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 協定包含了四大部分: 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

AMBA 協定的目地是爲了要推出 on-chip bus 的規範，一開始 AMBA 1.0 只有 ASB 與 APB，爲了節省面積，所以這時候的 bus 協定都是 tristate 的 bus，而到後來 2.0 的 AHB 爲了能更方便設計者(trisate bus 要花更多精力去注意 timing)，因此改用 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 來講較爲簡單，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：

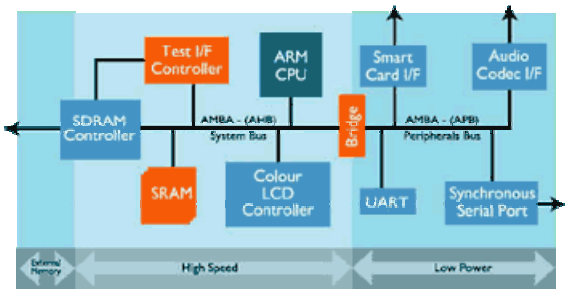

圖 2.1

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 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，是因爲它支援 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(爲了不使用 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。

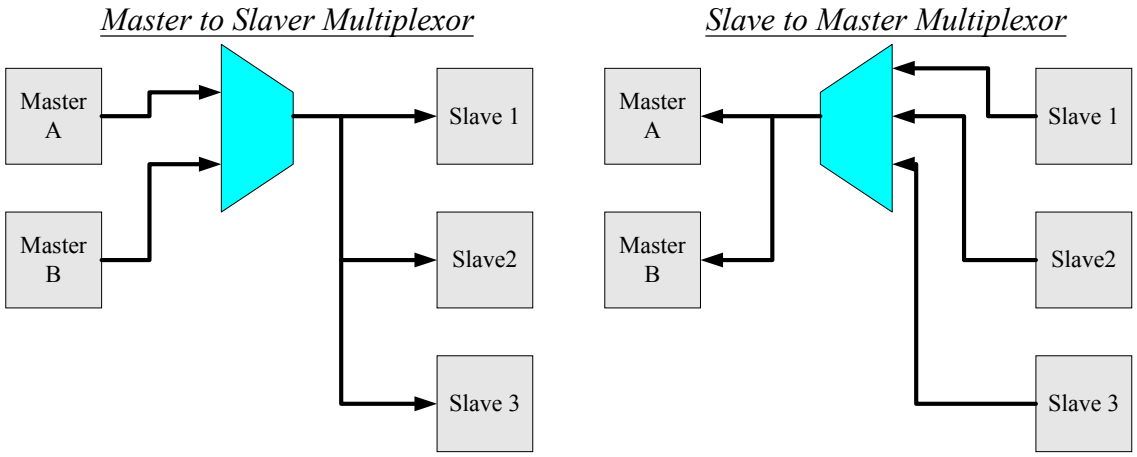

圖 3.1

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。
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，以作爲 selection signal。

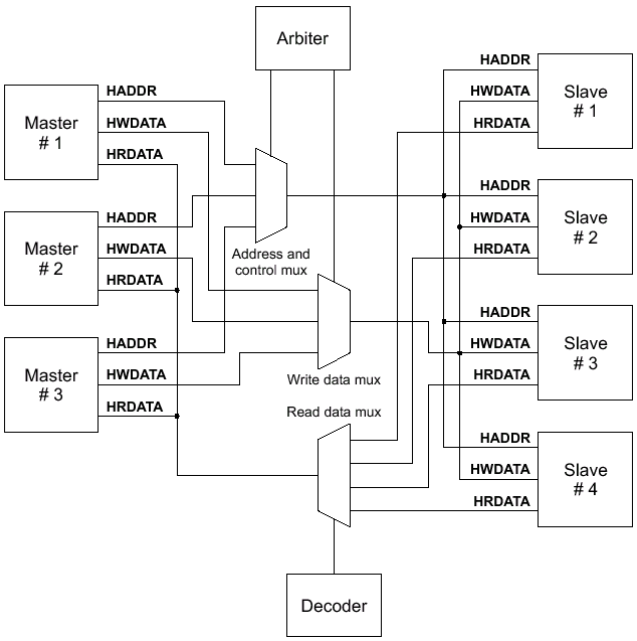

圖 3.2

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。

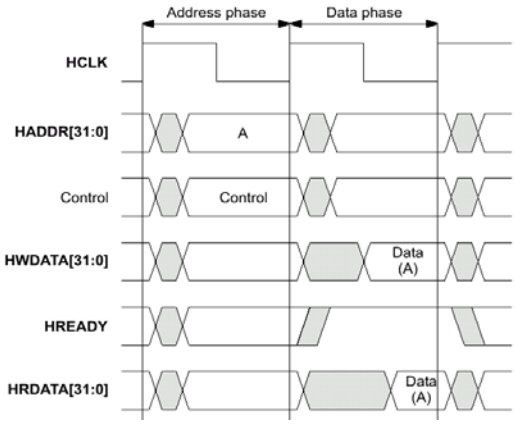

圖 3.3

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 爲 LOW 時，表示 transfer 尚未結束，爲 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)。

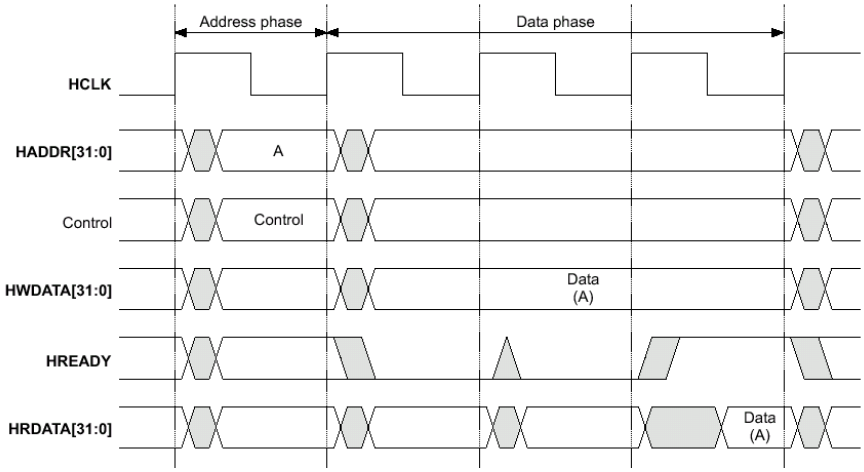

圖 3.4

由於一次 transfer 需要兩個 phase 才能完成，爲了增進 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。

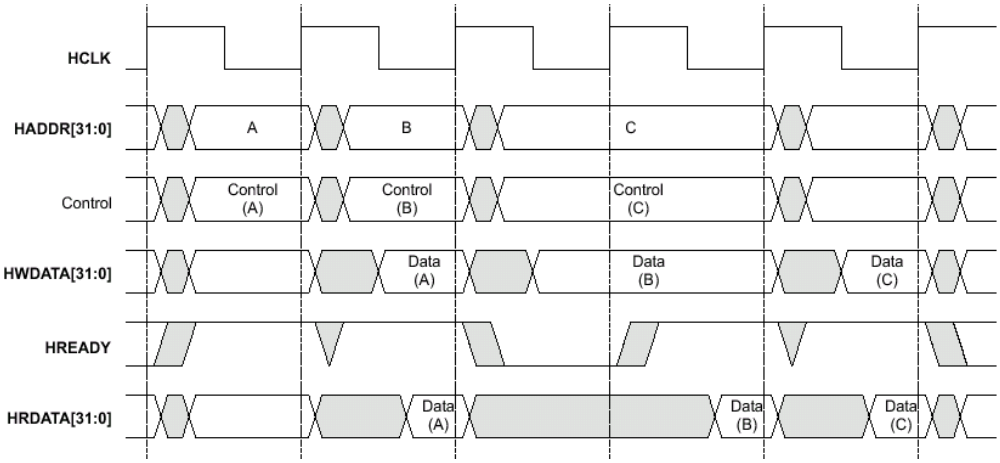

圖 3.5

3.3 Control signal

AHB 上的 Control signal 共有五類，分別爲

- 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 的 address 和 control 訊號與上一筆 transfer 無關。
- **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 爲 transfer type 的 example。從時序圖裡我們可以看出：第一筆 burst transfer 的 type 一定爲 NONSEQ，另外因爲 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。

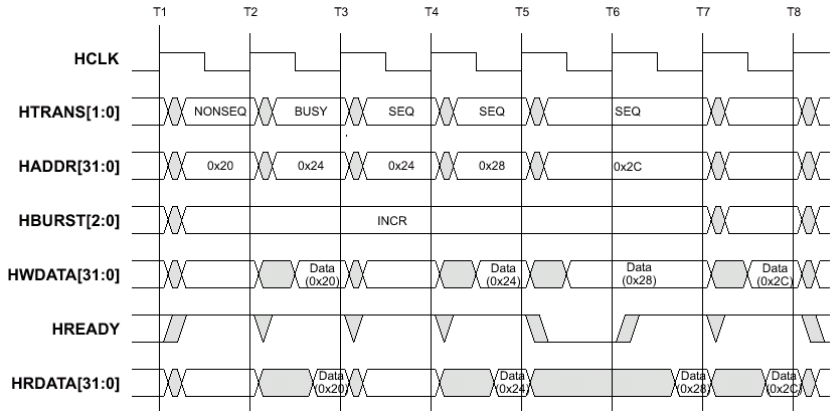

圖 3.6

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 爲 word (4 byte)，第一筆 transfer 的 address 爲 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。

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 爲 word，所以 address 爲在 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。

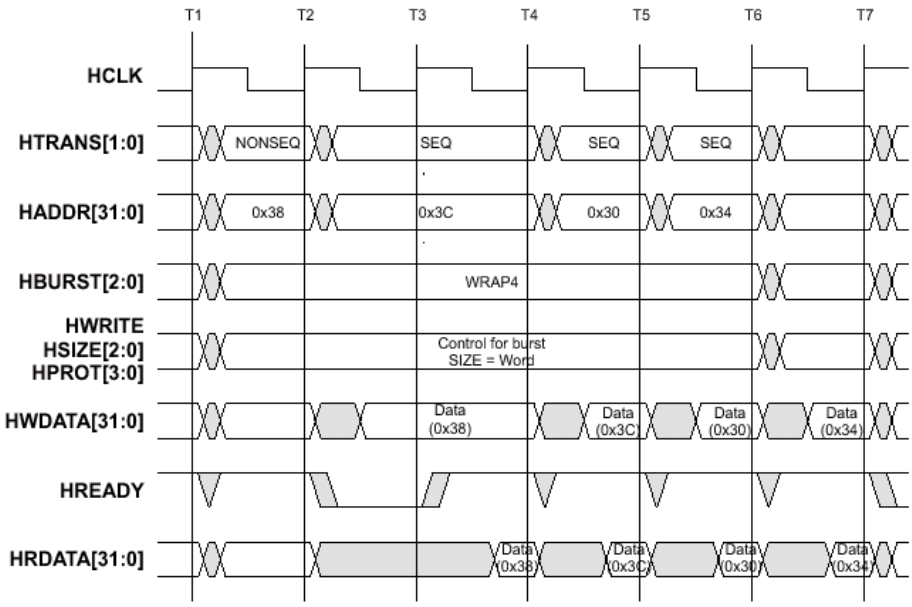

圖 3.7

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

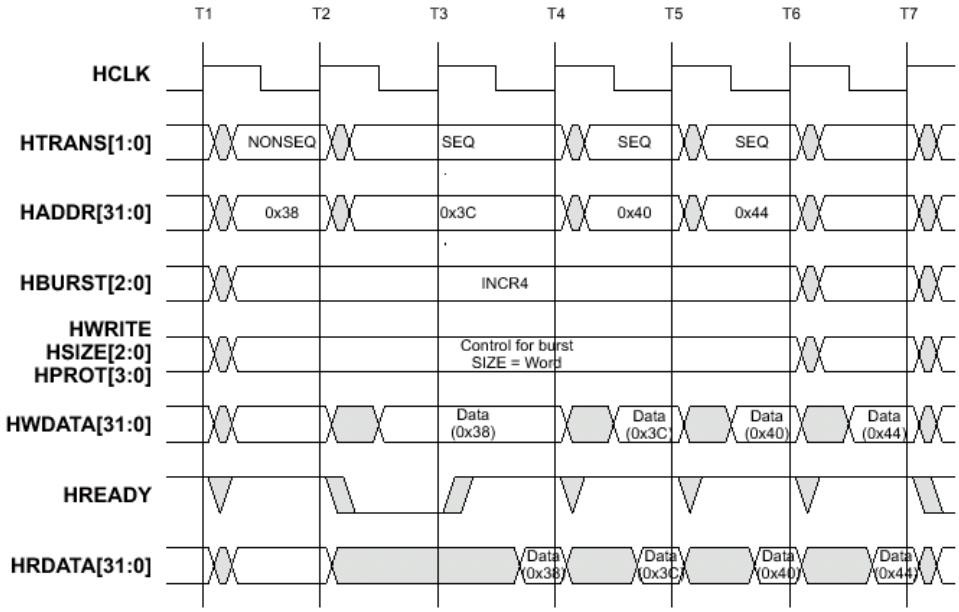

圖 3.8

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

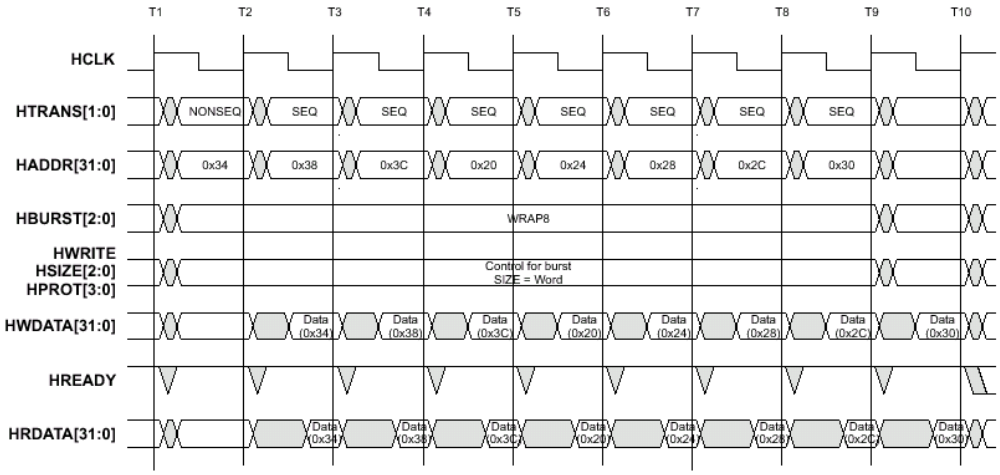

圖 3.9

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

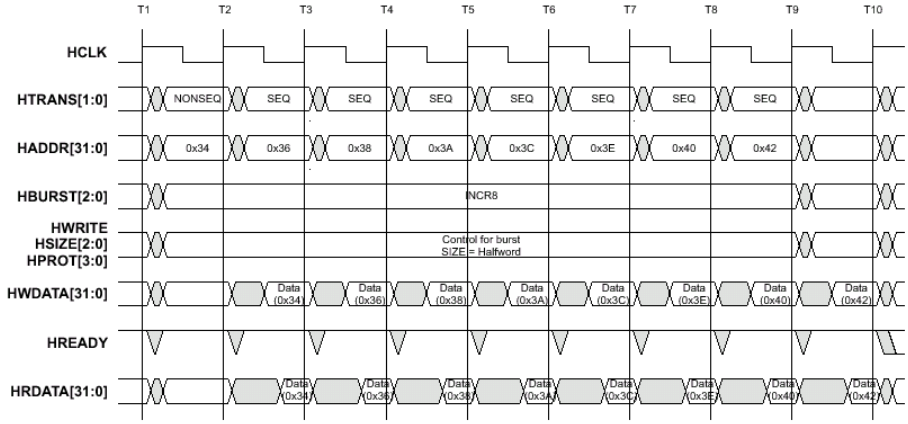

圖 3.10

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。

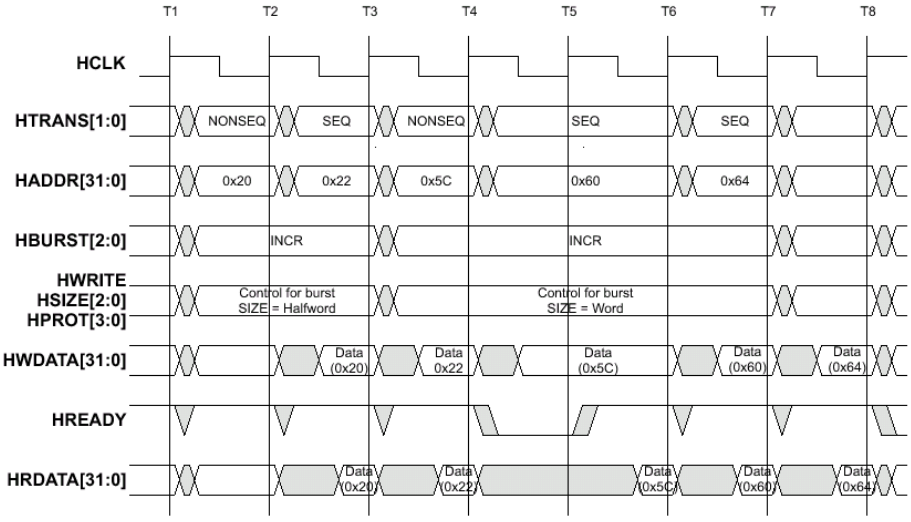

圖 3.11

3.3.3 Transfer Direction

當 HWRITE 爲 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 爲 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 爲 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

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 爲 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，爲了避免因爲目前的 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 爲 low，第二個 cycle 爲 HIGH。

之所以需要兩個 cycle，這是因爲 AHB 爲 pipeline operation，current transfer 的 data phase 與 next transfer 的 address phase 是 overlap 的。而這三種 Response 皆可能因爲目前的 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，因爲 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：

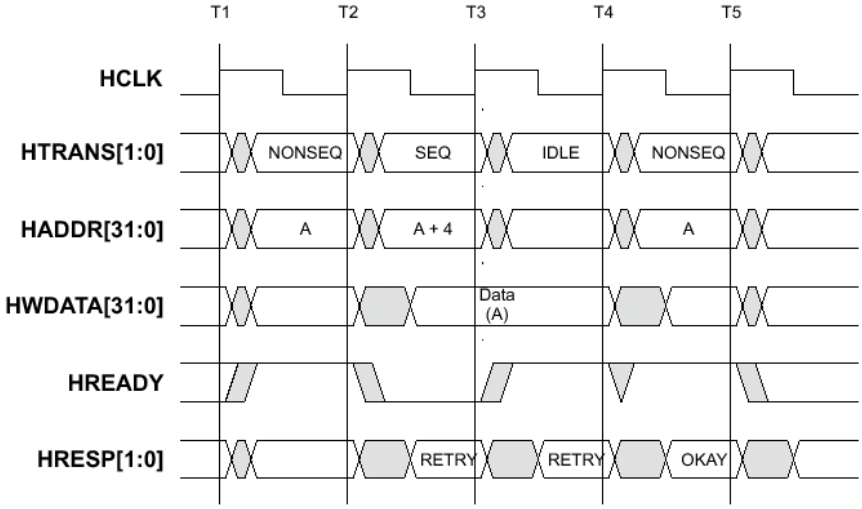

圖 3.12

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 爲 LOW，不僅將 data phase 給 extend，同時也表示 address A+4 的 transfer 並沒有開始，所以 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(因爲 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，這其實是爲了讓 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 爲 OKAY。

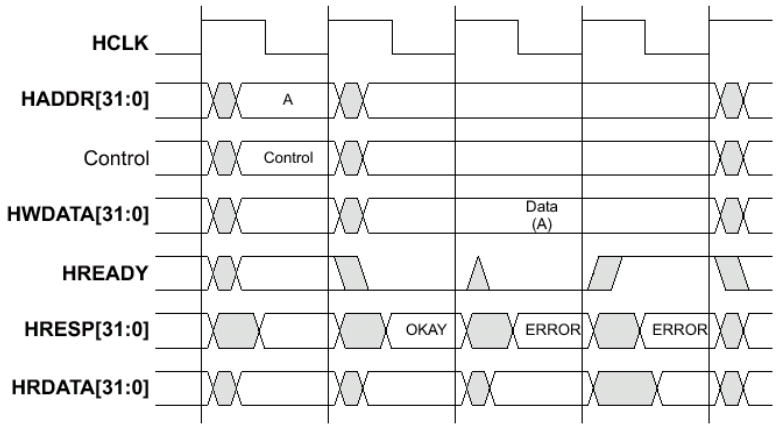

圖 3.13

3.5 Arbitration

由於 AHB 爲 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 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)，另外因爲 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。

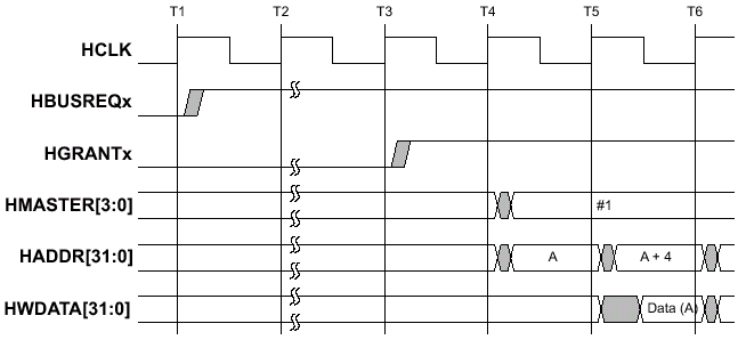

圖 3.14

當 master 的 HGRANT 被 assert 後，且 HREADY 爲 High，則代表 master 在下個 cycle 就可以 access bus 了。若 HREADY 的訊號一直爲 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 顯示出因爲 HREADY 訊號而延遲了 GRANT 的時間。

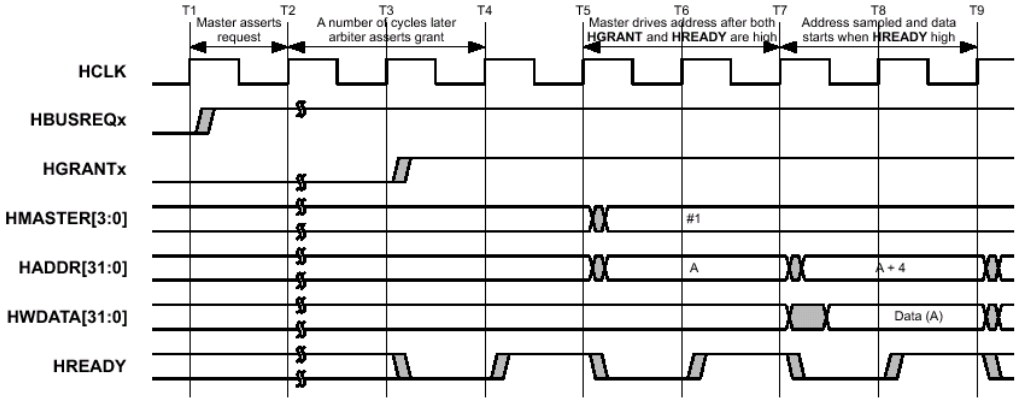

圖 3.15

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。

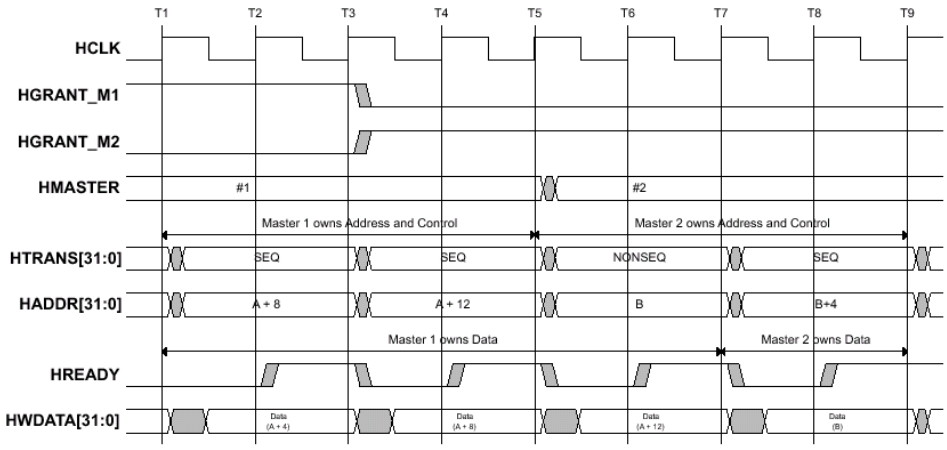

圖 3.16

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。

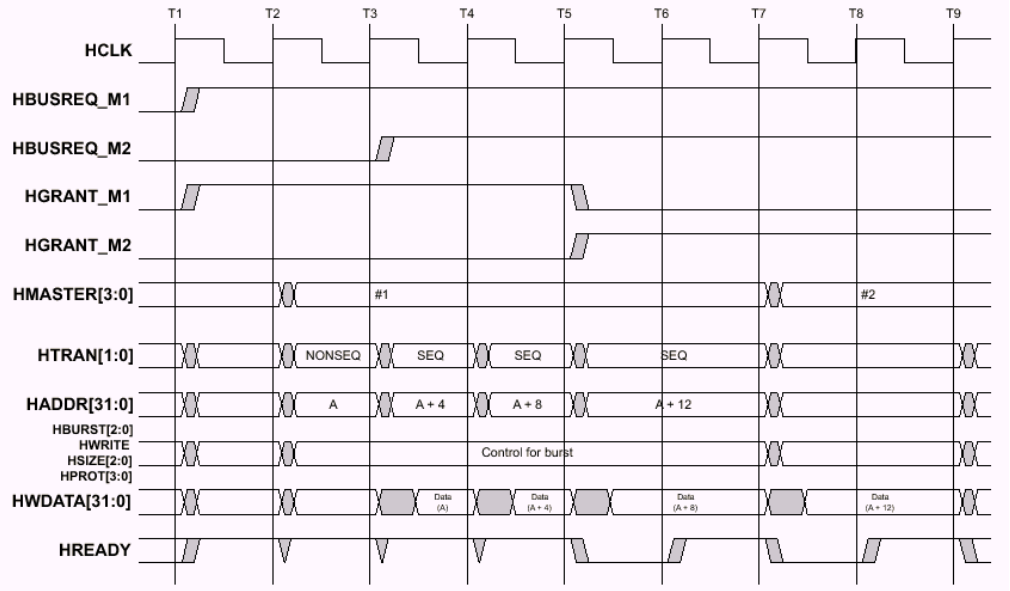

圖 3.17

在前一節介紹 RETRY/SPLIT response 時，我們曾經介紹其爲 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 HGRANT 訊號如何利用 IDLE transfer 的 cycle 去變化。

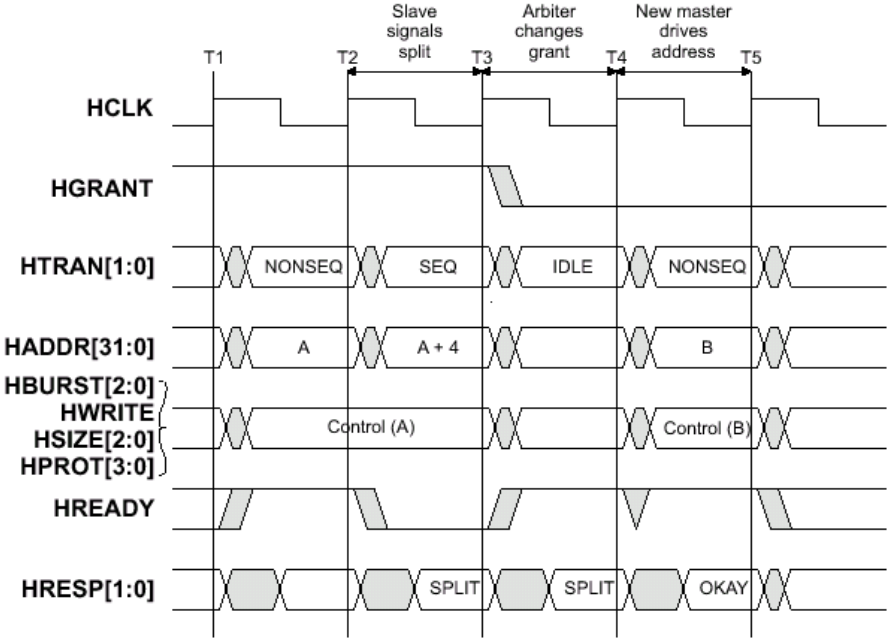

圖 3.18

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 爲 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，底下爲 APB 的特性：

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

4.1 APB Overview

我們先列出 APB 的訊號名稱與功用，下表 4.1 爲所有的 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：

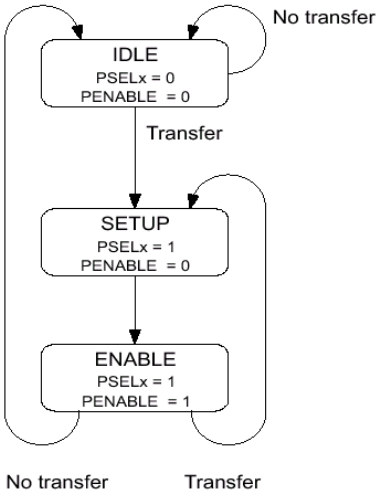

圖 4.1

- 一開始爲 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。在圖 4.2 裡，時間 T2->T4 爲一個 APB 的 write transfer(記住 APB transfer 是 always two-cycle)，第一個 cycle : T2->T3 爲 SETUP Cycle，第二個 cycle:T3->T4 爲 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。

另外爲了節省 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。

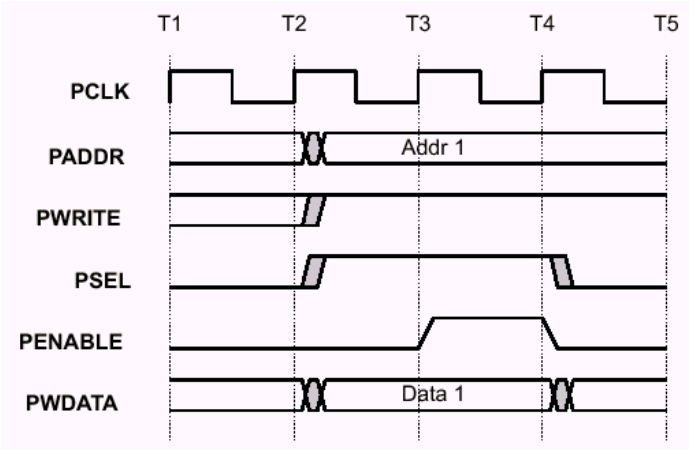

圖 4.2

4.3 Read Transfer

圖 4.3 爲 read transfer 的時序圖，除了 PWRITE 爲 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。

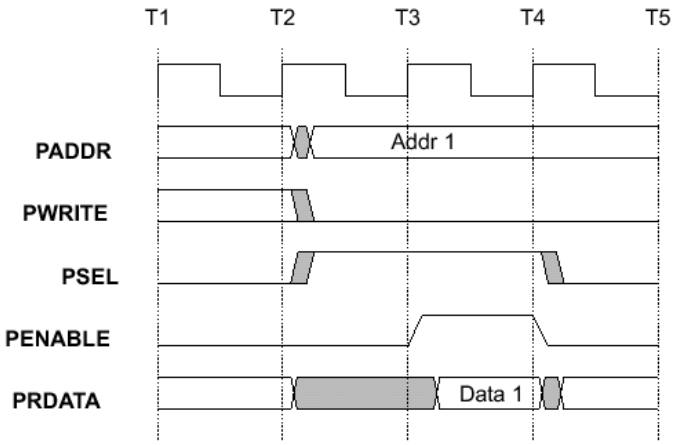

圖 4.3

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

對於 read data，slave 則可以在 PWRITE 爲 LOW 而 PSEL, PENABLE 皆爲 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。

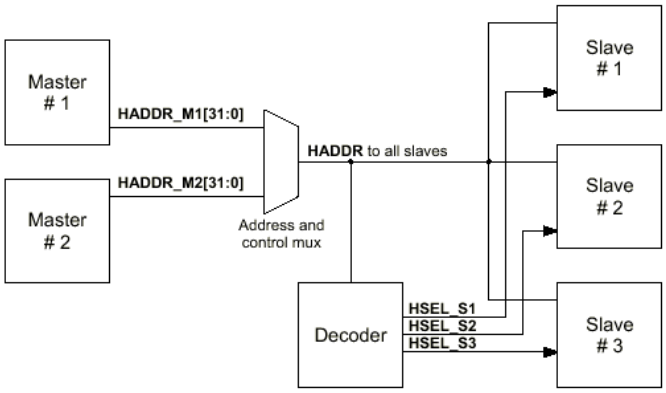

圖 5.1

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 爲 Low)，因此 AHB Slave 在設計時要特別注意只有在 HSELx 與 HREADY 皆爲 High 的情形下去 sample address/control 訊號。

由於 AHB 的 address line 有 32 條，爲了簡化 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。

爲了避免 AHB master 在進行 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 爲系統中的正常 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

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

爲了避免 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.的建議值爲 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 就顯的不必要了，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(因爲不可能有其他 master 去 access bus 了)。下圖 5.2 爲 AHB-Lit 系統的 block diagram。

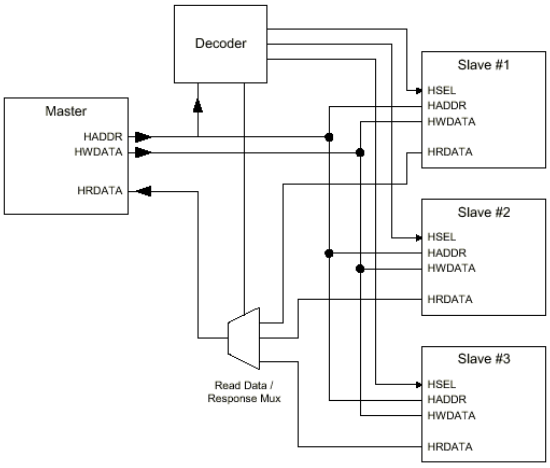

圖 5.2

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，爲了解決這個問題，ARM 推出了 Multi-layer AHB，其基本構想如下圖 5.3。

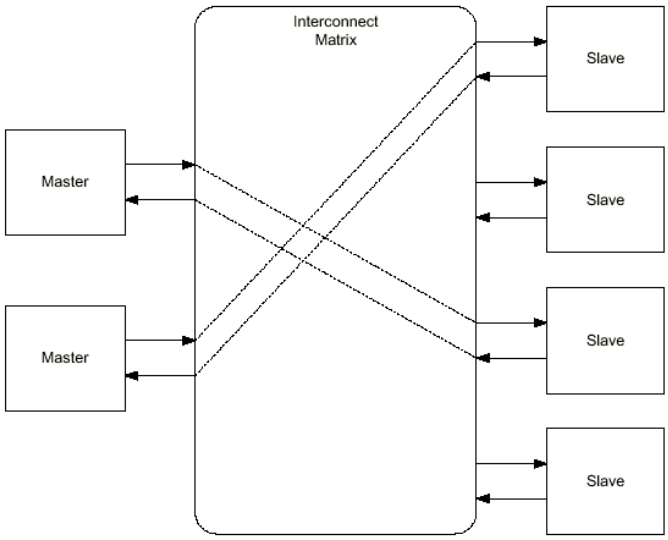

圖 5.3

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。

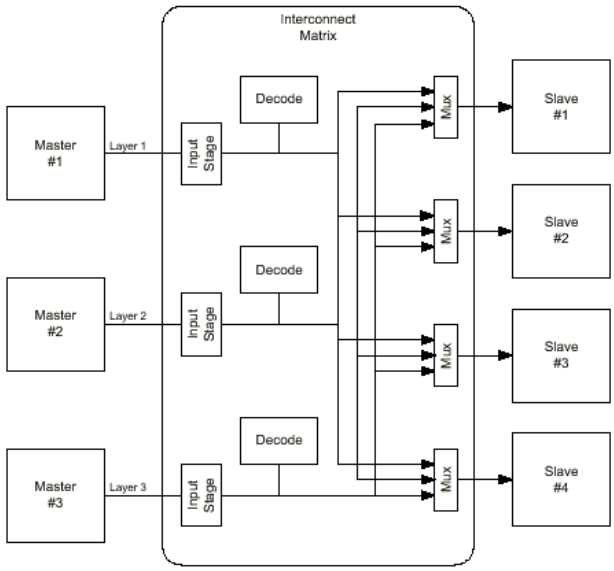

圖 5.4

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。

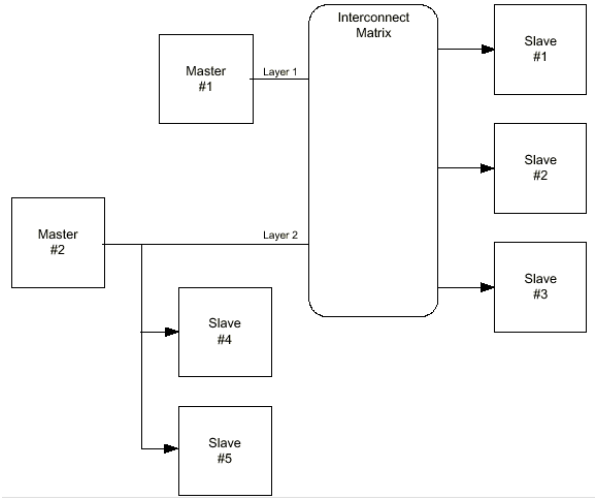

圖 5.5

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。