edge operation
- non-tristate implementation
- burst transfers
- split transaction
- multiple bus master

以下我們將簡單的介紹AHB的協定及這些特性。

3.1 Overview

AHB System是由Master、Slave、Infrastructure三部分所組成。整個AHB bus上的傳輸(transfer)都是由master所發出，由slave負責回應。而

ISSN 1609-8633

infrastructure則由arbiter、master to slave multiplexor、slave to master multiplexor、decoder、dummy slave、dummy master所組成。

AHB 之所以會需要arbiter，是因為它支援multiple master，因此需要arbiter來仲裁。而decoder則是負責位址的解碼，從multiple slave中選擇要回應transfer的slave。而兩個multiplexor則是負責bus的routing（為了不使用tristate bus），將bus上的訊號在master和slave中傳送，圖3.1說明了multiplexor與master/slave連結的情形。

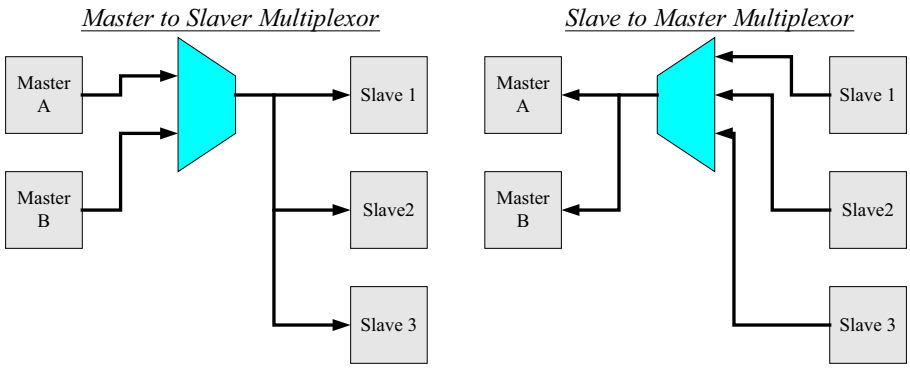

圖 3.1

基本上bus上傳輸的訊號，可以分成clock、arbitration、address、control signal、write data、read data、response signal七種。除了clock與arbitration訊號之外，其餘的訊號皆會經過multiplexor。會經過master to slave multiplexor的訊號有address、control signal、write data，而會經過slave to master multiplexor的則有read data與response signal。

下面的table列出所有的AHB訊號，以及它的用途。我們將在後面的章節介紹這些訊號，讀者可以先瀏覽一遍，有個基本的印象。

Name	Source	Description
HCLK	Clock source	Bus Clock。All signal timings are related to the rising edge of HCLK。
HRESETn	Reset controller	active LOW。reset whole system。
HADDR[31:0]	Master	32-bit system bus。
HTRANS[1:0]	Master	current transfer type。
HWRITE	Master	HIGH : write transfer。LOW : read transfer。
HSIZE[2:0]	Master	the size of the transfer。
HBURST[2:0]	Master	Indicates if the transfer forms part of a burst。
HPROT[3:0]	Master	Implement some level of protection。

ISSN 1609-8633

HWDATA[31:0]	Master	write data bus。
HSELx	Decoder	slave select signal。
HRDATA[31:0]	Slave	read data bus。
HREADY	Slave	High : transfer done。LOW : extending transfer。
HRESP[1:0]	Slave	transfer response。
HBUSREQx	Master	bus request。
HLOCKx	Master	Locked transfer。
HGRANTx	Arbiter	Bus grant signal。
HMASTER[3:0]	Arbiter	Indicate granted master number。
HMASTLOCK	Arbiter	Locked sequence。
HSPLITx[15:0]	Slave	Split completion request。

在看過了AHB的訊號後，下圖3.2則介紹AHB大概的bus interconnection。在圖3.2中，省略的部分有（1）各種control signal（HBURST、HTRANS等）的連接，其實他們的連線與HADDR一致。（2）Master與Arbiter之間的Request/Grant訊號。（3）Decoder與各個Slave間會有Selection的訊號。（4）control mux的output會有部分control訊號除了接到Slave外也會接到Arbiter（HTRANS/HBURST）。（5）Response signal（HREADY、HRESP）的mux。另外Arbiter 會輸出HMASTER訊號，這個訊號會接到master-to-slave multiplexor，以作為selection signal。

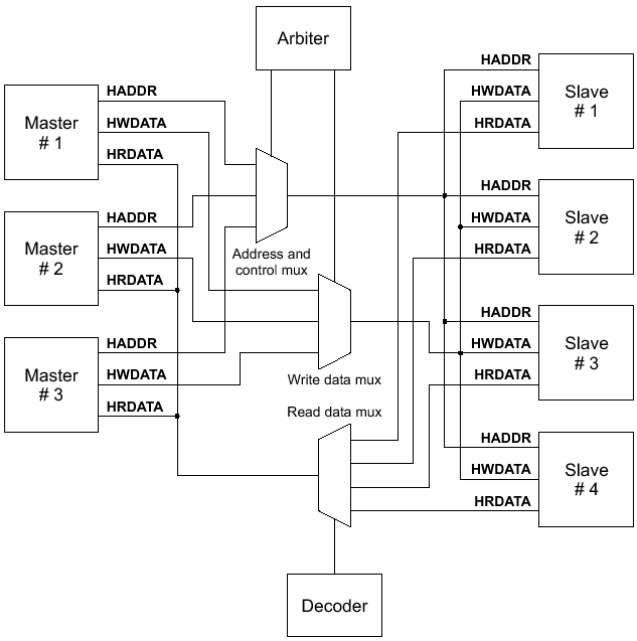

圖3.2

ISSN 1609-8633

3.2 **Basic transfer**

在AHB bus上，一次完整的transfer可以分成兩個phase：address phase與data phase。address phase傳送的是address與control signal，而data phase則是write/read data與response signal。下圖3.3說明AHB上的basic transfer。

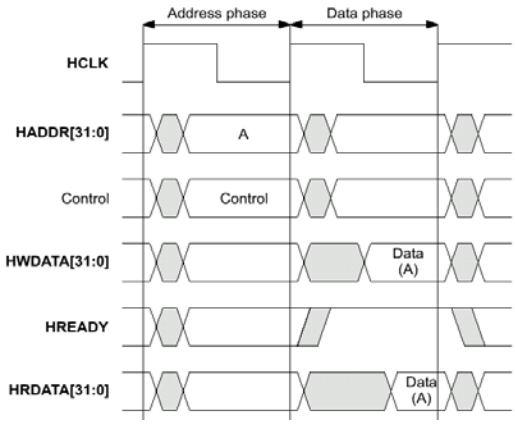

圖3.3

transfer在data phase時若無法在1個clock cycle內完成，slave可用HREADY訊號去延長（extend）transfer。請參考下圖3.4，當HREADY為LOW時，表示transfer尚未結束，為HIGH時，則代表目前的transfer結束了，但結束時的status則需看Slave回應的HRESP訊號（可能是OKAY、ERROR等）。

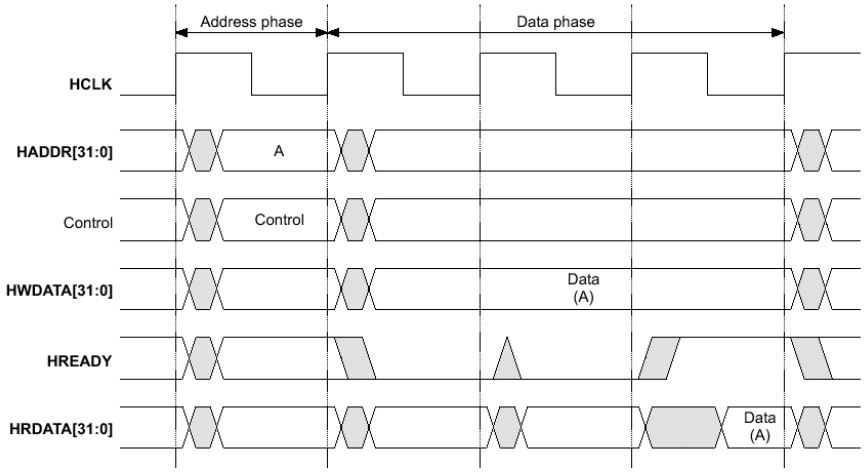

圖3.4

ISSN 1609-8633

由於一次transfer需要兩個phase才能完成，為了增進bus的performance，AHB將multiple transfer給pipeline起來，transfer間的address phase和data phase是overlap在一起的，請見下圖3.5。從圖3.5中我們可以看到由於現在目前transfer的data phase與下一次transfer的address phase是overlap的，所以當目前transfer的data phase被extend時，address phase也得跟著延長。

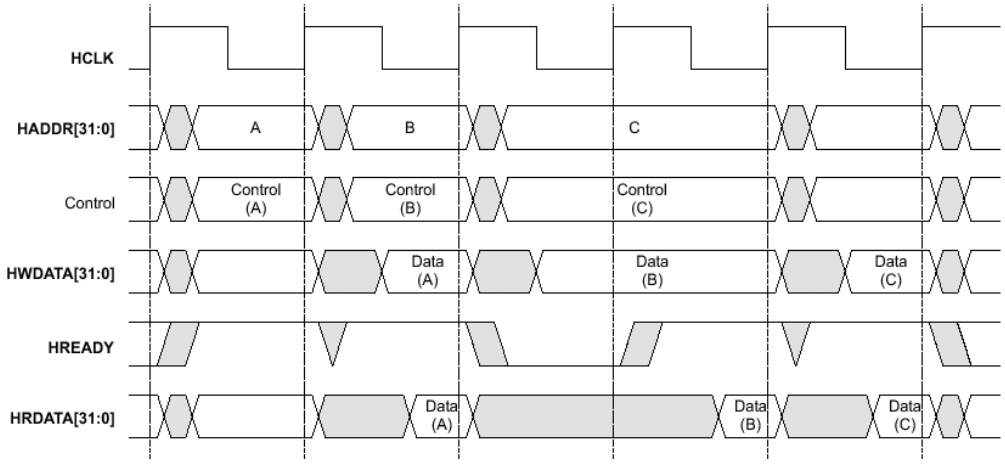

圖 3.5

3.3 **Control signal**

AHB上的Control signal 共有五類，分別為：

- HTRANS[1:0]：Transfer Type
- HBURST[2:0]：Burst Type
- HPROT[3:0]：Protection Control
- HSIZE[2:0]：Transfer Size
- HWRITE：Transfer Direction

底下我們將一一介紹。

3.3.1 **Transfer Type**

AHB上共有四種transfer type：

- IDLE：指示slave需忽略目前的transfer。用於當master沒有資料需要傳送時，而此時slave需在transfer的data phase回應zero wait cycle的OKAY response。
- BUSY：在burst transfer時，master傳送連續的transfer給slave，若master因某些原因無法及時將資料準備好，則發出使用此 transfer type通知slave，slave的response應該與回應IDLE transfer時相同，也就是zero wait cycle的OKAY response。
- NONSEQ(Non-sequential)：指示目前transfer

ISSN 1609-8633

的adress和control訊號與上一transfe關。

●SEQ（Sequential）：指示address和上一筆transfer相關，而control訊號則和上一筆transfer相同，通常用在burst transfer中。

下圖3.6為transfer type的example。從時序圖裡我們可以看出：第一筆burst transfer的type一定為NONSEQ，另外因為master無法把下一筆資料在第二個cycle準備好，因此使用BUSY type去延遲第二筆transfer。

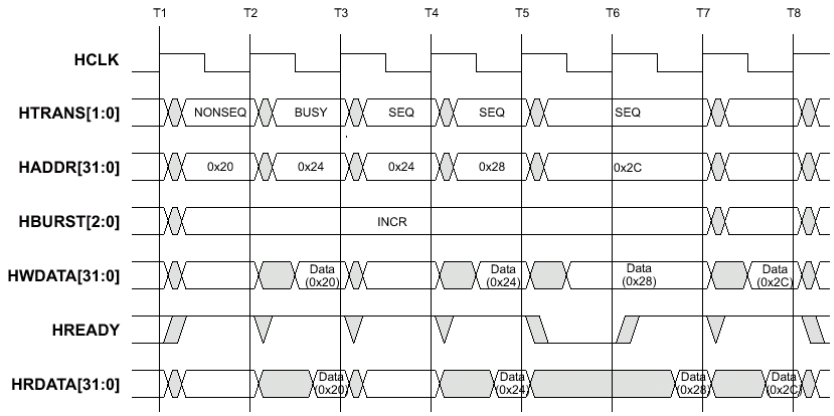

圖3.6

3.3.2 **Burst Type**

Burst type是用來讓AHB master發出address彼此相關的連續transfer（control訊號需相同）。AHB支援八種的burst type，用來指示burst的長度（transfer的個數，在AHB Spec.中使用beat這個英文字），與address間的關係。請見表3.1。其中incrementing的burst，每一筆的transfer address必定是前一筆transfer的address加上transfer size。而wrapping burst則將memory切割成了（transfer size X transfer beat）大小的一個個memory boundary，當transfer address要跨越boundary時，下一筆transfer address會繞回boundary起點。舉例來說，現在我們要傳送4筆wrapping burst，transfer size為word（4 byte），第一筆transfer的address為0x34，此時（4 byte x 4 transfer）則transfer會在16-byte boundary繞回，所以4筆transfer的address分別是0x34, 0x38, 0x3C, 0x30。

下面列出AHB支援的八種transfer type。

ISSN 1609-8633

HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

表3.1

圖3.7 介紹4-beat的wrapping burst，由於transfer size為word，所以address為在16-byte boundary繞回，在圖3.7中我們可以看到0x3C之後的位址就變成0x30。

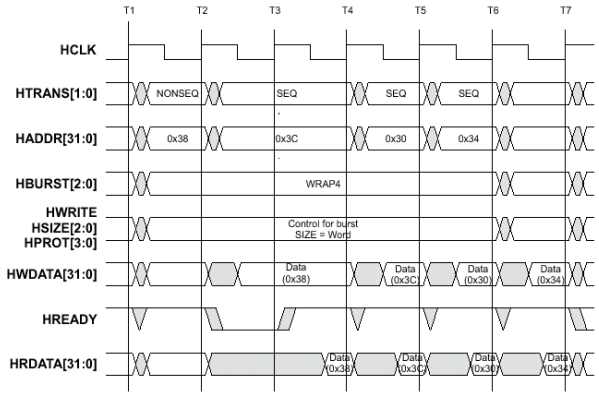

圖3.7

相對於圖3.7，在圖3.8中，因為是INCR type，所以address 0x3C之後會跨過16-byte boundary，直接到0x40。

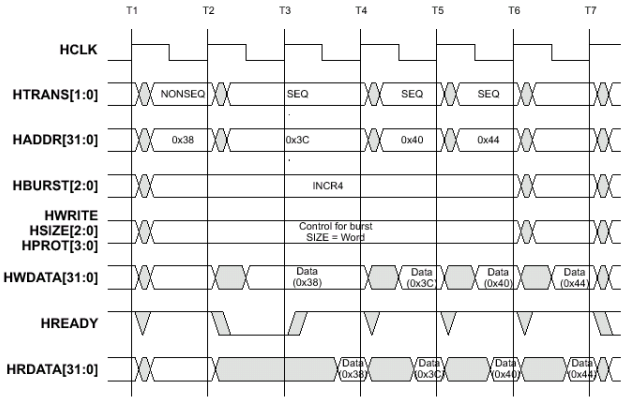

圖3.8

ISSN 1609-8633

圖3.9則是8-beat wrapping burst的例子，這次memory boundary為32-byte，所以0x3C後會繞回0x20。

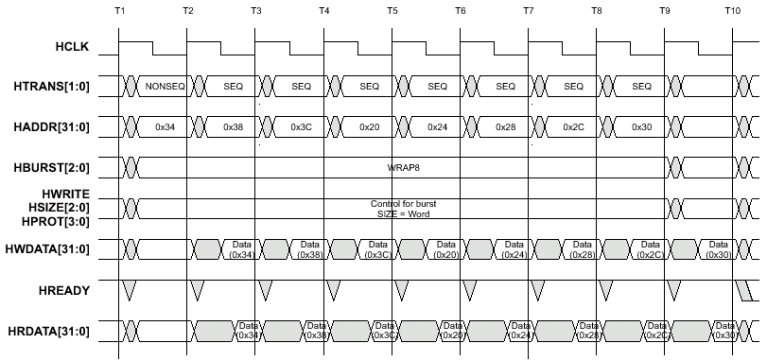

圖 3.9

圖3.10則是8-beat incrementing burst，不過這次的transfer size為Halfword。

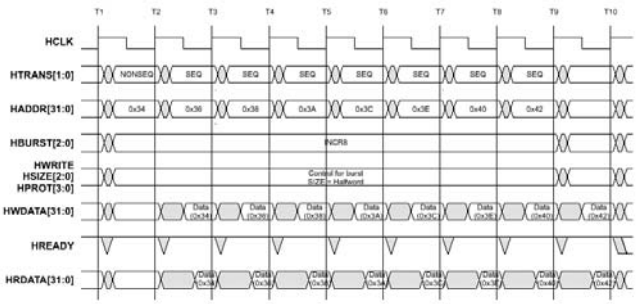

圖 3.10

最後圖3.11介紹兩筆undefined length burst的例子，而transfer size一筆是Halfword，另一筆是word。

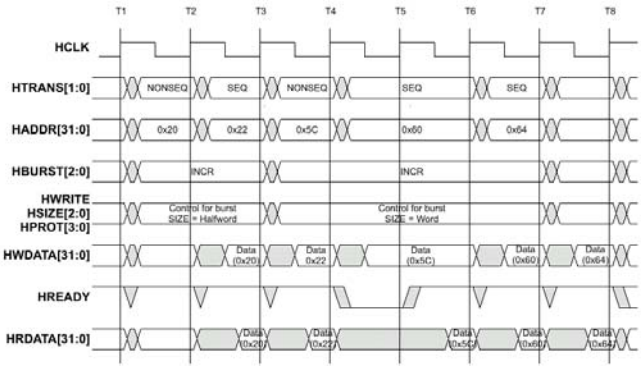

圖 3.11

ISSN 1609-8633

3.3.3 **Transfer Direction**

當HWRITE為HIGH，則master會在data phase時將資料放在write data bus HWDATA[31:0]，讓slave去sample 資料。當HWRITE為LOW時，slave會在data phase將資料放在read data bus HRDATA[31:0]，讓master去讀取資料。

3.3.4 **Transfer Size**

AHB共支援八種的transfer size，請參考表3.2。

HSIZE[2]	HSIZE[1]	HSIZE[0]	Size	Description
0	0	0	8 bits	Byte
0	0	1	16 bits	Halfword
0	1	0	32 bits	Word
0	1	1	64 bits	-
1	0	0	128 bits	4-word line
1	0	1	256 bits	8-word line
1	1	0	512 bits	-
1	1	1	1024 bits	-

表3.2

3.3.5 **Protection Control**

HPROT[3:0]可以讓master提供額外的protection information，譬如：指示transfer是opcode fetch或data access等。下表3.3為AHB所支援的protection。由於AHB Spec.並未規定所有的master都要指示精確的protection information，所以slave在設計時若非必須，盡量不要使用HPROT訊號。（若master沒有protection transfer的考慮，HPROT 可以output 4’b0001訊號）

Table 3-4 Protection signal encodings

HPROT[3] cacheable	HPROT[2] bufferable	HPROT[1] privileged	HPROT[0] data/opcode	Description
-	-	-	0	Opcode fetch
-	-	-	1	Data access
-	-	0	-	User access
-	-	1	-	Privileged access
-	0	-	-	Not bufferable
-	1	-	-	Bufferable
0	-	-	-	Not cacheable
1	-	-	-	Cacheable

表3.3

ISSN 1609-8633

3.4 **Slave Response**

在AHB協定中，slave除了可以使用HREADY訊號去extend transfer（插入幾個wait cycle）外，在transfer結束時（HREADY在data phase為high），Slave還可以使用HRESP[1:0]去告訴master transfer結束時的status。在AHB裡transfer結束時可以表示的status共有四種，OKAY、ERROR、RETRY、SPLIT。

四個status中，OKAY表示transfer成功的完成了，ERROR表示transfer失敗了，失敗的可能原因，譬如說企圖寫入read-only的memory location，或讀寫根本不存在的memory location等。而RETRY和SPLIT則是用在當slave判斷目前的transfer將需要很多的bus cycle來完成，為了避免因為目前的transfer將bus一直lock住，而回應RETRY/SPLIT response給master，表示目前的transfer尚未完成，master需要重新發出相同的transfer再試一次，而此時arbiter就能將bus release給其他有需要的master使用。至於Retry和Split的差別在於arbiter 的master優先權管理（Priority Scheme）：

- RETREY response：arbiter內master的優先權不變，當有更高優先權的master發出request時，bus access的權力會由高優先權的master取得，但如果原來得到RETRY response的master是當時request bus的master中擁有最高優先權者，則bus還是會繼續被佔住而無法release給其他有需要的master。
- SPLIT response：當arbiter觀察到master收到SPLIT response時，則會將master的優先權給mask起來，當mask後，master將無法再獲得bus access的權力，即使已經沒有其他master向arbiter 發出request也一樣。若所有的master都收到SPLIT response，則arbiter會把bus access的權力給dummy master （只會發出IDLE transfer的master）。當回應SPLIT的slave處理完transfer的要求後，會發出HSPLIT訊號給arbiter，則arbiter會將master的優先權unmask，如此master的優先權就回復原狀而有機會access bus了。

SPLIT response與RETRY response比起來能讓低優先權的master在合理的情況下（無更高優先權master要求bus時）取得bus的擁有權，因此可以讓AHB bus有更好的設計，但缺點是硬體設計的複雜。首先arbiter必須要去觀察HRESP訊號，且要有當收到SPLIT時更動master優先權的設計。再者Slave需要紀錄master的number以便以後要通知arbiter恢復那個master的優先權，Slave可以從arbiter發出的HMASTER訊號查得。在AHB系統裡最多可以有16個master，所以HMASTER是四個bit，然而通知arbiter可以unmask的訊

ISSN 1609-8633

號HSPLIT因是one-hot的表示方式，所以需要16 bit。

另外要注意的是AHB並沒有強制規定Slave需支援Retry或Split response，Slave可以只用HREADY去delay對transfer的回應（delay的時間無上限值，但Spec.中建議不要超過16個cycle）。然而master則需要支援對Retry或Split response的處理：重新再發出相同的transfer。

這四個response中，除了OKAY response只需one -cycle之外，其餘的三個response都需要two-cycle去完成。在這兩個cycle中HRESP維持想要回應的status不變（ERROR or RETRY or SPLIT），而HREADY則在第一個cycle為low，第二個cycle為HIGH。

之所以需要兩個cycle，這是因為AHB 為pipeline operation、current transfer的data phase與next transfer的address phase是overlap的。而這三種Response皆可能因為目前的transfer並未成功而影響了下一個transfer的存取，使的master 需要cancel預備要進行的下一個transfer （在AHB的規定裡，因為Retry/SPLIT是代表目前transfer尚在處裡，所以下一個transfer必須一定要取消，而ERROR response表示transfer failed，在某些情況下，master仍然會嘗試下一筆transfer，因此當master 收到ERROR response時，繼續存取下一筆transfer是允許的，不過ERROR response此時還是需要花two-cycle）。我們使用下圖3.12來說明這個情形：

在圖3.12裡，我們可以看到master原本預備進行address A/A+4的兩個transfer，可是在transfer A的data phase（同時也是A+4 transfer的address phase）的第一個cycle，master偵測到retry的response，由於這個cycle的HREADY為LOW，不僅將data phase給extend，同時也表示address A+4的transfer並沒有開始，所

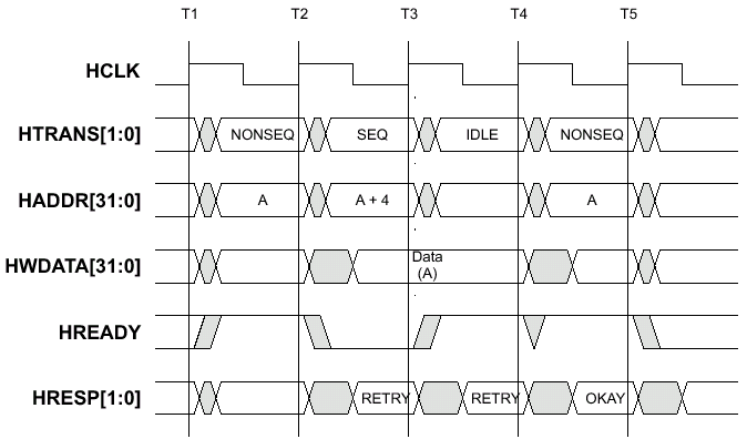

圖3.12

ISSN 1609-8633

以master才有機會在下個cycle將address A+4的transfer給替換成IDLE transfer。如果RETRY response只有one-cycle，則transfer A收到retry 的同時，transfer A+4也開始進行了，而它很有可能也收到RETRY response（因為slave還在處理transfer A），如此下去，以後的transfer可能都無法完成了。

在RETRY response的第二個cycle裡，我們看到master改成發出IDLE transfer，這其實是為了讓arbiter能順利的完成arbitration，我們在下一節會有詳細的介紹。

在slave決定要回應何種response給master前，slave可能會需要多幾個wait cycle去衡量，此時的wait cycle除了將HREADY給drive LOW之外，還需要將HRESP給drive成OKAY response。下圖3.13顯示出負責回應transfer A的slave在回應ERROR response之前還多extend一個cycle，而此時的response為OKAY。

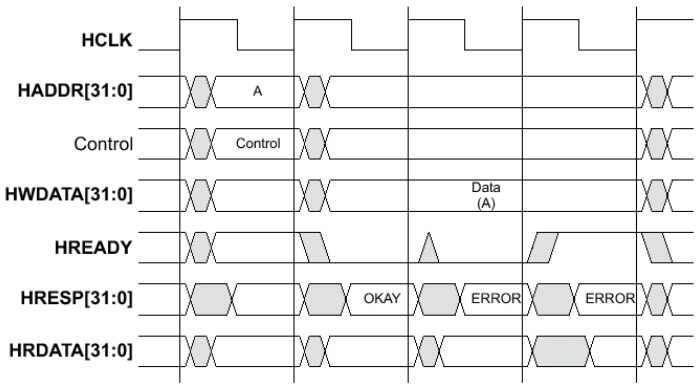

圖3.13

3.5 **Arbitration**

由於AHB為Multi-Master的系統，而同一時間只允許一個master去access bus，因此需要arbiter去做Arbitration，底下我們先簡述AHB Arbitration的機制：

當master想要access bus時，master將HBUSREQ signal給drive high（每個master都有自己的HBUSREQ訊號），同一時間可能有多個master都想要access bus，因此arbiter在HCLK的rising edge 去sample各個master的HBUSREQ訊號後，需決定那個發出request的master有最高的priority（AHB並無規定priority algorithm，由系統設計者自訂），然後將此master的HGRANT訊號drive high，表示他可以access bus了。（若原本已有master在access bus，arbiter會將原本master的HGRANT訊號給

ISSN 1609-8633

drive LOW表示他已喪失access的權力了）

當master正進行fixed-length burst transfer時，如果有更高priority的master發出了request， arbiter可以等待burst完成後再將bus grant給新的master，也可以在burst進行中，就中斷原有master的bus擁有權（ownership），讓高priority的master去access bus。而被中斷的master就需要重新發出request，等待下次Grant bus時繼續完成burst了。

若master想要進行的連續transfer是不可被中斷的（譬如在access shared memory時），則master可以在request時，同時將HLOCK給drive high，告訴arbiter我要進行的是不可被中斷的transfer，則當master獲得access bus權力後，arbiter將不會再把bus release給其他master，直到master自行將HLOCK給drive LOW，arbiter才會再進行arbitration的動作。

下圖3.14顯示出基本的bus handover的情形，從這張圖裡我們可以看到HMASTER會顯示出bus所有者的master number（每個master在arbiter裡皆有一個number），另外因為pipeline operation的原因，master獲得data phase的擁有權時會比address phase 延遲一個cycle，所以當master獲得bus時第一個transfer的address phase是與前一個master的data phase overlap在一起的。而當前一個master的data phase被slave 延遲時，現在擁有address bus的master也跟著被延遲了，我們將在後面的時序圖裡介紹這樣的情形。

當master的HGRANT被assert後，且HREADY為High，則代表master在下個cycle就可以access bus了。若HREADY的訊號一直為LOW，而此時arbiter接到更高priority 的master的request而更改了Grant訊號，那麼原本被grant的master就沒有access的權力了。下圖3.15顯示出因為HREADY訊號而延遲了GRANT的時間。

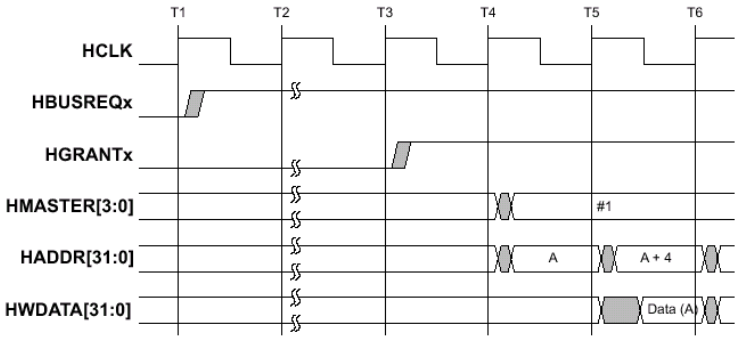

圖3.14

ISSN 1609-8633

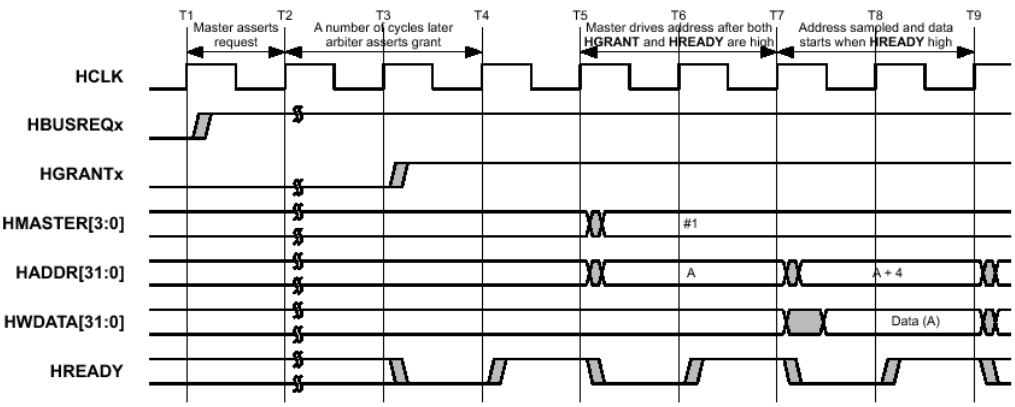

圖 3.15

下圖3.16主要在介紹Data Bus在bus handover時轉移的情形，及address bus因前一個master的transfer被延遲的情形。

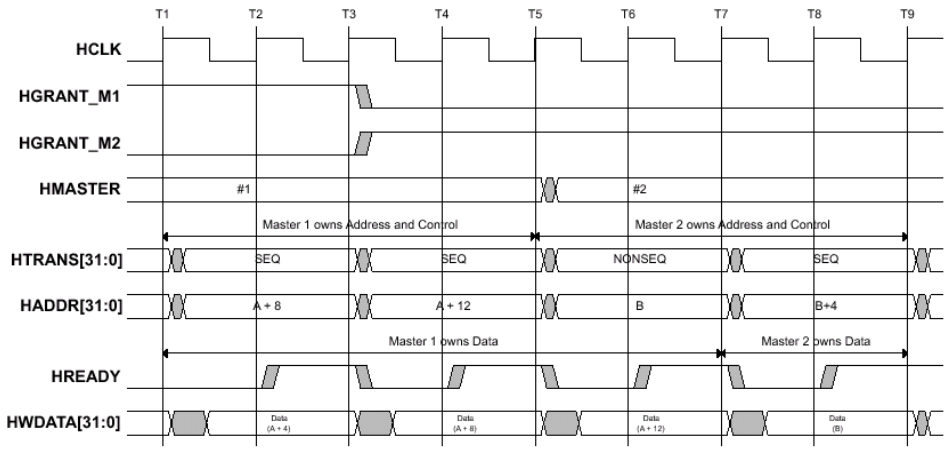

圖 3.16

圖3.17主要在說明HGRANT在bus handover時變化的情形。以burst transfer的情形來說，當arbiter決定在burst結束後，轉換bus 的擁有權時，arbiter會在倒數第二個transfer的address被sample後，改變HGRANT訊號，所以新的HGRANT訊號將跟最後一個transfer的address一起在同一個時間被sample。

在前一節介紹RETRY/SPLIT response時，我們曾經介紹其為two-cycle的response，且第二個cycle必須是IDLE transfer以便讓arbiter有機會去改變bus owner，下圖3.18就介紹

ISSN 1609-8633

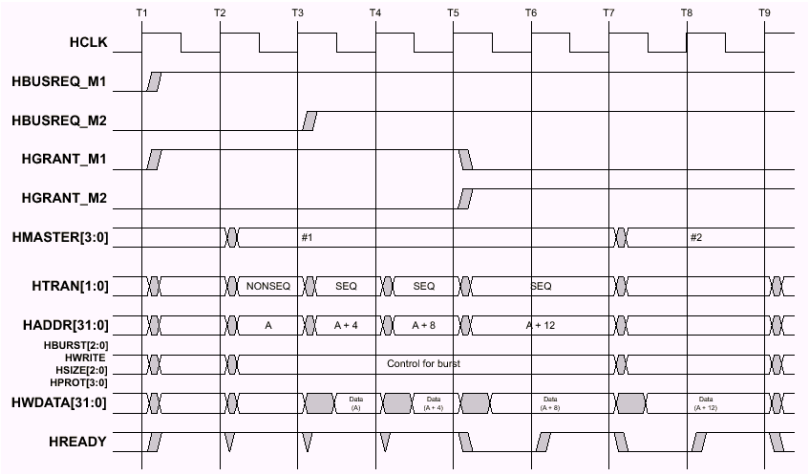

圖3.17

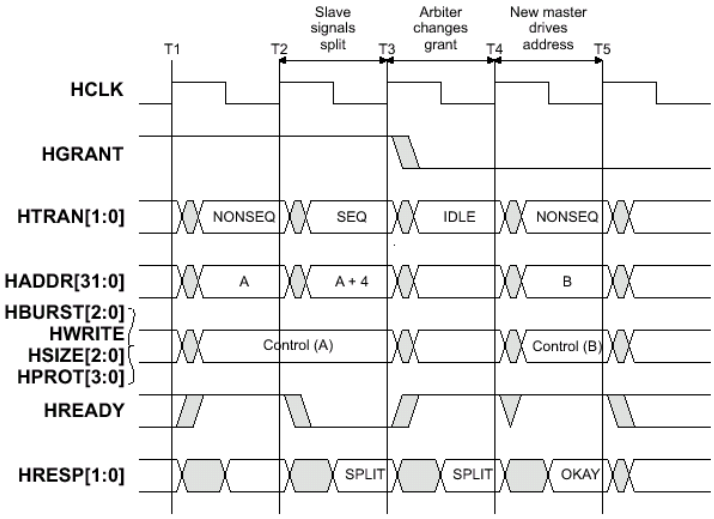

圖3.18

HGRANT訊號如何利用IDLE transfer的cycle去變化。

4. APB 簡介

APB主要是用在連接low-bandwidth的周邊上面，例如UART、1284等。它的Bus架構不像AHB為Multi-Master，在APB裡唯一的master就是APB Bridge（與AHB Bus相接），因此不需要arbiter以及一些request/grant訊號。APB協定十分簡單，甚至不是pipeline operation，底下為APB的特性：

- always two-cycle transfer
- no wait cycle & response signal

ISSN 1609-8633

4.1 APB Overview

我們先列出APB的訊號名稱與功用，下表4.1為所有的APB訊號：

Name	Description
PCLK	Bus clock，rising edge is used to time all transfers.
PRESETn	APB reset。active Low.
PADDR[31:0]	APB address bus.
PSELx	Indicates that the slave device is selected. There is a PSELx signal for each slave.
PENABLE	Indicates the second cycle of an APB transfer.
PWRITE	Transfer direction. High for write access, Low for read access.
PRDATA	Read data bus
PWDATA	Write data bus

APB上的transfer可以用下面的state diagram來說明：

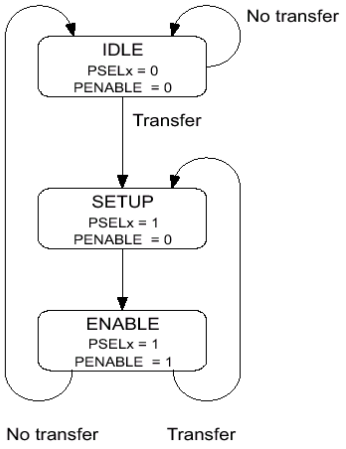

圖 4.1

- 一開始為IDLE state，此時並沒有transfer在進行，也沒有slave被選擇到（PSELx=0）。
- 當有transfer要進行時，PSELx=1、PENABLE=0，進入SETUP state，而且只會在SETUP state停留one-cycle，當PCLK下一個rising edge時則進入ENABLE state。
- 在進入ENABLE state時，之前在SETUP state的PADDR、PSEL、PWRITE皆維持不變，只有PENABLE被assert。transfer也只會在ENABLE state維持one-cycle，transfer在經過SETUP與ENABLE state後就算完成了，之後若沒有transfer則又進入IDLE state，若有連續的transfer則進入SETUP state。

4.2 Write Transfer

圖4.2介紹基本的write transfer。

ISSN 1609-8633

在圖4.2裡，時間T2->T4為一個APB的write transfer（記住APB transfer是always two-cycle），第一個cycle：T2->T3為SETUP Cycle，第二個cycle：T3->T4為ENABLE cycle。在整個transfer裡，address/control/data 訊號都需維持不變。

在transfer結束後，PENABLE訊號一定會deasserted（go LOW），而PSELx訊號則視情況而定，若有transfer要接著對同一個slave進行，則維持不變（HIGH），反之，則會被dirve Low。

另外為了節省power，若接下來無transfer需要進行，address/write 訊號會一直維持不變，直到又有transfer發生。

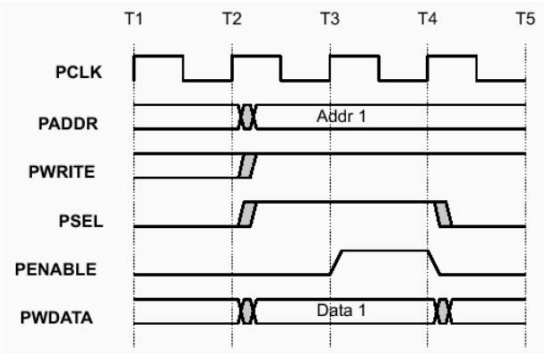

圖4.2

4.3 **Read Transfer**

圖4.3為read transfer的時序圖，除了PWRITE 為LOW外，其餘的address/select/strobe訊號時脈皆與write transfer相同。在read transfer時，slave必須在ENABLE cycle提供data給APB Bridge在T4的時間點sample。

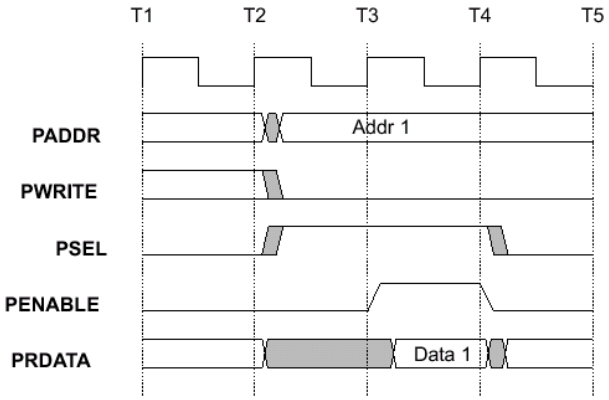

圖4.3

ISSN 1609-8633

4.4 APB Slave與APB bridge

當我們要設計APB slave時，可以選擇在下面兩種情況之一去latch write data：

- 在PSEL為High時的任一cycle
- 當PSEL為HIGH時，PENABLE的rising edge

對於read data，slave則可以在PWRITE為LOW而PSEL、PENABLE皆為HIGH時，去drive read data bus。

經由以上的介紹，我們可以瞭解APB Bus的protocol十分的簡單，但要設計一個有效率的 APB Bridge，將複雜的AHB transfer轉成APB transfer則不太簡單，在APB的Spec.裡有介紹轉換時的時序圖，若讀者對這部分有興趣的話，請自行參考Spec.。

5. AHB重要特性探討

在介紹過AHB與APB後，我們將針對AHB的一些重要或特別的性質加以補充說明。

5.1 **Address Decoding**

在AHB系統中，有一個central address decoder，提供HSELx訊號到各個AHB Slave，這個decoder本身只負責位址的解碼，所以是純combination 的電路。圖5.1中，顯示出系統decoder與HSEL的連接關係。

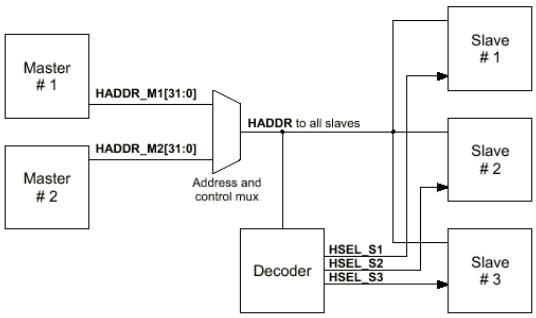

圖 5.1

從之前對於bus handover的介紹，我們可以知道在bus handover的過程中可能出現current master第一筆transfer的address phase被前一個master的data phase所delay（HREADY為Low），因此AHB Slave在設計時要特別注意只有在HSELx與HREADY皆為High的情形下去sample address/control訊號。

由於AHB的address line有32條，為了簡化decoder解碼所需的時間，AHB Spec.規定每個Slave最小的address space也有1 KB，在這種情況下，decoder最多只需要對22條位址線進行解碼。

為了避免AHB master在進行

ISSN 1609-8633

incrementing burst transfer時不小心跨越了slave address的boundary（這時會選擇到另一個slave），因此AHB Spec.規定所有的master在做burst transfer時，address皆不可以跨過1KB boundary，當有需要跨越時，則要用新的一筆transfer去跨越（也就是transfer type要從NONSEQ開始）。

5.2 Default Master & Dummy Master

在AHB系統裡有default master與dummy master的存在，其中default master為系統中的正常master，有HBUSREQ訊號接到arbiter。當系統中沒有一個master向arbiter發出request時，此時arbiter會把bus的擁有權給default master，用意是當default master需要向arbiter發出request時可以減少bus handover的cycle，所以一般default master通常是access bus頻率最高的master。（注意：當master沒有request bus而被grant 時需發出IDLE transfer）

而dummy master則沒有HBUSREQ訊號接至arbiter，它被arbiter grant bus的時機有兩個，一個是當所有master都收到slave的SPLIT response而不能access bus時，另一個是當有master在進行locked transfer時收到SPLIT response，此時dummy master就會被grant bus，而dummy master也只會不斷的發出IDLE transfer來維持AHB系統的正常運作。

5.3 Multiple SPLIT

在AHB Spec.中規定每個master一次只能處理一筆transfer，若transfer被SPLIT會RETRY，則master仍能只能等待transfer的完成。所以若有device想要在transfer被SPLIT或RETRY後，還可以去進行其他transfer（向其他slave去存取資料），理論上是不被允許的，除非device本身有多個master的interface，則可以利用其他master interface的HBUSREQ訊號去向arbiter request bus。

在AHB Spec.裡規定當SPLIT-capable Slave在處理被回應SPLIT response的transfer時，其他master仍然可以去access同一個Slave，而此時Slave只要記錄來access的master number後，（不需記錄address/control等資訊），仍然回應SPLIT response。等transfer處理完畢，再把之前前來access的其他master相對應的HSPLITx訊號drive HIGH，如此一來，master將會再次傳送transfer的address和control訊號過來。但是這種情況也表示master有可能要收到SPLIT response好幾次後才能完成一個transfer。

5.4 Multiple RETRY

ISSN 1609-8633

相對於Multiple SPLIT，AHB Spec.中規定回應Retry的Slave，在它處理完transfer並由master來存取前，不能再被不同的master給access，至於預防的方法需由系統設計者自己去解決（可能在OS 層次去預防）。如果真的發生在RETRY中被不同master所存取（Slave可以藉由記錄master number去辨別），AHB Spec.提供以下四個可能的解決辦法：

- 回應ERROR response
- 通知arbiter
- 發出中斷
- Reset整個系統

5.5 SPLIT-Capable Salve 設計

為了避免Slave使用HREADY將transfer extend太長甚至將bus整個鎖死，AHB Spec.規定每個Slave要先定義好最長的wait cycle是多少（spec.的建議值為16個cycle）。

在Multi-SPLIT中我們談到可能有多個master去access正忙於處理SPLIT transfer的slave，此時slave需記下master number，而在一個AHB系統中最多可以有16個master，所以SPLIT-capable的slave需要準備16組register去記錄master number。

另外如果當slave處理SPLIT-transfer中有master發出locked transfer給slave，此時slave要優先處理locked transfer（甚至要中斷之前處理中的transfer），以免造成整個bus被locked transfer給鎖死的情形。

5.6 **AHB Lite**

在某些AHB系統中可能只有一個master，此時系統中仍能使用arbiter就顯得不必要了，因此ARM在AMBA 2.0之外推出AHB 的變化型- AHB Lite。與AHB的不同在於少了arbiter與HGRANT/HBUSREQ訊號機制，以及Slave不能再回應RETRY/SPLIT response（因為不可能有其他master去access bus了）。下圖5.2為AHB-Lit系統的block diagram。

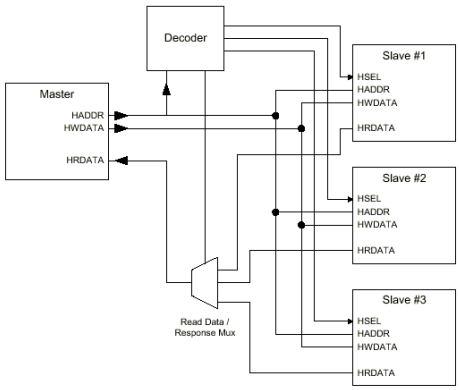

圖 5.2

ISSN 1609-8633

若原本master就設計成full-AHB的interface與功能，則不需任何改變就可以在AHB-Lite與full-AHB中應用。相反的若原本是設計給AHB-Lite的master則需要簡單的wrapper才能在full-AHB中應用。

若Slave原本不具SPLIT/RETRY的能力，則在full-AHB與AHB Lite中皆可應用，但若是具有SPLIT/RETRY能力的slave則一樣要wrapper才能使用在AHB Lite中。

底下我們列出所有AHB Lite與full-AHB的不同。

- 只有一個master，且不需Master to Slave Multiplexor。
- 沒有arbiter。
- Master沒有HBUSREQ訊號，若有則將訊號floating。
- Master沒有HGRANT訊號，如果有則tied HIGH。
- Slave不能產生SPLIT/RETRY response。
- AHB-Lite的lock 訊號，其timing要跟full-AHB的HMASTLOCK的timing一樣。

5.7 Multi-layer AHB

在AHB系統中，若有兩個master常需要access bus，則系統的performance必定會下降，為了解決這個問題，ARM推出了Multi-layer AHB，其基本構想如下圖5.3。

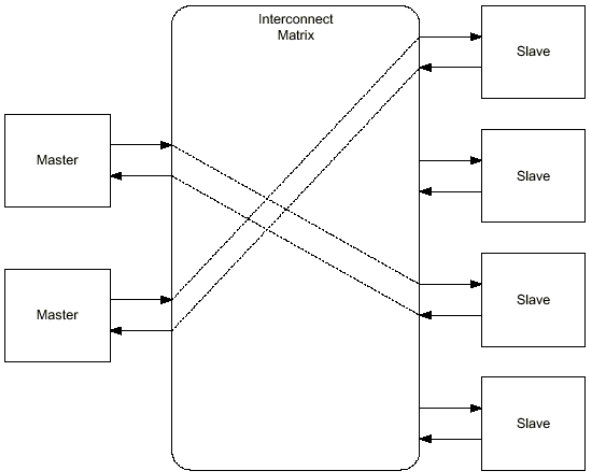

圖5.3

圖5.3 的基本構想是兩個master走不同的bus去access slave，若access的slave不同，則兩個master可以同步的進行transfer。若彼此access同一個slave則由slave去判斷要先處理誰的transfer。圖5.4則顯示connection matrix的內部細節。

ISSN 1609-8633

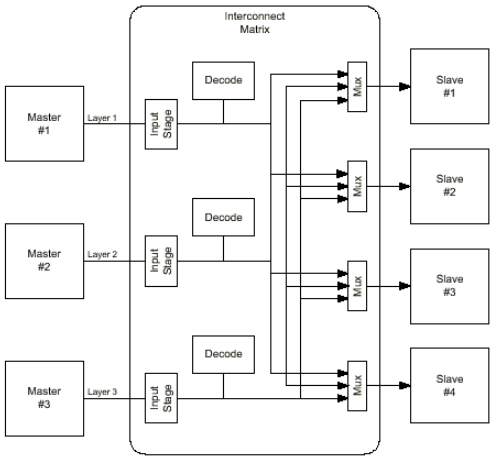

圖 5.4

在Multi-layer的文件中介紹了各種不同的bus configuration，例如同一個layer中可能有許多low-bandwidth的master，或是將AHB分成不同的sub-system，sub-system彼此不能互相存取，只有一個slave可以讓master透過connect matrix去彼此share等。下圖5.5則介紹其中的一種configuration：Local Slave。

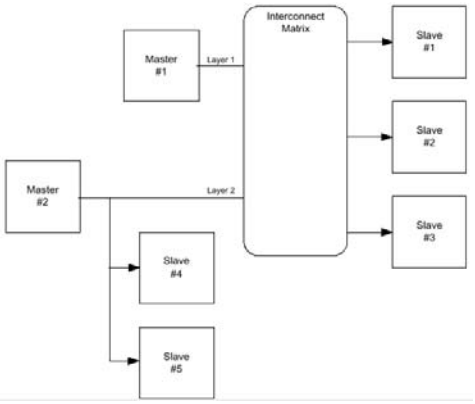

圖 5.5

6. 結論

在這篇文章中，我們首先介紹了基本的AHB與APB協定，由於AHB複雜的性質，因此我們也探討了一些AHB重要的規範或特性，另外文章最後也對ARM最新推出的AHB變形：AHB-Lite與Multi-layer AHB做了簡單的介紹。

ISSN 1609-8633