Introduction to AMBA AHB

2013 – 2017

Ando Ki
(adki@future-ds.com)

AMBA buses

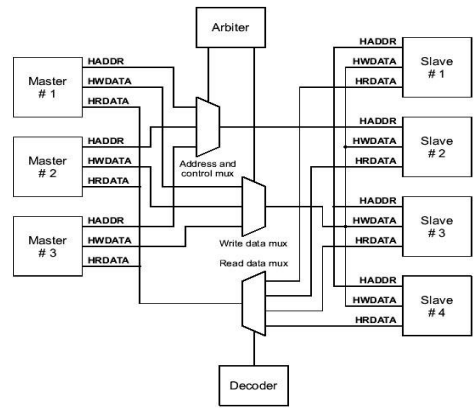
AMBA 4 AXI	AMBA 3 AXI	AMBA 2 AHB
an extension of AMBA 3 AXI Burst length up to 256 Quality-of-service Removal of lock transaction Removal of write interleaving	channel architecture registers slices one address for burst up to 16 multiple outstanding bursts out of order completion data interleaving low-power interface	burst transfers pipelined operation split transactions single-cycle bus master handover single-clock edge operation multiple bus masters (up to 16) two uni-directional 32-bit data bus for read and write

AMBA 2 naming convention

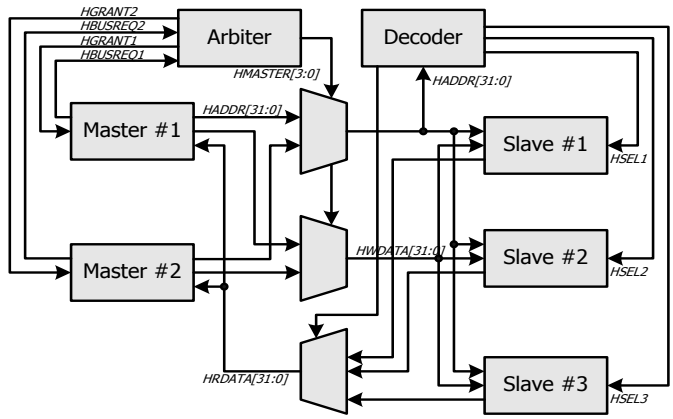
- All AMBA bus signals are named such that the first letter of the name indicates which bus the signal is associated with.
  - H: AHB signal, e.g., HCLK.
  - P: APB signal, e.g., PCLK.
  - B: ASB signal, e.g., BCLK.
  - A: unidirectional signal between ASB bus master to the arbiter, e.g., AGNTx, AREQx.
  - D: unidirectional ASB decoder signal, e.g., DSELx.
- A lower case 'n' at the end of the signal name indicates that the signal is active low. Otherwise all signal names are always upper case.
  - E.g., HRESETn, PRESETn.
- Test signals have a prefix 'T' regardless of the bus type.

AMBA AHB components

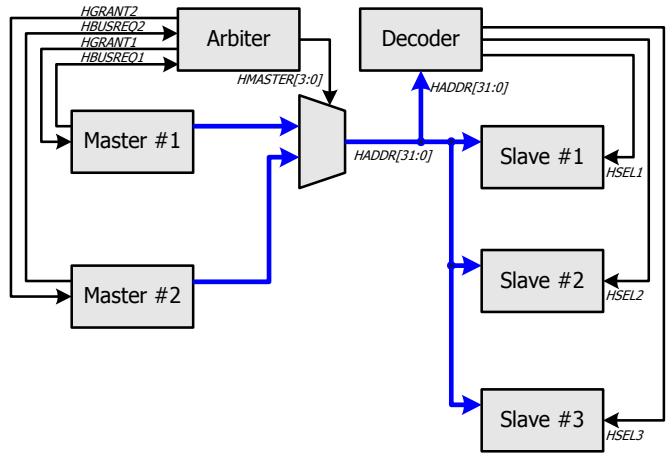
- Master
  - Initiate read and write operations by providing an address and control information.
  - E.g., processor, DMA, DSP, ahb2ahb bridge
- Slave
  - Responds to a read or write operation within a given address space range.
  - E.g., memory, ahb2apb bridge
- Arbiter
  - Ensures that only one bus master at a time is allowed to initiate data transfers.
- Decoder
  - Decode the address of each transfer and provide a select signal for the slave that is involved in the transfer.
- Etc.: Multiplexer, AHB2APB bridge, AHB2AHB bridge



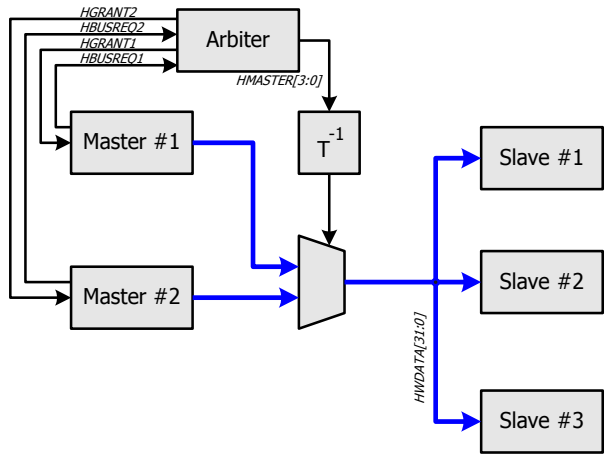
AMBA component connection



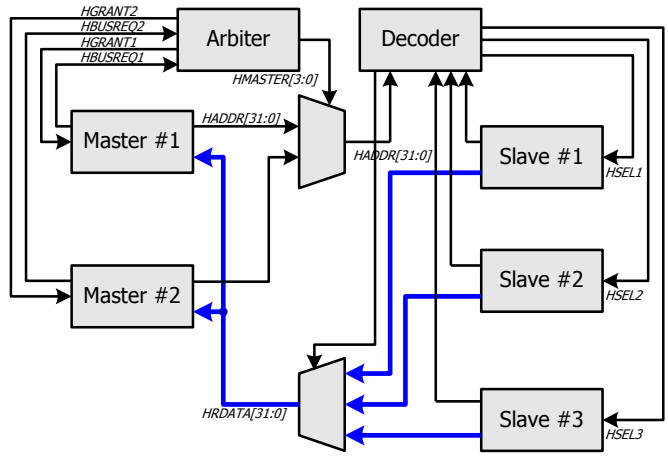
AMBA address path



AMBA data path (write)



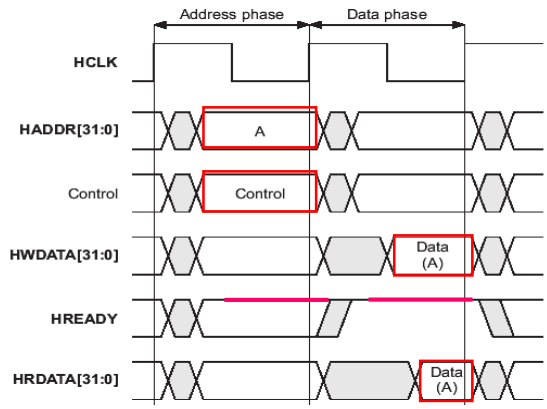
AMBA data path (read)



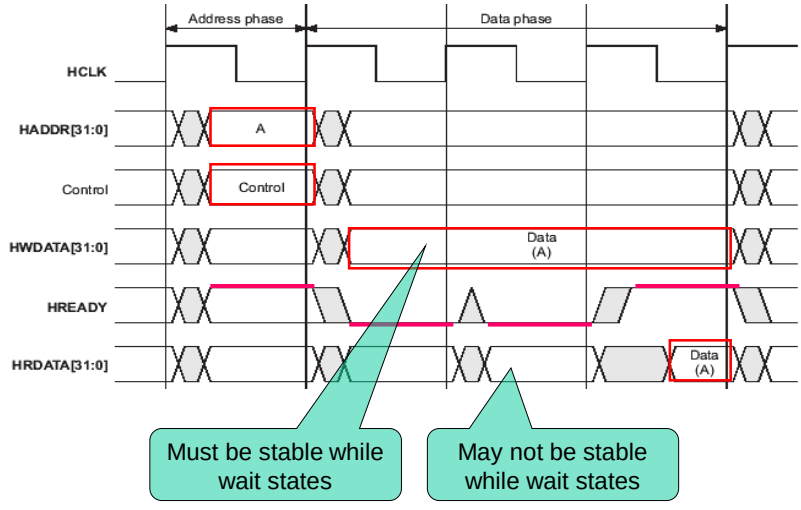
Data transfers and arbitration

- Basic data transfer with zero wait
- Basic data transfer with wait states
- Multiple transfers with overlapping
- AHB master, slave, decoder
- AHB2APB bridge
- APB slave
- AHB arbiter and signals
- AHB arbitration without wait
- AHB arbitration with wait
- AHB arbitration: bus ownership
- AHB arbitration: handover
- Handover after split transfer
- Handover after retry response
- Handover after error response

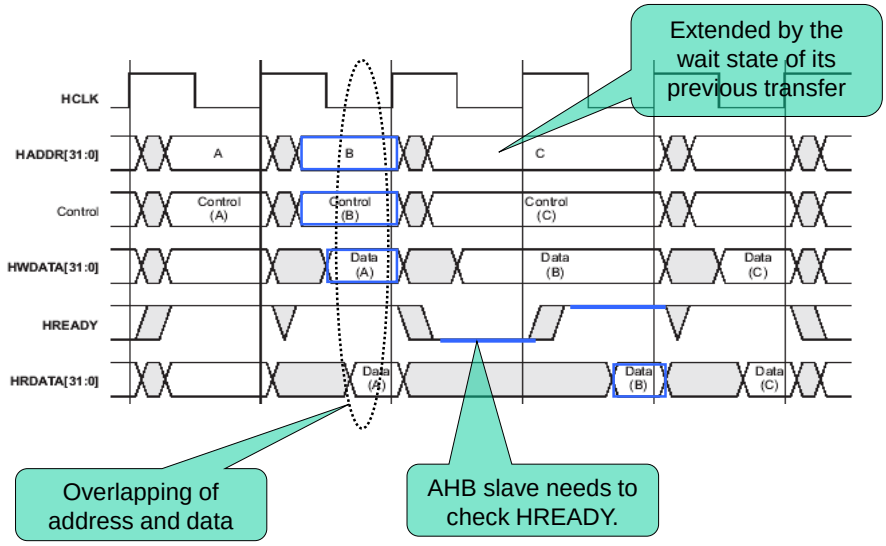
Basic data transfer with zero wait



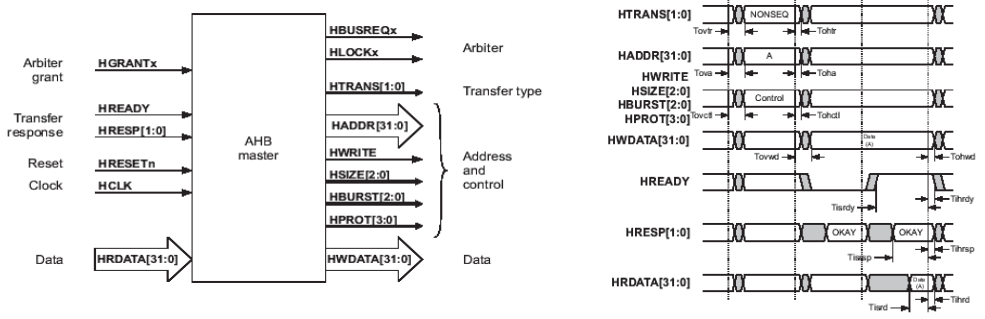
Basic data transfer with wait states



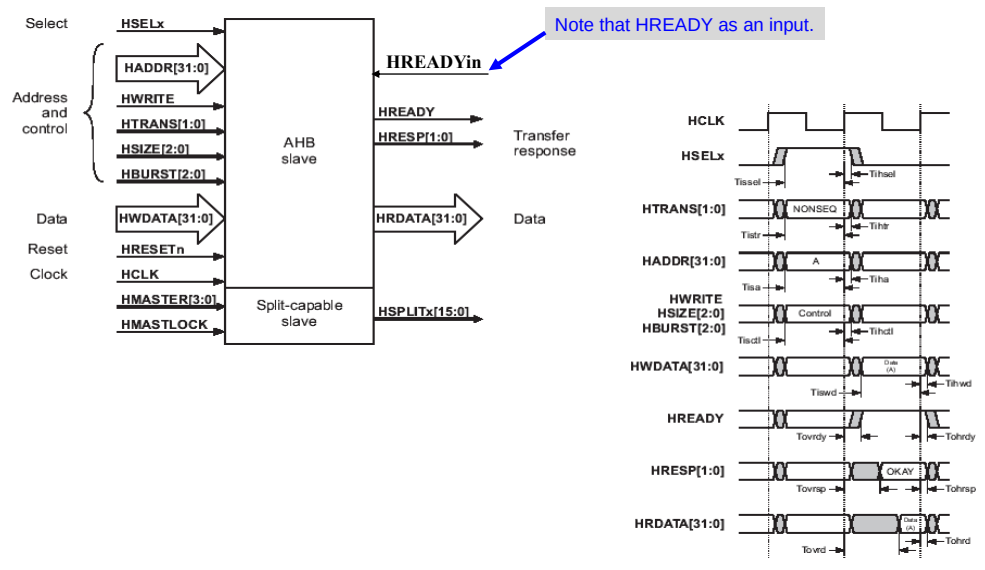
Multiple transfers with overlapping



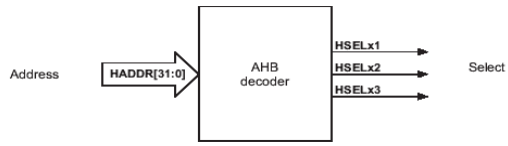
AHB master

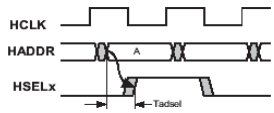


AHB slave

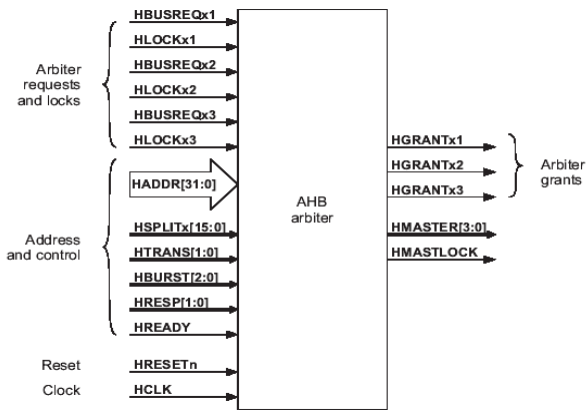


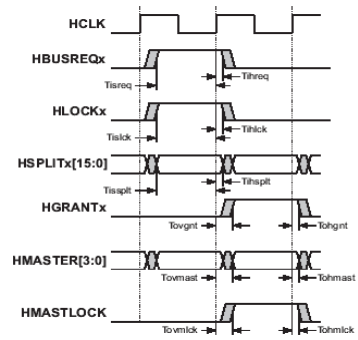
AHB decoder





AHB arbiter

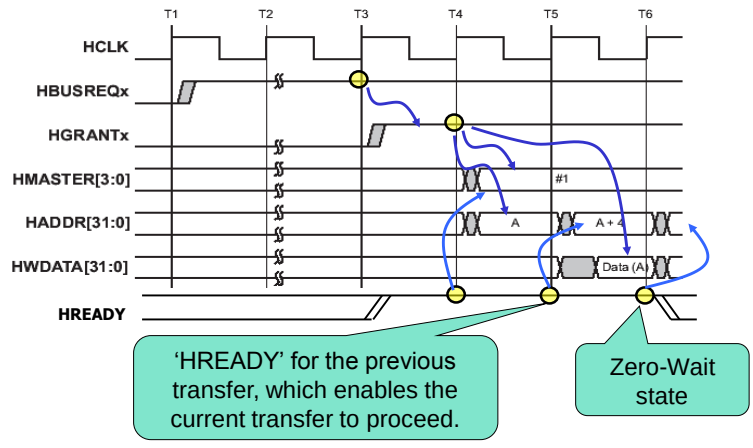




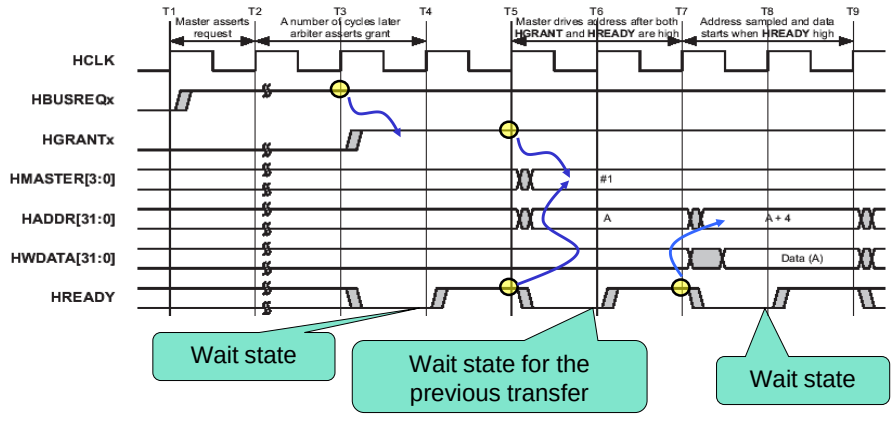
AHB arbiter signals

signal	from	to	remarks
HBUSREQ[15:0]	master	arbiter	
HGRANT[15:0]	arbiter	master	
HLOCK[15:0]	master	arbiter	
HMASTER[3:0]	arbiter	Mux, slave	
HMASTERLOCK	arbiter		
HSPLIT[15:0]	slave	arbiter	
HTRANS[1:0]	master	arbiter	
HBURST[2:0]	master	arbiter	
HRESP[1:0]	slave	arbiter	
HREADY	slave	arbiter	

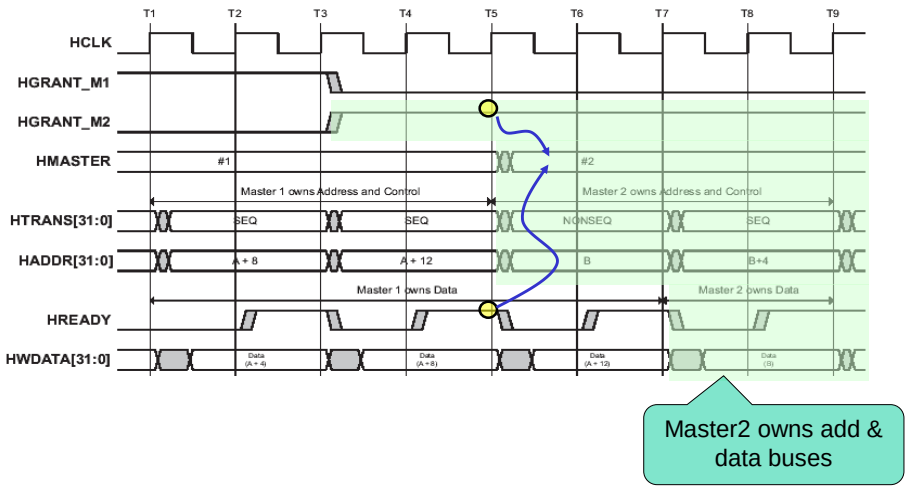
AHB arbitration without wait



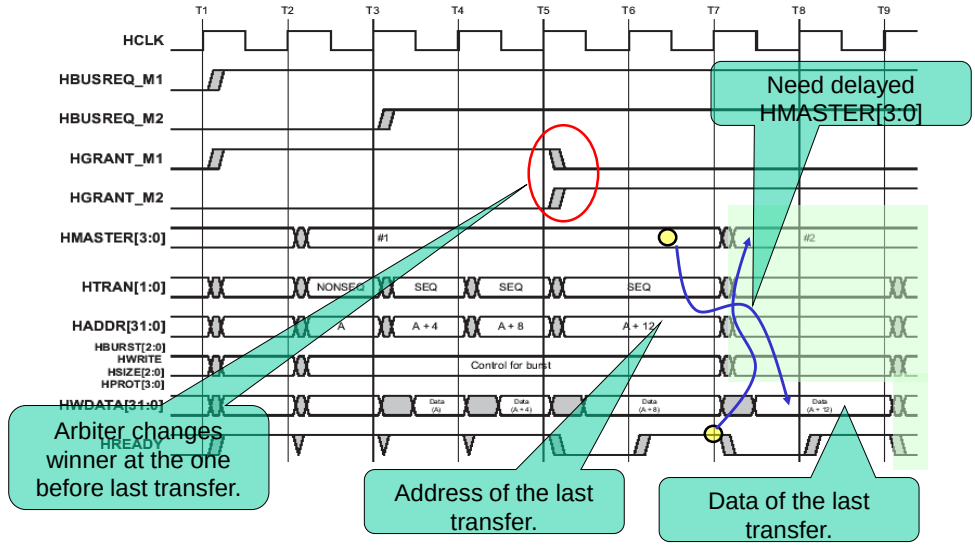
AHB arbitration with wait



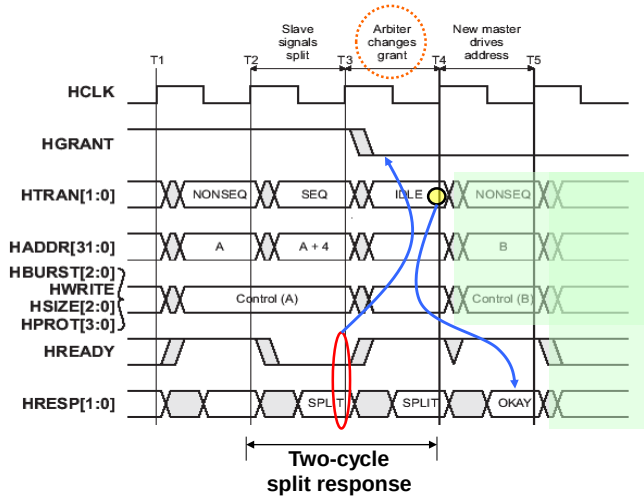
AHB arbitration: bus ownership



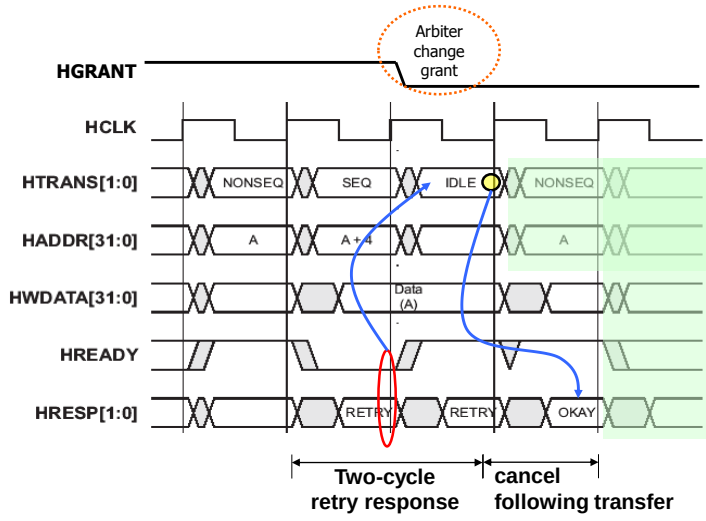
AHB arbitration: handover



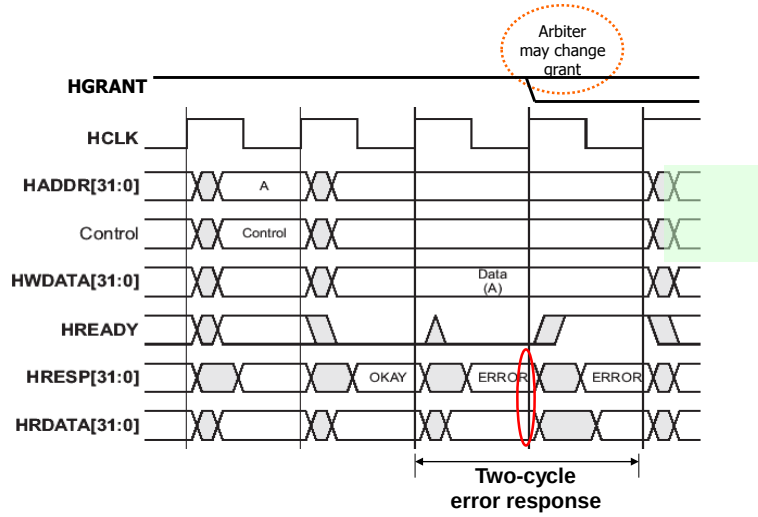
Handover after split transfer



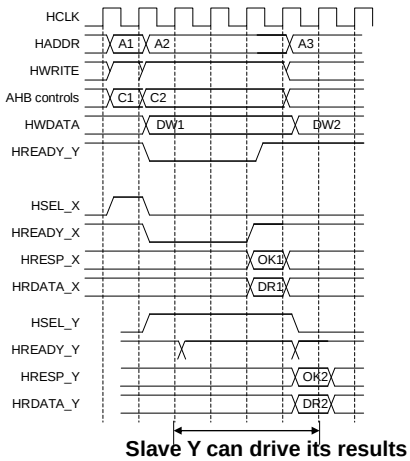
Handover after retry response



Handover after error response

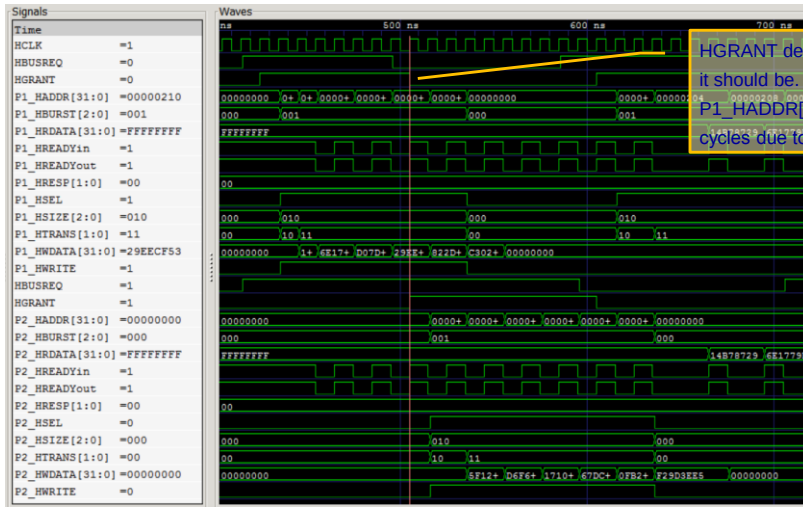


Why slave should check HREADY



- Slave Y needs to know when Slave X ends its transfer, by watching 'HRE ADY'.
  - When 'HREADY' is 1, Slave Y knows Slave X completes its transfer.

Why arbiter should check HREADY



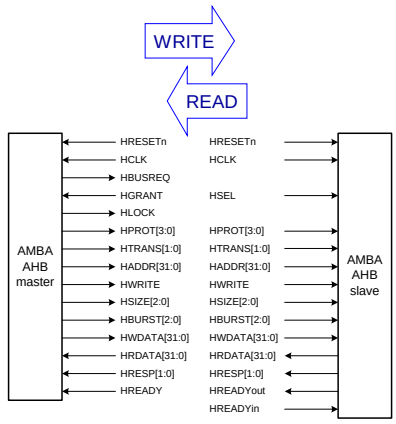
To prevent this problem, arbiter needs to check HREADY before it de-asserts HGRANT.

Data transfers in details

- Transfer direction and size
- Transfer types
- Burst operation
- Why wrapping burst is defined
- Early termination
- Protection control
- Slave transfer responses
- Two-cycle response
- Multi-byte endianness

Transfer direction and size

- HWRITE
  - 1'b1 (high): write transfer
  - 1'b0 (low): read transfer
- HSIZE[2:0] indicates the size of transfer in as beat
  - HWDATA[...] and HRDATA[...]

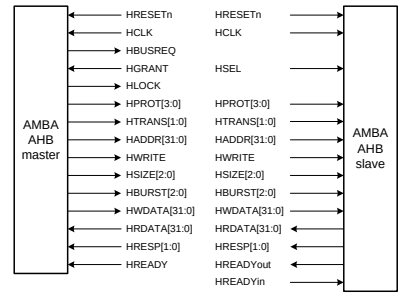


HSIZE[2]	HSIZE[1]	HSIZE[0]	Size	Description
0	0	0	8 bits	Byte
0	0	1	16 bits	Halfword
0	1	0	32 bits	Word
0	1	1	64 bits	-
1	0	0	128 bits	4-word line
1	0	1	256 bits	8-word line
1	1	0	512 bits	-
1	1	1	1024 bits	-

Transfer types

HTRANS[1:0]	Type	Description
00	IDLE	Indicates that no data transfer is required. The IDLE transfer type is used when a bus master is granted the bus, but does not wish to perform a data transfer. Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer should be ignored by the slave.
01	BUSY	The BUSY transfer type allows bus masters to insert IDLE cycles in the middle of bursts of transfers. This transfer type indicates that the bus master is continuing with a burst of transfers, but the next transfer cannot take place immediately. When a master uses the BUSY transfer type the address and control signals must reflect the next transfer in the burst. The transfer should be ignored by the slave. Slaves must always provide a zero wait state OKAY response, in the same way that they respond to IDLE transfers.
10	NONSEQ	Indicates the first transfer of a burst or a single transfer. The address and control signals are unrelated to the previous transfer. Single transfers on the bus are treated as bursts of one and therefore the transfer type is NONSEQUENTIAL.
11	SEQ	The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer. The control information is identical to the previous transfer. The address is equal to the address of the previous transfer plus the size (in bytes). In the case of a wrapping burst the address of the transfer wraps at the address boundary equal to the size (in bytes) multiplied by the number of beats in the transfer (either 4, 8 or 16).

- AHB master informs a type of transfer through 'HTRANS'.

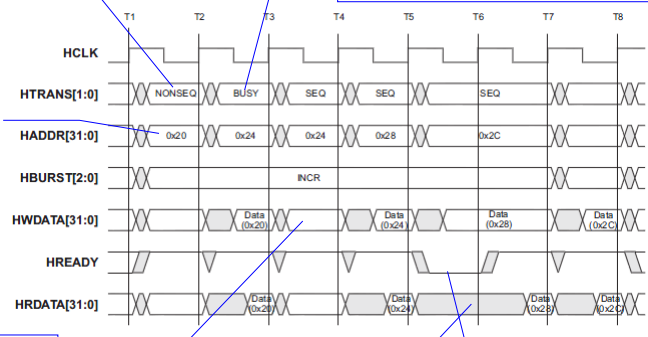


Transfer types

'NOSEQ' transfer type indicates it is the first access of a transfer.

'BUSY' means the master cannot perform transaction due to some reasons. As a result, the selected slave should wait or ignore this cycle.

a transfer accessing resources starting from 0x20



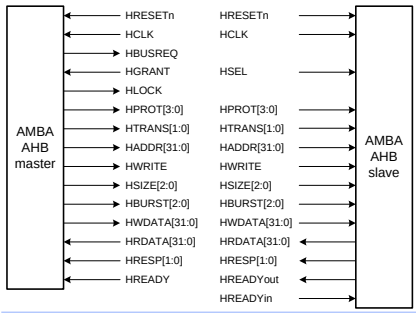
This is not valid due to 'BUSY' for the case of write.

This is not valid due to 'HREADY' for the case of read.

Low state of 'HREAY' means the selected slave cannot perform transaction due to some reasons. As a result, the master should wait or ignore this cycle.

Burst operation

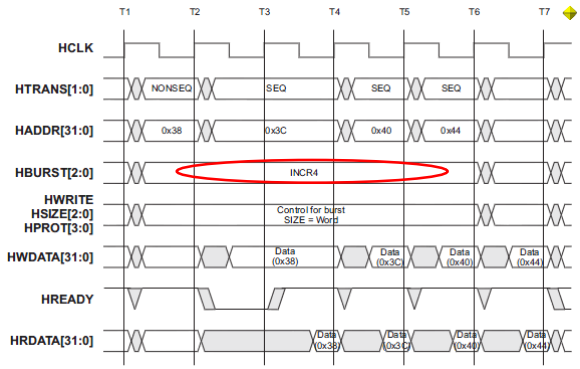
HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing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



- 1, 4, 8 and 16-beat bursts are defined in AMBA AHB.
  - Beat means a single cycle of data transfer.
  - Note that bust size does not indicates the number of bytes.
- Incremental and wrapping bursts are defined.
- Burst must not cross a 1Kbyte address boundary
  - The minimum address space that can be allocated to a single slave is 1kB.
  - (AXI spec. requires 4Kbyte boundary for burst)
- All transfers within a burst must be aligned to the address boundary equal to the size of the transfer. For example, word transfers must be aligned to word address boundaries (that is A[1:0] = 00), halfword transfers must be aligned to halfword address boundaries (that is A[0] = 0).

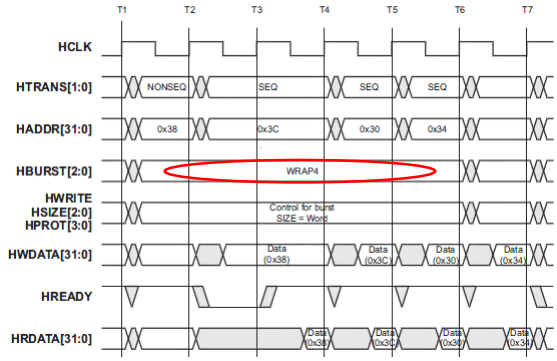
Four-beat incrementing burst

- 4-beat incremental (HBURST=3'b011) with word size (HSIZE=3'b010) starting from A
  - A, A+4, A+8, A+12
- 4-beat incremental (HBURST=3'b011) with byte size (HSIZE=3'b000) starting from A
  - A, A+1, A+2, A+3



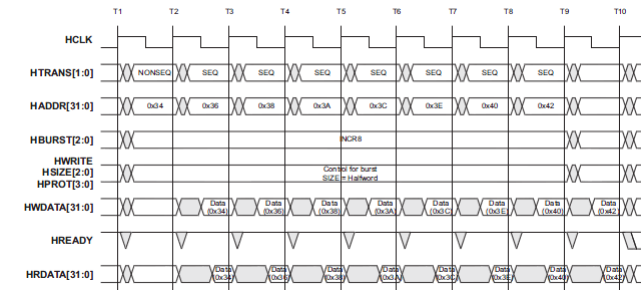
Four-beat wrapping burst

- a four-beat wrapping burst of word (4-byte) accesses will wrap at 16-byte boundaries.
  - 4-beat x 4-byte/word = 16-byte boundary



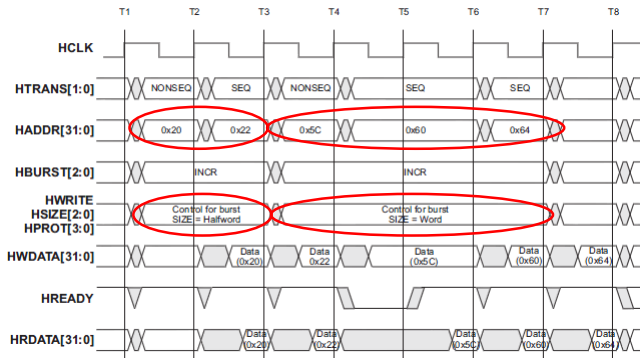
Eight-beat incrementing burst

- 8-beat incremental (HBURST=3'b101) with half-word size (i.e., 2-byte, HSIZE=3'b001) starting from A
  - A, A+2, A+4, A+6, A+8, A+10, A+12, A+14

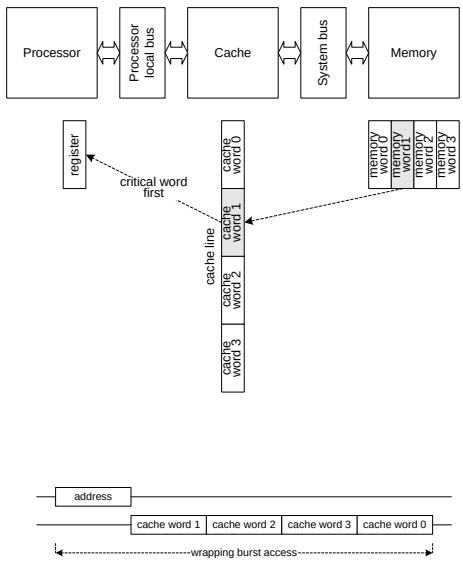


Undefined-length bursts

- Two half-word (two 2-byte) transfer starting 0x20
- Three word (three 4-byte) transfer starting 0x5C



Why wrapping burst is defined?



- Cache miss can be resolved more faster by filling the critical world first.
  - This requires wrapping burst access through system bus.
- Does critical word first policy always give a better result than sequential cache miss refill policy?
  - Think of burst-oriented memory, e.g., DDR, DDR2, DDR3

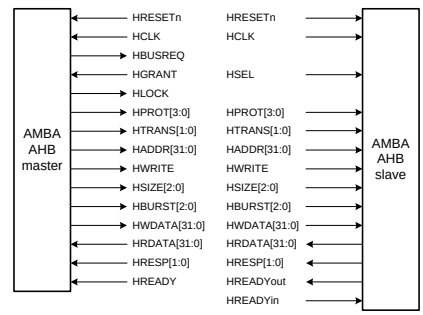
Early termination

- A fixed beat transfer should start with 'NON-SEQ' beat and be followed pre-defined 'SEQ' beats with 'BUSY' if required.
- When bus master cannot continue a complete burst transfer, it can drop the transfer at any cycles.
  - Thus, bus slave should be capable to determine an early termination by detecting 'NON-SEQ' beat in the middle of a transfer.
  - Then, bus slave ends the early terminated transfer and starts the new transfer.

Protection control

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

- More information about transfer can give more opportunity to optimize access related issues.
  - Op-code fetch means there will be no write on this access.



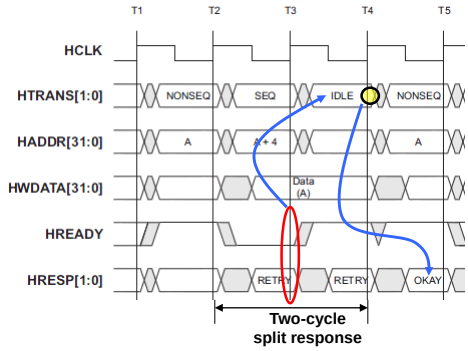
The master sets HPROT to 0b0011 to correspond to a Non-cacheable, Non-bufferable, privileged, data access.
Slaves do not use HPROT unless absolutely necessary.

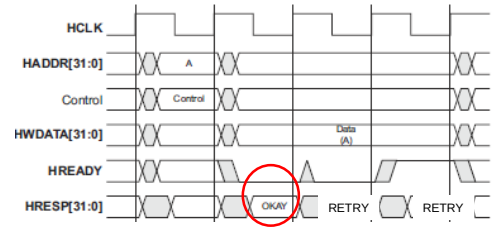
Slave transfer responses

HRESP[1]	HRESP[0]	Response	Description
0	0	OKAY	When **HREADY** is HIGH this shows the transfer has completed successfully. The OKAY response is also used for any additional cycles that are inserted, with **HREADY** LOW, prior to giving one of the three other responses.
0	1	ERROR	This response shows an error has occurred. The error condition should be signalled to the bus master so that it is aware the transfer has been unsuccessful. A two-cycle response is required for an error condition.
1	0	RETRY	The RETRY response shows the transfer has not yet completed, so the bus master should retry the transfer. The master should continue to retry the transfer until it completes. A two-cycle RETRY response is required.
1	1	SPLIT	The transfer has not yet completed successfully. The bus master must retry the transfer when it is next granted access to the bus. The slave will request access to the bus on behalf of the master when the transfer can complete. A two-cycle SPLIT response is required.

- After a master has started a transfer, the slave then determines how the transfer should progress. No provision is made within the AHB specification for a bus master to cancel a transfer once it has commenced.
  - There is no way to cancel transfer by master.
- The SPLIT and RETRY responses provide a mechanism for slaves to release the bus when they are unable to supply data for a transfer immediately. Both mechanisms allow the transfer to finish on the bus and therefore allow a higher-priority master to get access to the bus.

Two-cycle response





- Only an OKAY response can be given in a single cycle. The ERROR, SPLIT and RETRY responses require at least two cycles
- By driving one more 'RETRY/ERROR/SPLIT' respond before actual response, the master can put 'IDLE' cycle before starting the next cycle.
- This 'IDLE' cycle will be cancelled due to 'RETRY/ERROR/SPLIT'.
- When the slave needs more than two cycles to respond 'RETRY/ERROR/SPLIT', it puts 'OKAY' respond with 'HREADY' is low before two-cycle response.

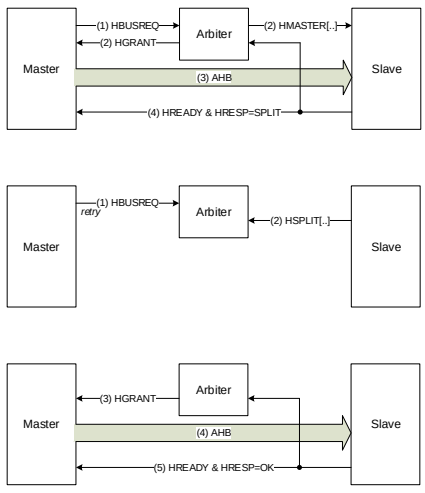
Retry and split

- The SPLIT and RETRY responses provide a mechanism for slaves to release the bus when they are unable to supply data for a transfer immediately.
  - Both mechanisms allow the transfer to finish on the bus and therefore allow a higher-priority master to get access to the bus.

- RETRY
  - the arbiter will continue to use the normal priority scheme
  - thus, higher priority master can gain the bus.
  - 슬레이브가 RETRY 응답을 하면, master는 전송을 재시도하고, 중재기는 원래 중재방식으로 중재한다. → 즉, retry 한 것을 포함하여 최고 우선순위가 다시 버스를 사용한다.

- SPLIT
  - the arbiter adjust priority by using information from slave through HSPLIT[15:0].
  - SPLIT 응답을 받은 마스터는 retry한다. 다만 중재기가 HSPLIT이 올 때까지 중재에서 배제하고, HSPLIT가 오면 우선순위에 무관하게 먼저 버스 사용을 허락한다.
  - A bus master should treat SPLIT and RETRY in the same manner. It should continue to request the bus and attempt the transfer until it has either completed successfully or been terminated with an ERROR response.

Split



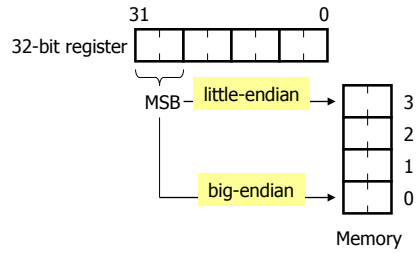
Multi-byte endianness

- Little-endian data bus for 32-bit case

Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	-	-	✓	✓
Halfword	2	✓	✓	-	-
Byte	0	-	-	-	✓
Byte	1	-	-	✓	-
Byte	2	-	✓	-	-
Byte	3	✓	-	-	-

- Big-endian data bus for 32-bit case
  - Word-invariance scheme (BE32)

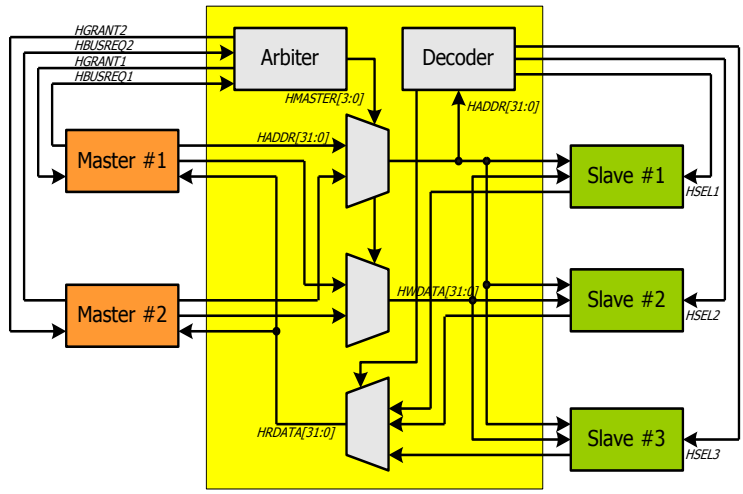
Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	✓	✓	-	-
Halfword	2	-	-	✓	✓
Byte	0	✓	-	-	-
Byte	1	-	✓	-	-
Byte	2	-	-	✓	-
Byte	3	-	-	-	✓



AMBA bus systems

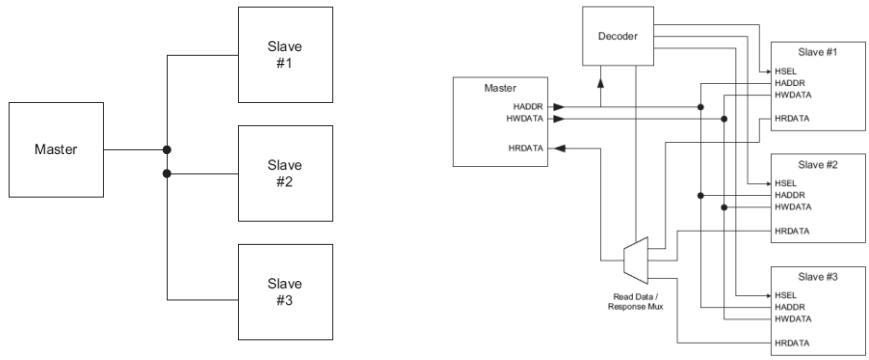
- AMBA AHB bus system
- AMBA AHB-Lite
- Multi-layer AHB

AMBA AHB bus system



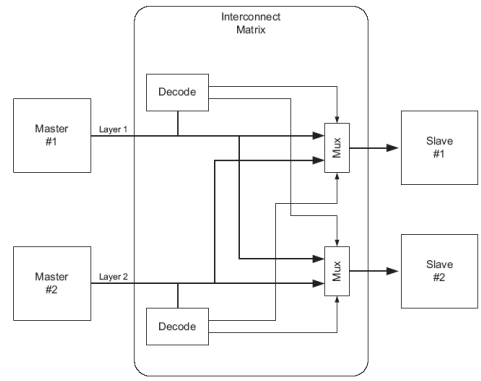
AMBA AHB-Lite

- AHB-Lite is a subset of the full AHB specification, where only a single AHB master is used.
  - A single master → No master-to-slave multiplexor
  - No request/grant protocol to the arbiter → No arbiter
  - No split/retry responses from slaves



Multi-layer AHB

- Multi-layer AHB is an interconnection scheme, based on the AHB protocol, that enables parallel access paths between multiple masters and slaves in a system.
- Standard AHB master and slave modules can be used without modification.
- Bus contention can be avoided.



Issues and quiz

- How to deal with accesses to non-existent locations
- How to deal with bus grant for no requesting
- Q1: How does master postpone cycle(s) of transfer?
  - How to make slave wait by the master?
- Q2: How does slave postpone cycle(s) of transfer?
  - How to make master wait by the slave?
- Q3: Why slave needs to see HREADY?

References

- ARM, AMBA Specification, Chapter 3 AMBA AHB, ARM IHI 0011A, 1999.
- ARM, AMBA3 AHB-Lite Protocol Specification, v1.0, ARM IHI 0033A, 2006.
- ARM, Multi-layer AHB Overview, ARM DVI 0045B, 2004.
- ARM, AMBA 5 AHB Protocol Specification, AHB5, AHB-Lite, ARM IHI 0033B, 2015.