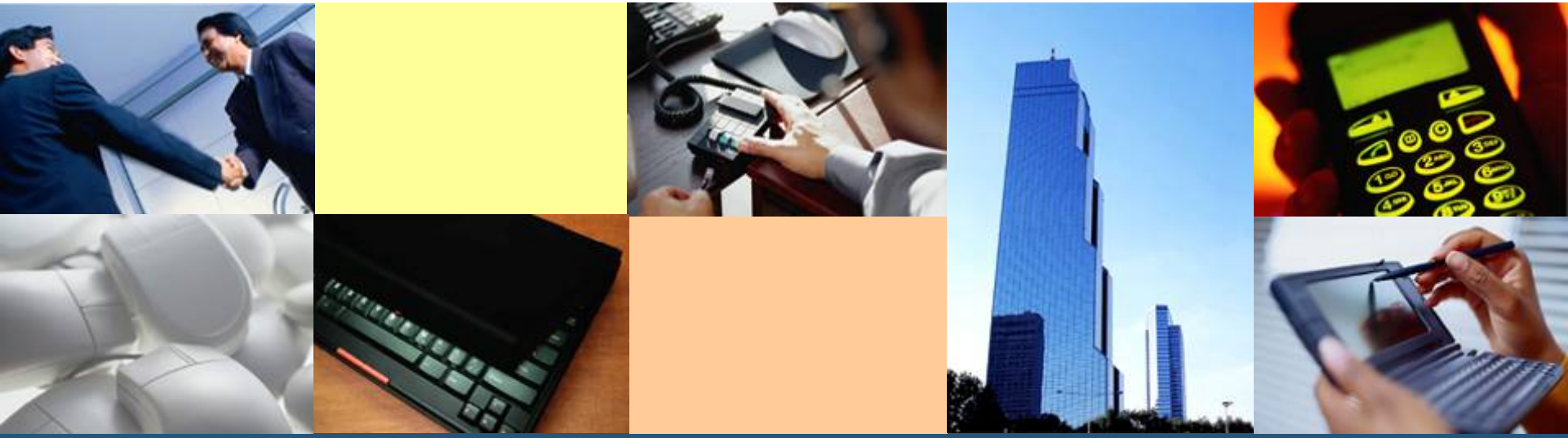
ANDES TECHNOLOGY

國立交通大學 National Chiao Tung University

Introduction of AMBA Bus System

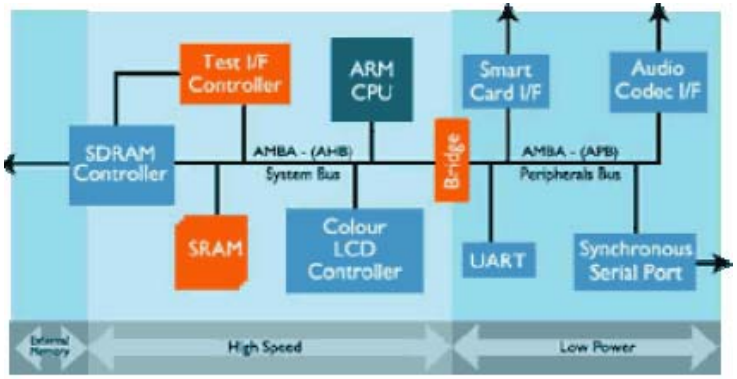


AMBA

- Advanced Micro-controller Bus Architecture
- AMBA 協定目前是open 且free
- AMBA 協定包含
  - AHB
  - APB
  - ASB
  - Test Methodology

AMBA

- 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Andes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。



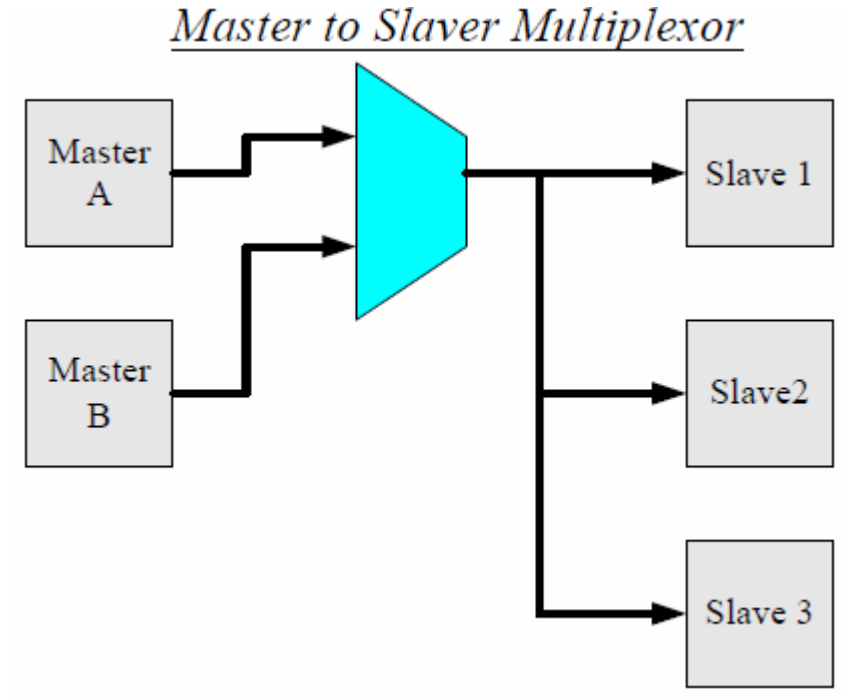
AMBA特性

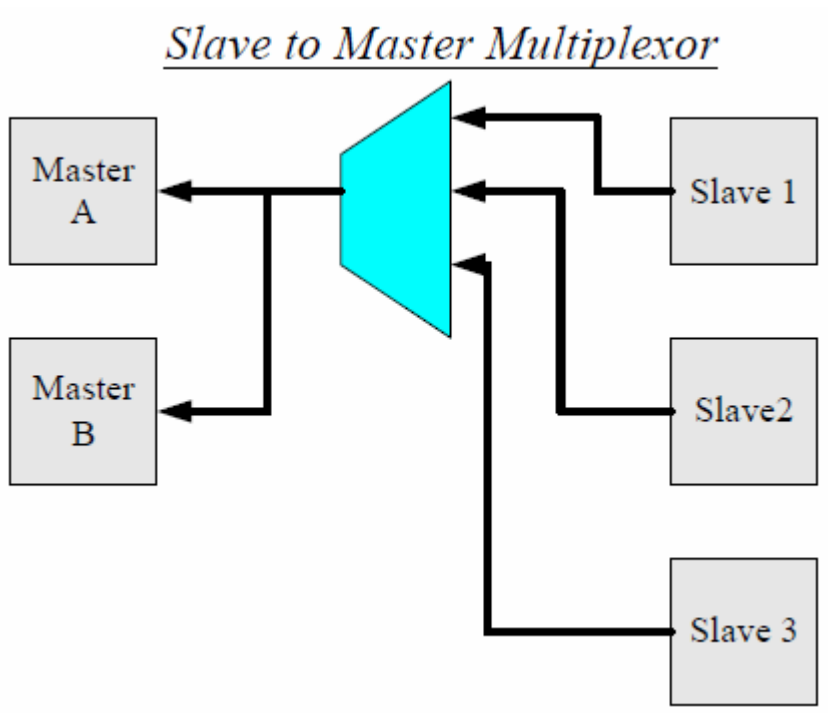
- single-clock edge operation
- non-tristate implementation
- burst transfers
- split transaction
- multiple bus master

AHB system

- 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 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

AHB system


Master to Slaver Multiplexor
Master A
Master B
Slave 1
Slave2
Slave 3


Slave to Master Multiplexor
Master A
Master B
Slave 1
Slave2
Slave 3

AHB system

- 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。

AHB system

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 ◦

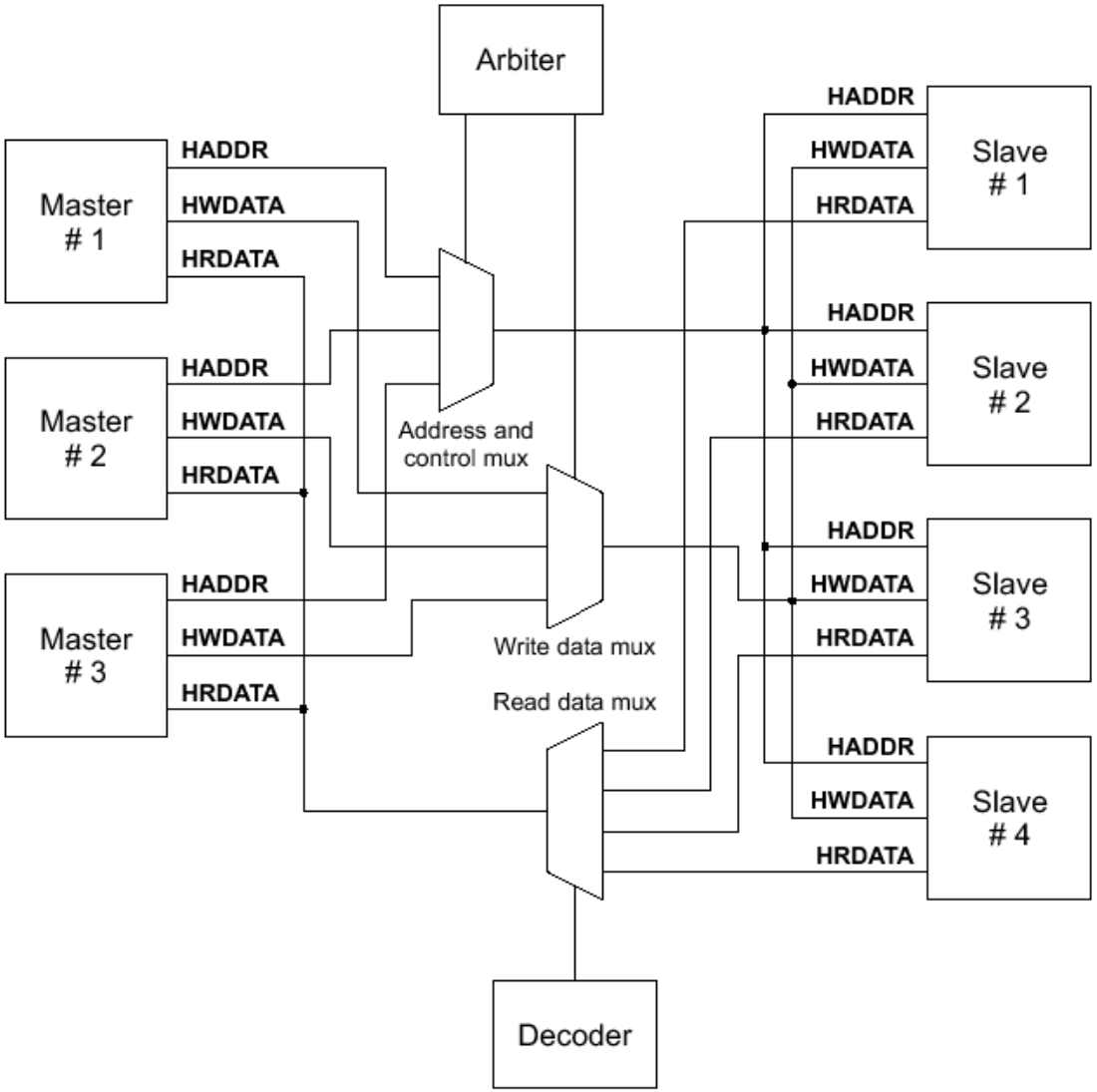
AHB system

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 ◦

AHB system

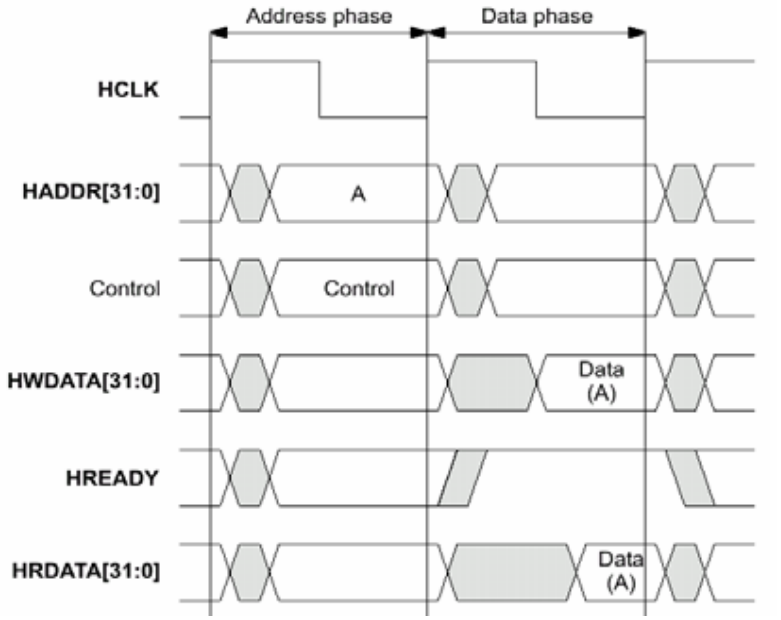
- AHB的bus interconnection
  - 各種control signal (HBURST, HTRANS 等)的連接，其實他們的連線與HADDR一致。
  - Master 與Arbiter 之間的Request/Grant 訊號。
  - Decoder 與各個Slave 間會有Selection 的訊號。
  - Control mux 的output 會有部分control 訊號除了接到Slave 外也會接到Arbiter (HTRANS/HBURST)。
  - Response signal (HREADY, HRESP)的mux。另外Arbiter 會輸出HMASTER 訊號，這個訊號會接master-to-slave multiplexor，以作爲selection signal。

AHB的bus interconnection



AHB system

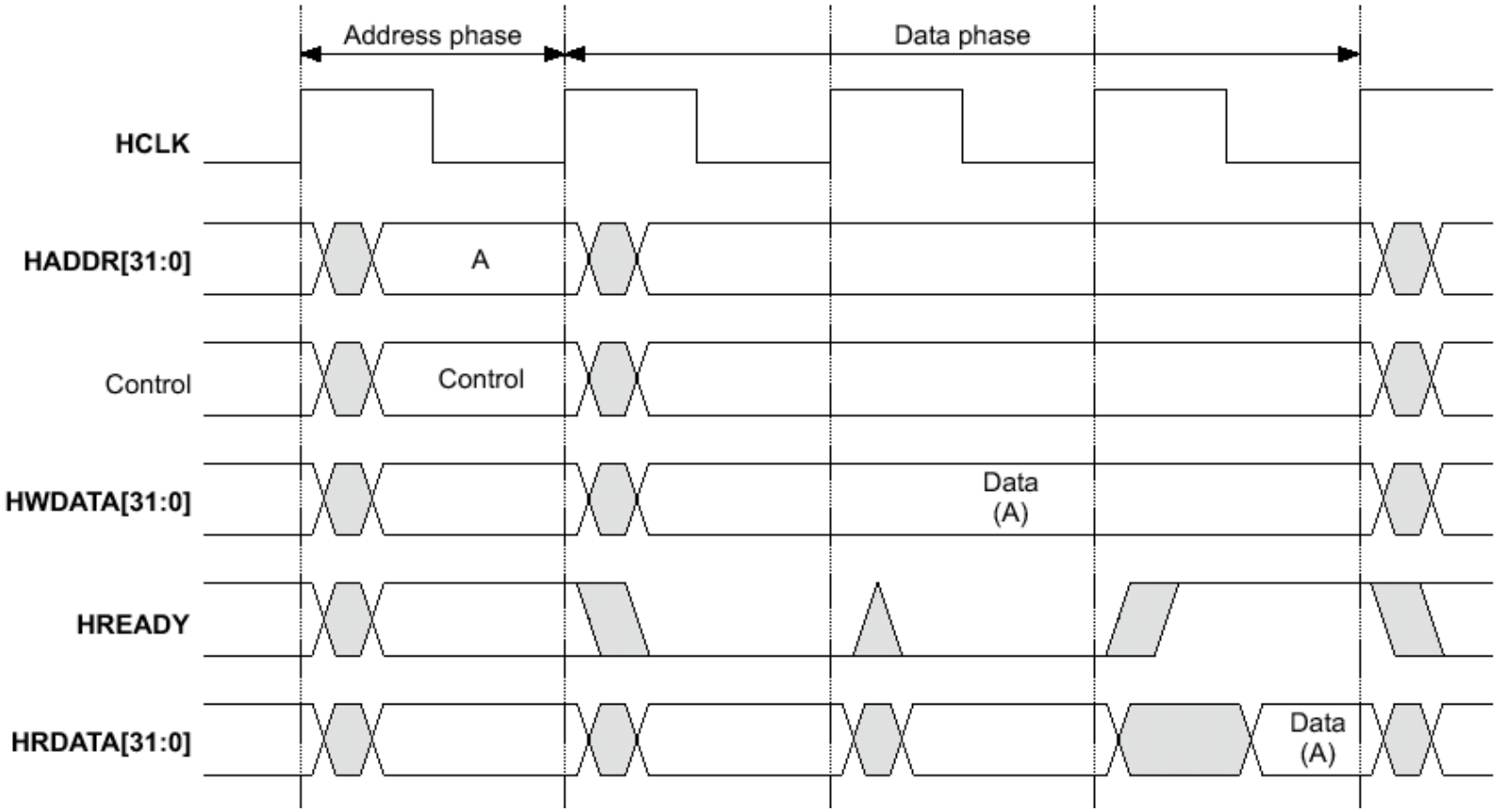
- Basic transfer
  - address phase
    - 傳送的是address 與control signal
  - data phase
    - 是write/read data 與response signal



Basic transfer

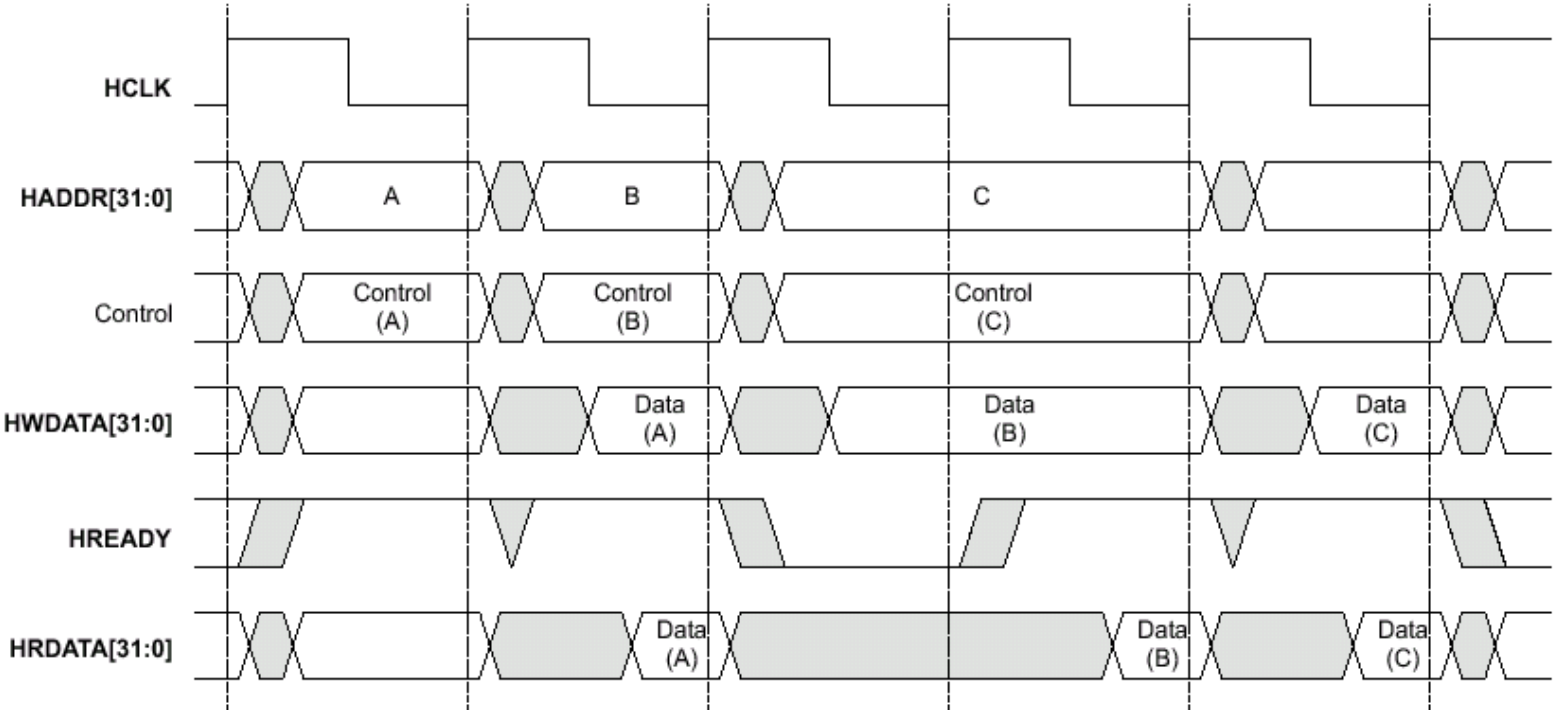
- transfer在data phase 時若無法在1個clock cycle內完成，slave可用HREADY訊號去延長(extend) transfer。當HREADY爲LOW時，表示transfer尚未結束，爲HIGH時，則代表目前的transfer結束了，但結束時的status 則需看Slave回應的HRESP訊號(可能是OKAY, ERROR 等)。

Basic transfer



Basic transfer

- 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 data phase是overlap 在一起的，



Basic transfer

- 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。

Control signal

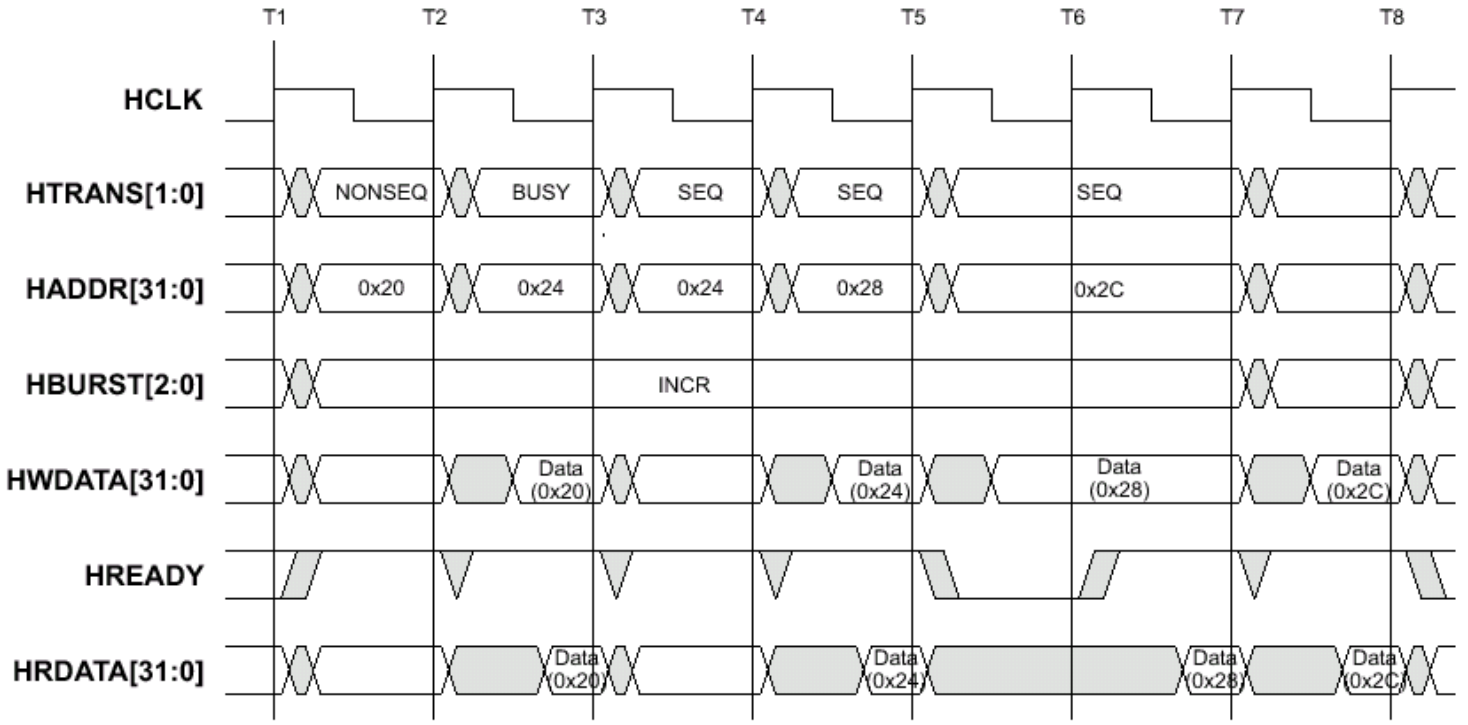
- HTRANS[1:0] ：Transfer Type
- HBURST[2:0] : Burst Type
- HPROT[3:0] : Protection Control
- HSIZE[2:0] : Transfer Size
- HWRITE :Transfer Direction

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。

Transfer Type 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。



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。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。

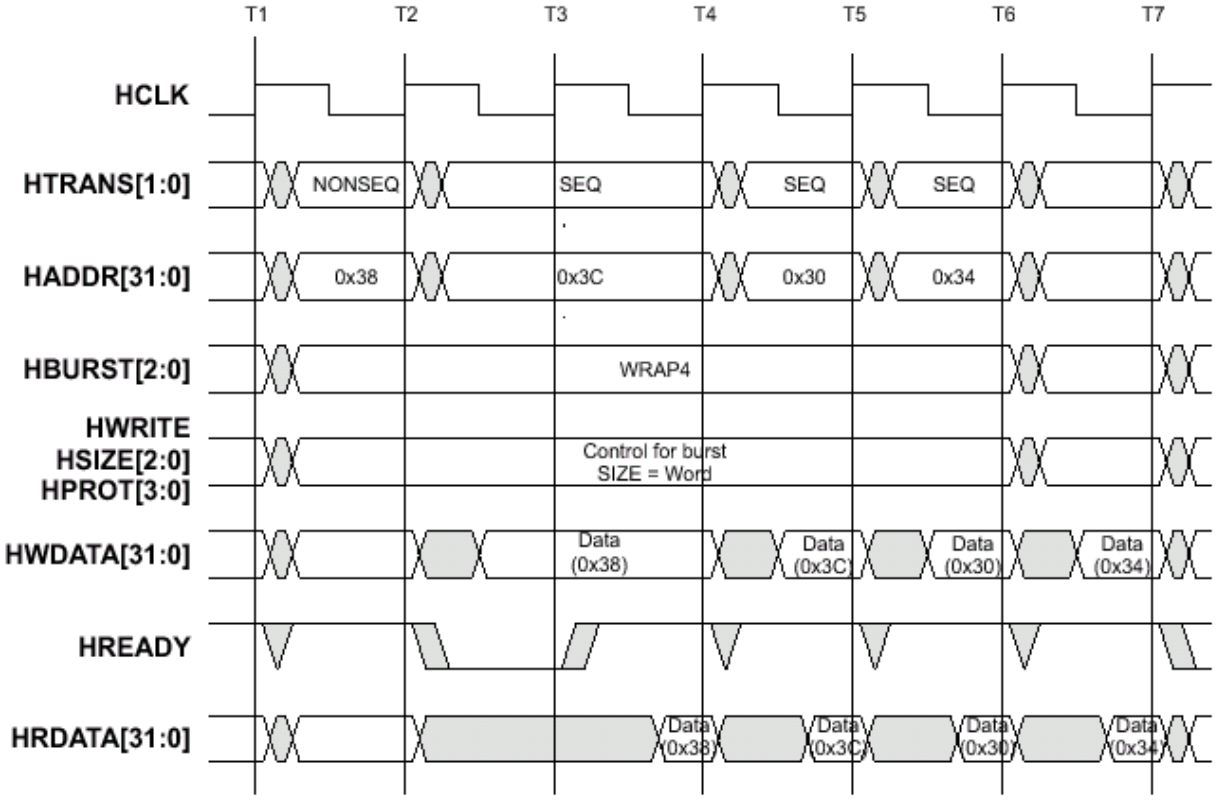
Burst Type

- 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

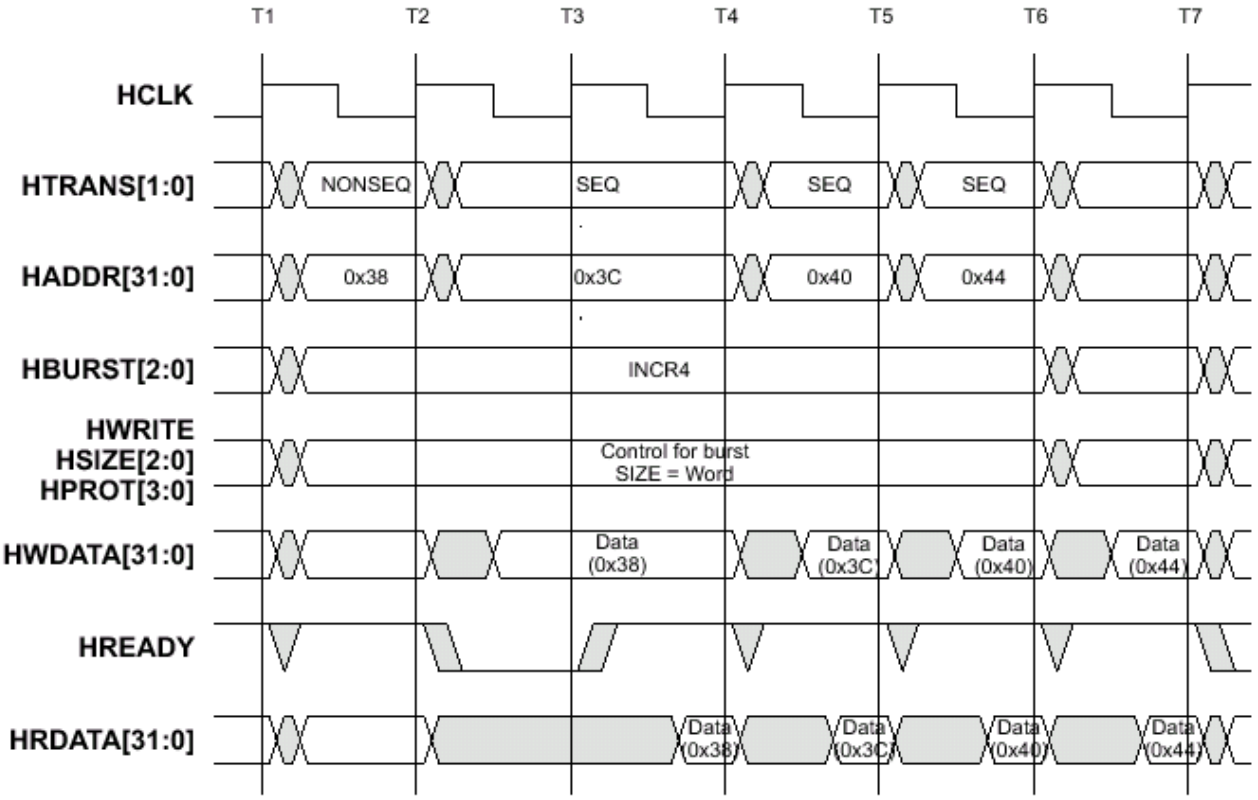
Burst Type

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



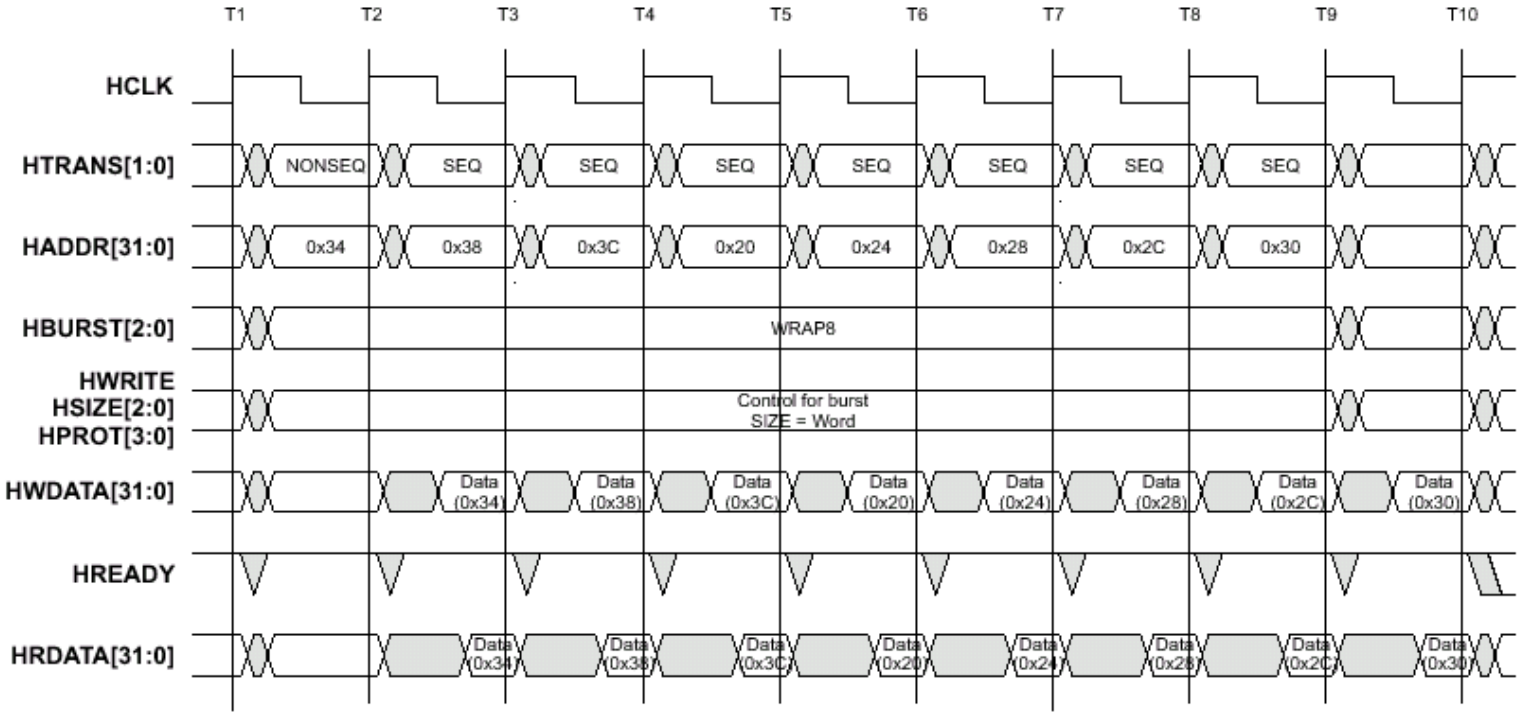
Burst Type

- 在圖中，因爲是INCR type，所以address 0x3C之後會跨過16-byte boundary，直接到0x40。



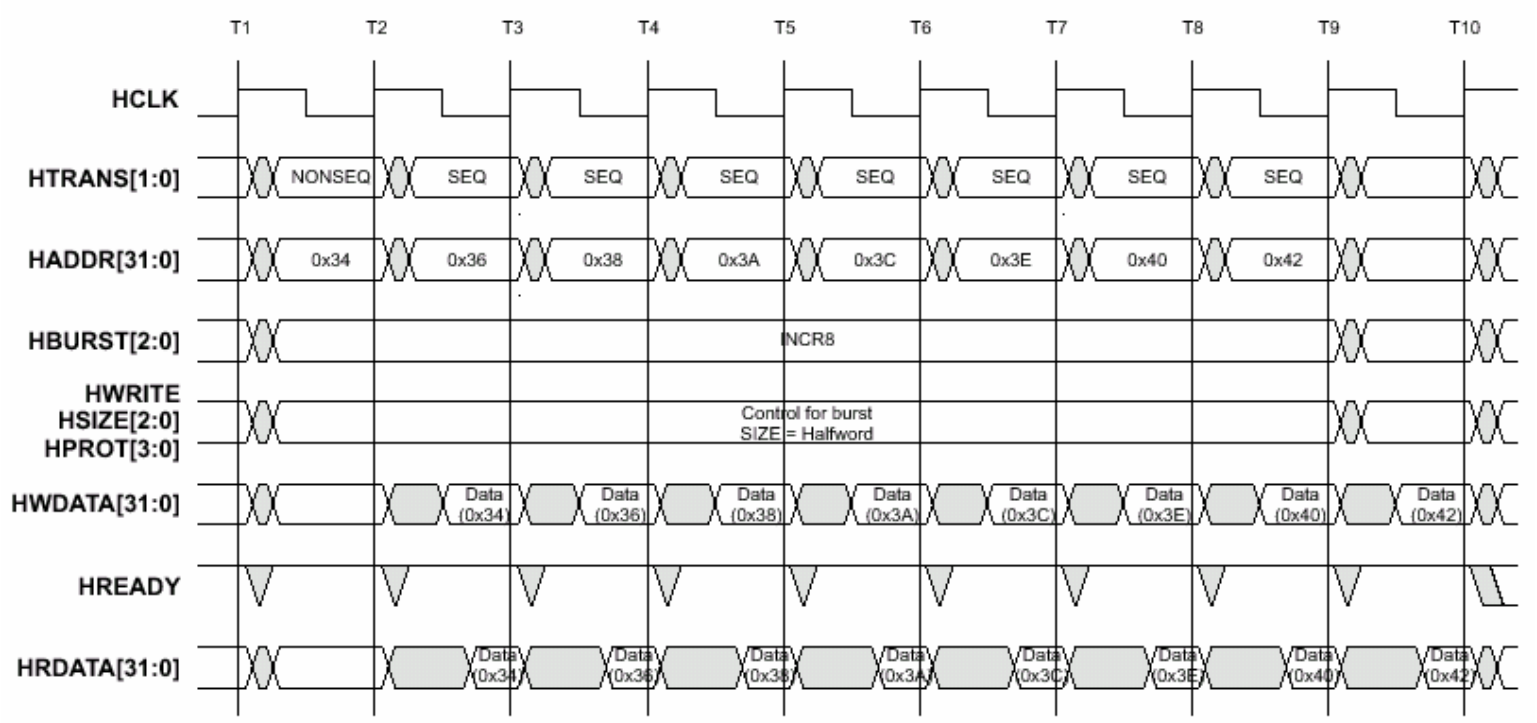
Burst Type

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



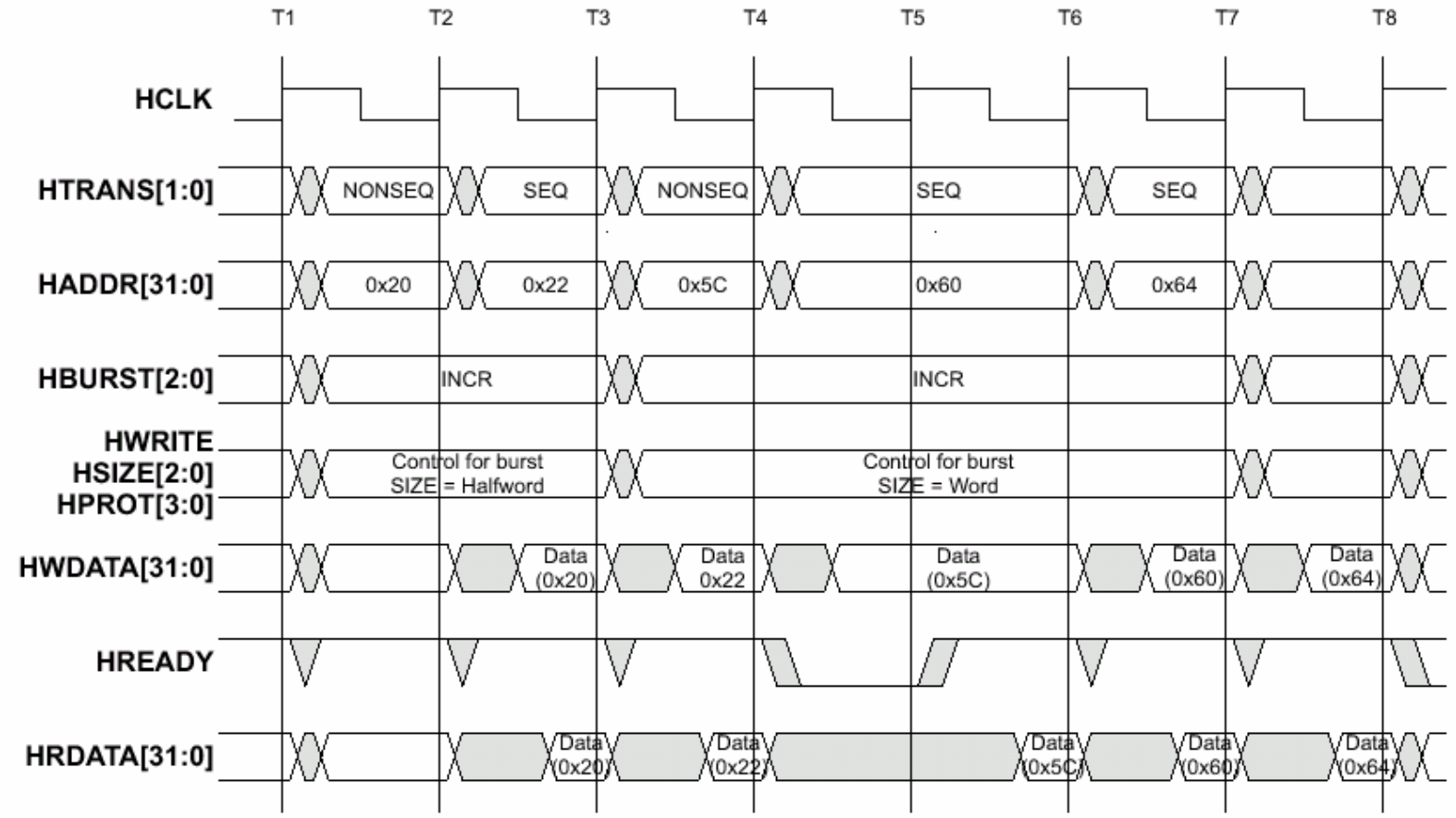
Burst Type

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



Burst Type

- 圖中介紹兩筆undefined length burst的例子，而transfer size一筆是Halfword，另一筆是word。



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。

Transfer Size

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	-

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。

Slave Response

- 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)

Slave Response

- 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。

Arbitration

- 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 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)

Arbitration

- 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。

APB

- 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

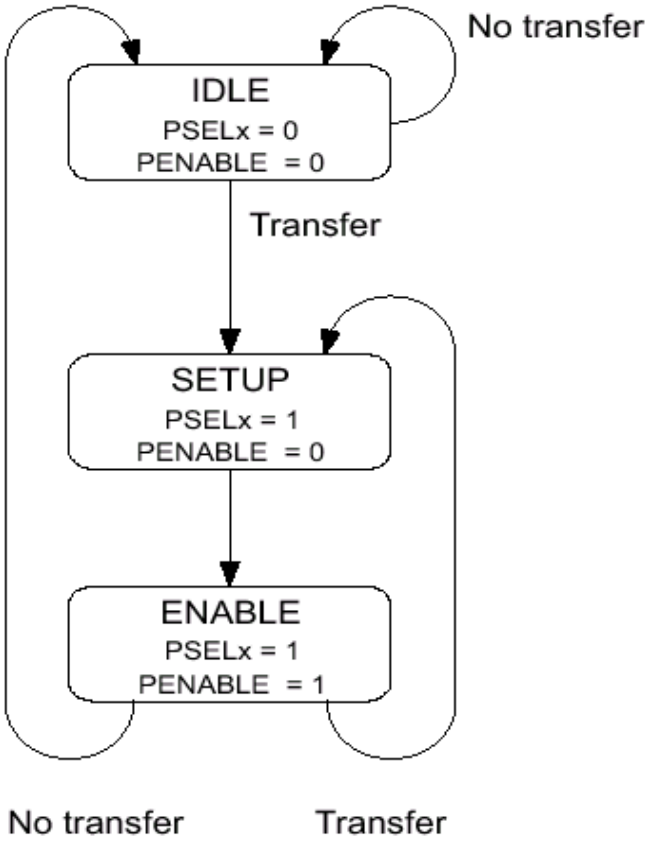
APB

❖APB 的訊號名稱與功用

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

- APB上的transfer可以用下面的state diagram來說明



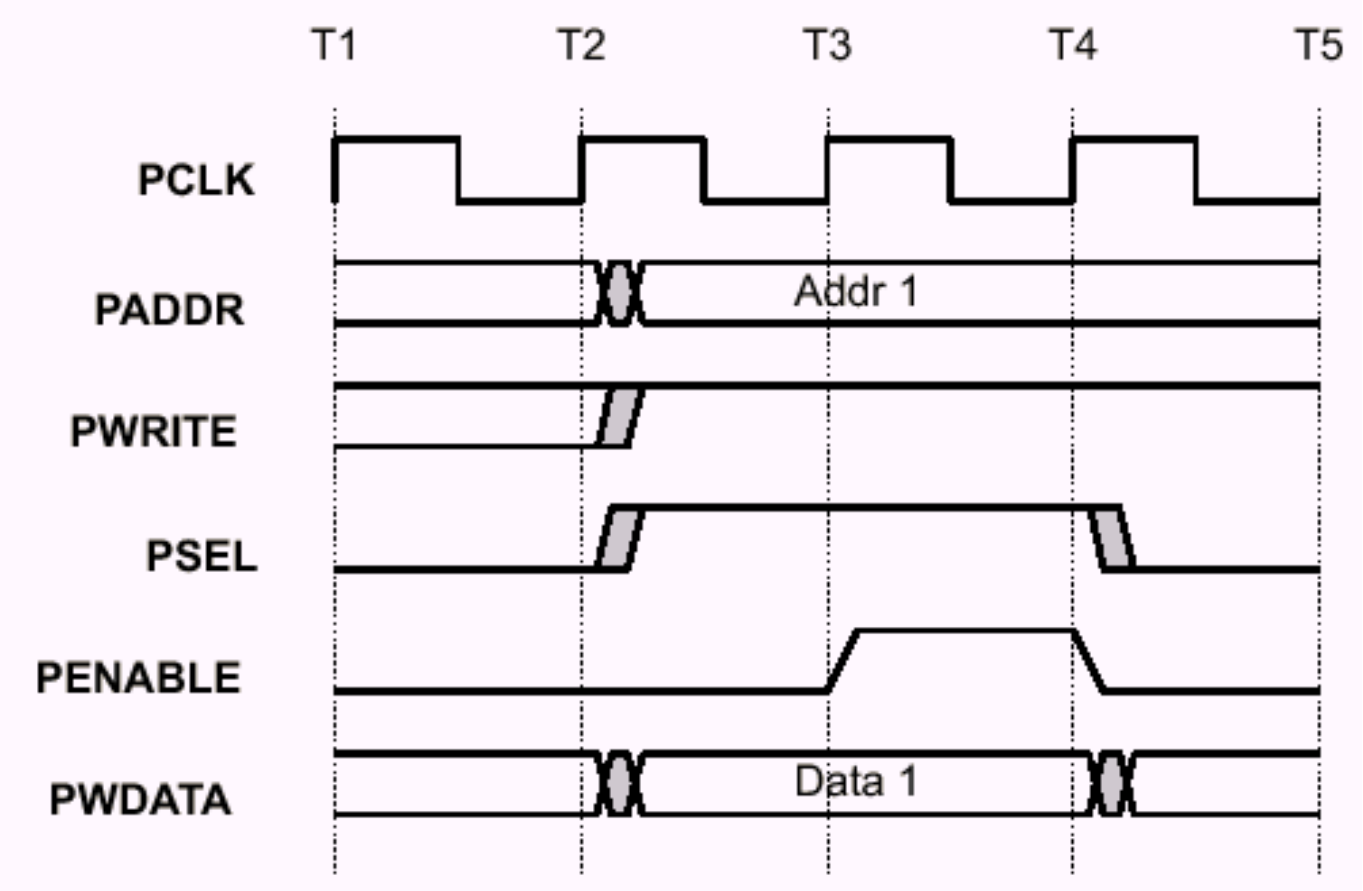
APB

- 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。

Write Transfer

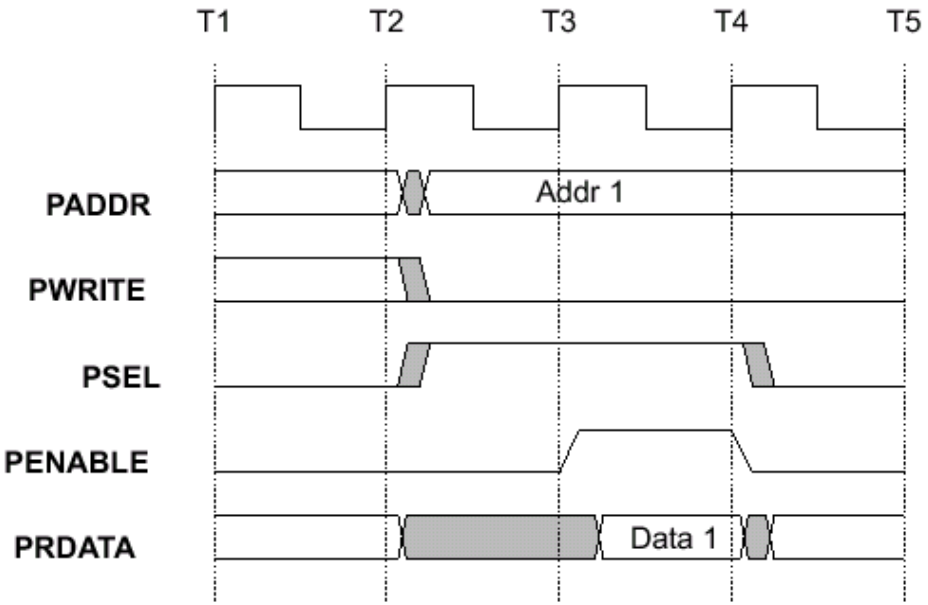
- 基本的write transfer。時間T2->T4 爲一個APB的write transfer(記住APB transfer 是always two-cycle)，第一個cycle : T2->T3 爲SETUP Cycle，第二個cycle:T3->T4 爲ENABLE cycle。在整個transfer裡，address/control/data訊號都需維持不變。在transfer結束後，PENABLE訊號一定會deasserted (go LOW)，而PSELx訊號則視情況而定，若有transfer要接著對同一個slave進行，則維持不變(HIGH)，反之，則會被dirve Low。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。

Write Transfer



Read Transfer

- Read transfer ，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。



APB Slave 與APB bridge

- 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。

Thank You