AMBA AHB Protocol

By: A R. Chaurasiya

June 2008

Agenda

- Features
- AMBA Based System
- AHB Components
- AHB Signal Descriptions
- AHB Single Read/Write
- AHB Control Signals
- AHB Burst Operation
- Address Decoding
- Slave Responses
- Data Buses
- Arbitration
- Split Transfers

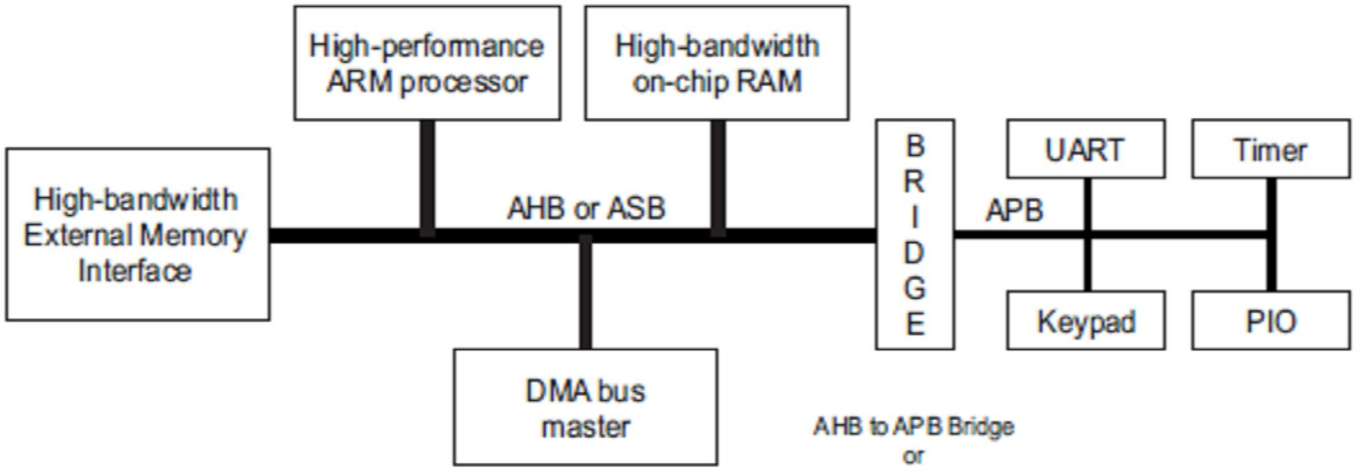
Features

AHB - Advanced High-performance Bus

Implements features for high performance and high frequency systems :

- Burst transfers
- Split transactions
- Pipelined Operation
- Single-clock edge operation
- Non-tristate implementation
- Single-cycle bus master handover
- Wider data bus configurations (64/128 bits)
- Multiple Master Multiple Slave handling Capability.

AMBA Based System



AMBA AHB

* High performance
* Pipelined operation
* Multiple bus masters
* Burst transfers
* Split transactions

AMBA ASB

* High performance
* Pipelined operation
* Multiple bus masters

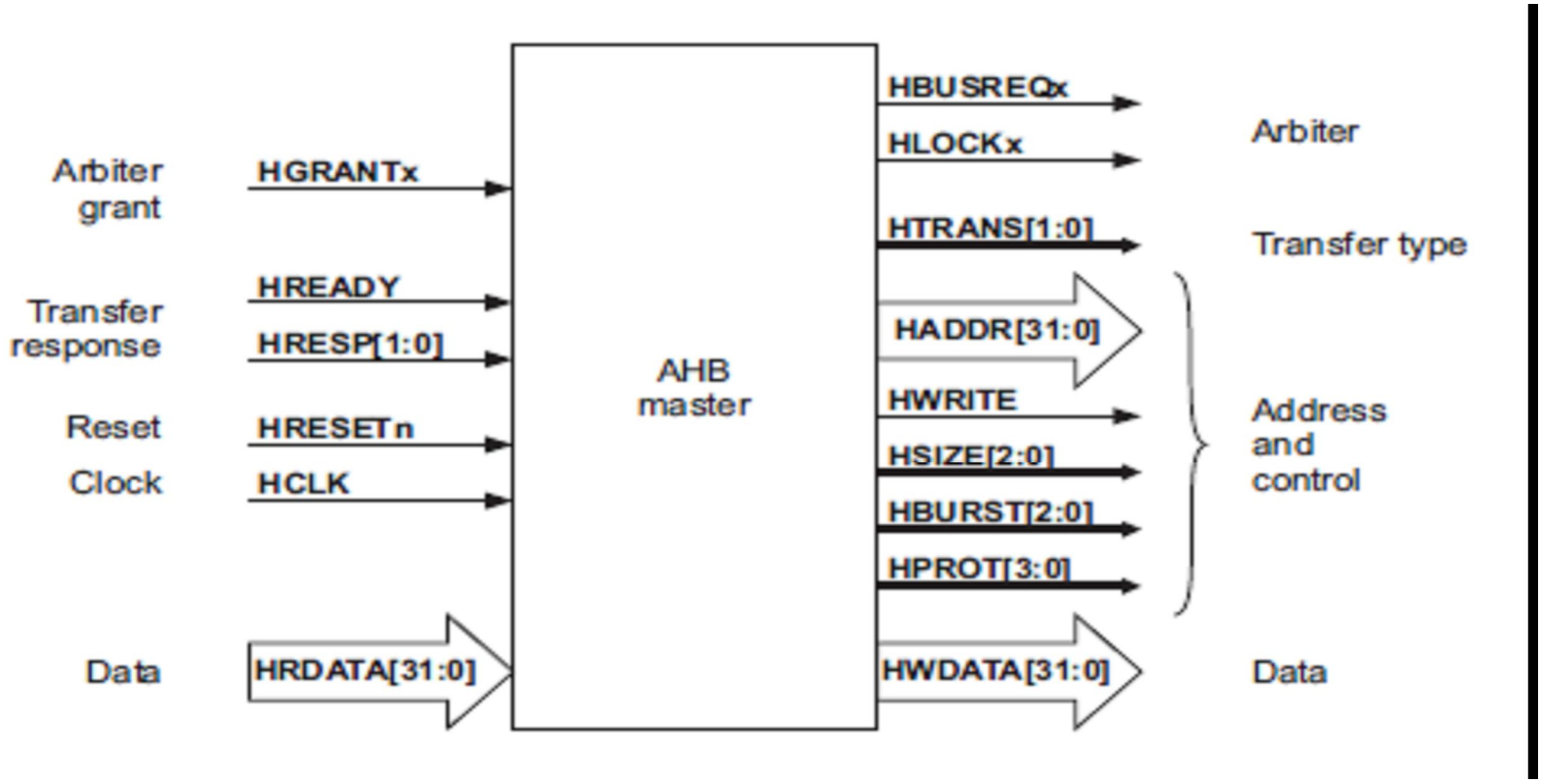
AMBA APB

* Low power
* Latched address and control
* Simple interface
* Suitable for many peripherals

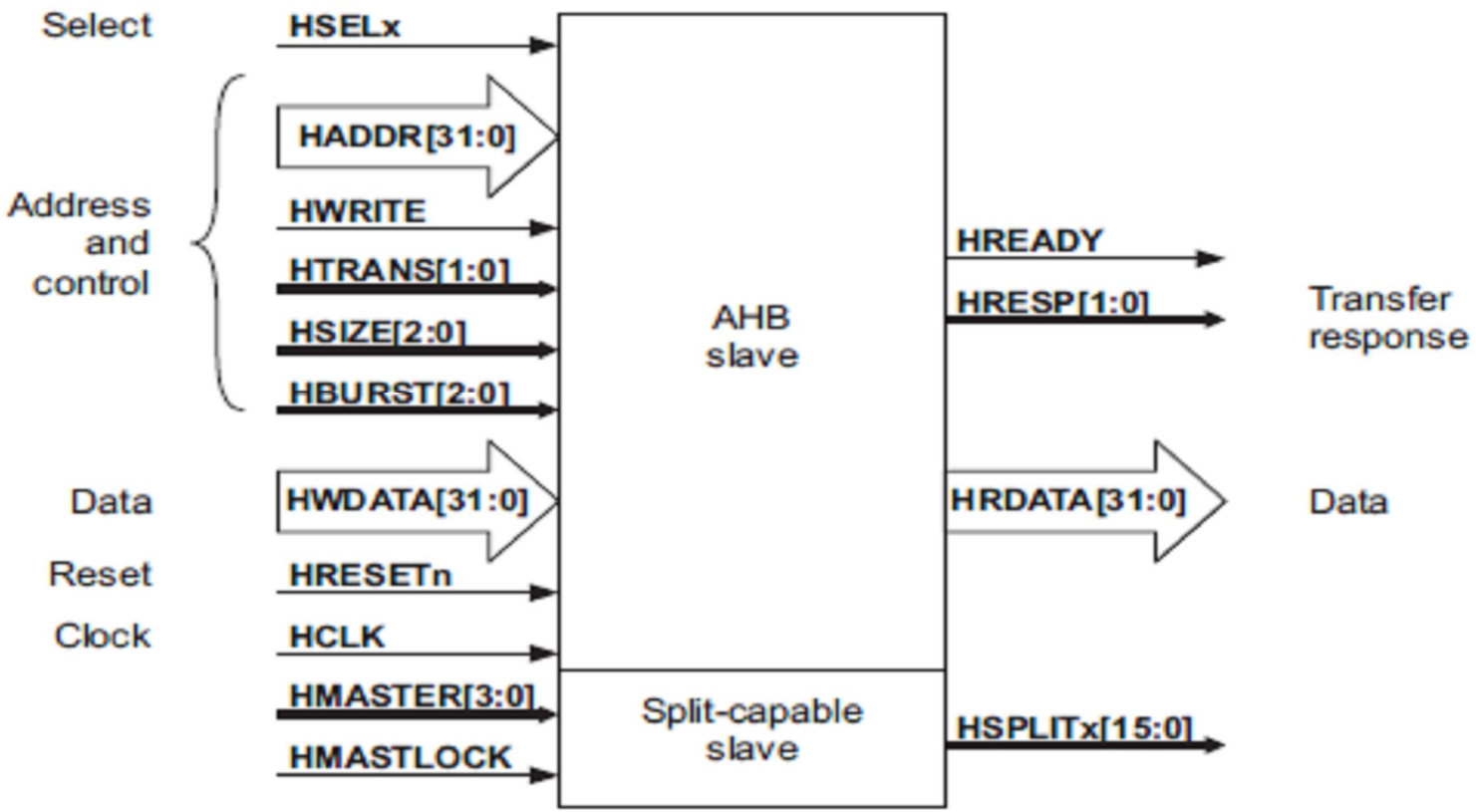
AHB Components

- Master
- Slave
- Arbiter
- AHB Signal Descriptions
- Arbitration Signals

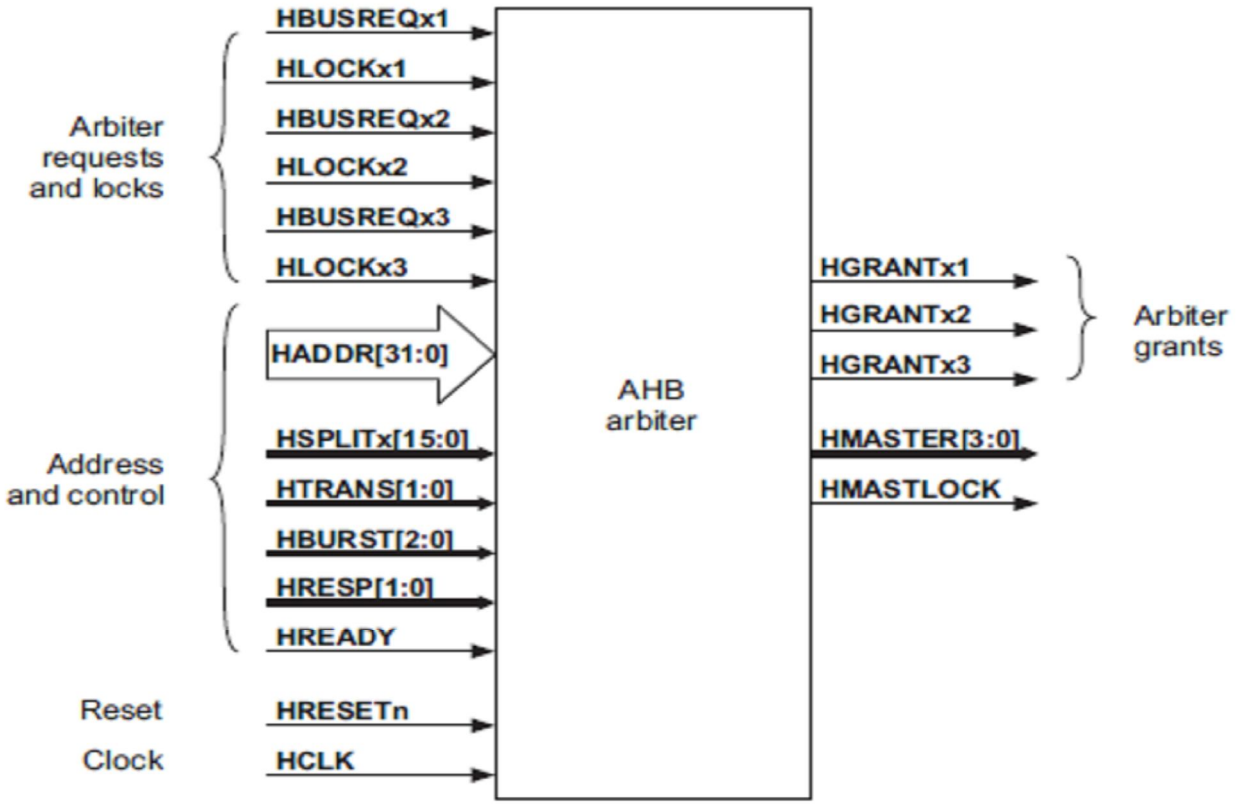
AHB Master


Arbiter grant
HGRANTx
Transfer response
HREADY
HRESP[1:0]
Reset
HRESETn
Clock
HCLK
Data
HRDATA[31:0]
AHB master
HBUSREQx
HLOCKx
HTRANS[1:0]
HADDR[31:0]
HWRITE
HSIZE[2:0]
HBURST[2:0]
HPROT[3:0]
HWDATA[31:0]
Arbiter
Transfer type
Address and control
Data

AHB Slave


Select
HSELx
Address and control
HADDR[31:0]
HWRITE
HTRANS[1:0]
HSIZE[2:0]
HBURST[2:0]
Data
HWDATA[31:0]
Reset
HRESETn
Clock
HCLK
HMASTER[3:0]
HMASTLOCK
AHB slave
Split-capable slave
HREADY
HRESP[1:0]
Transfer response
HRDATA[31:0]
Data
HSPLITx[15:0]

AHB Arbiter


HBUSREQx1
HLOCKx1
HBUSREQx2
HLOCKx2
HBUSREQx3
HLOCKx3
Arbiter requests and locks
HADDR[31:0]
HSPLITx[15:0]
HTRANS[1:0]
HBURST[2:0]
HRESP[1:0]
HREADY
Address and control
Reset
HRESETn
Clock
HCLK
AHB arbiter
HGRANTx1
HGRANTx2
HGRANTx3
Arbiter grants
HMASTER[3:0]
HMASTLOCK

AHB Signal Descriptions

Name	Source	Description
HCLK Bus clock	Clock source	This clock times all bus transfers. All signal timings are related to the rising edge of **HCLK**.
HRESETn Reset	Reset controller	The bus reset signal is active LOW and is used to reset the system and the bus. This is the only active LOW signal.
HADDR[31:0] Address bus	Master	The 32-bit system address bus.
HTRANS[1:0] Transfer type	Master	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE Transfer direction	Master	When HIGH this signal indicates a write transfer and when LOW a read transfer.
HSIZE[2:0] Transfer size	Master	Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
HBURST[2:0] Burst type	Master	Indicates if the transfer forms part of a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.
HPROT[3:0] Protection control	Master	The protection control signals provide additional information about a bus access and are primarily intended for use by any module that wishes to implement some level of protection. The signals indicate if the transfer is an opcode fetch or data access, as well as if the transfer is a privileged mode access or user mode access. For bus masters with a memory management unit these signals also indicate whether the current access is cacheable or bufferable.

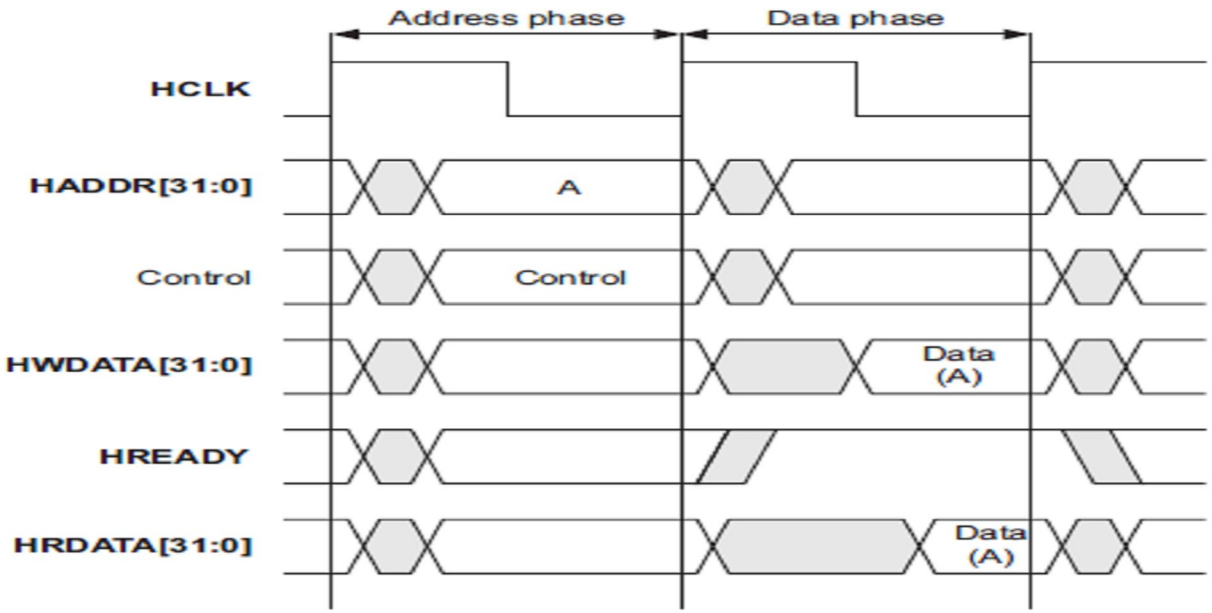
Name	Source	Description
HWDATA[31:0] Write data bus	Master	The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HSEL*x* Slave select	Decoder	Each AHB slave has its own slave select signal and this signal indicates that the current transfer is intended for the selected slave. This signal is simply a combinatorial decode of the address bus.
HRDATA[31:0] Read data bus	Slave	The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HREADY Transfer done	Slave	When HIGH the **HREADY** signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer. Note: Slaves on the bus require **HREADY** as both an input and an output signal.
HRESP[1:0] Transfer response	Slave	The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT.

Arbitration Signals

Name	Source	Description
HBUSREQx Bus request	Master	A signal from bus master x to the bus arbiter which indicates that the bus master requires the bus. There is an **HBUSREQx** signal for each bus master in the system, up to a maximum of 16 bus masters.
HLOCKx Locked transfers	Master	When HIGH this signal indicates that the master requires locked access to the bus and no other master should be granted the bus until this signal is LOW.
HGRANTx Bus grant	Arbiter	This signal indicates that bus master x is currently the highest priority master. Ownership of the address/control signals changes at the end of a transfer when **HREADY** is HIGH, so a master gets access to the bus when both **HREADY** and **HGRANTx** are HIGH.
HMASTER[3:0] Master number	Arbiter	These signals from the arbiter indicate which bus master is currently performing a transfer and is used by the slaves which support SPLIT transfers to determine which master is attempting an access. The timing of **HMASTER** is aligned with the timing of the address and control signals.
HMASTLOCK Locked sequence	Arbiter	Indicates that the current master is performing a locked sequence of transfers. This signal has the same timing as the **HMASTER** signal.
HSPLITx[15:0] Split completion request	Slave (SPLIT-capable)	This 16-bit split bus is used by a slave to indicate to the arbiter which bus masters should be allowed to re-attempt a split transaction. Each bit of this split bus corresponds to a single bus master.

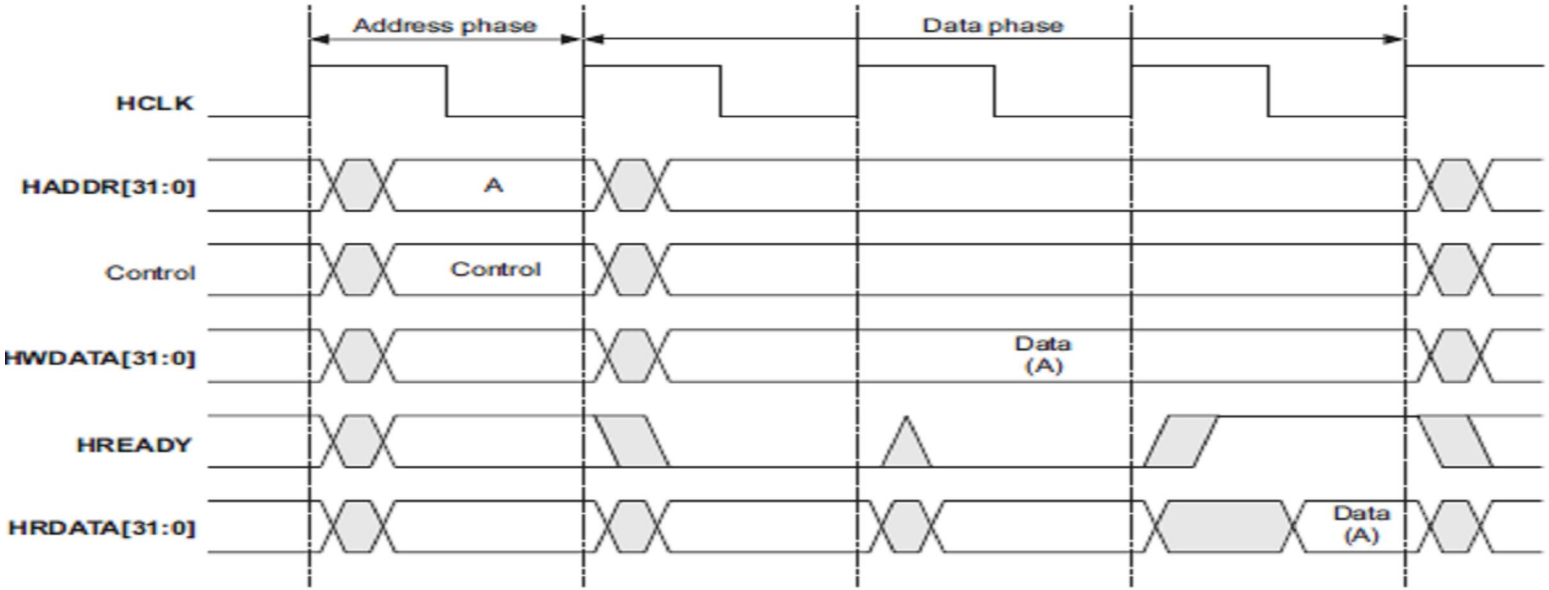
Single Read/Write

- Master Drives Address & Control signals after rising edge of HCLK.
- Slave Samples the Address & Control information on the next rising edge of the HCLK.
- Then slave drives the appropriate response.
- This is sampled by master on the third rising edge of the clock.



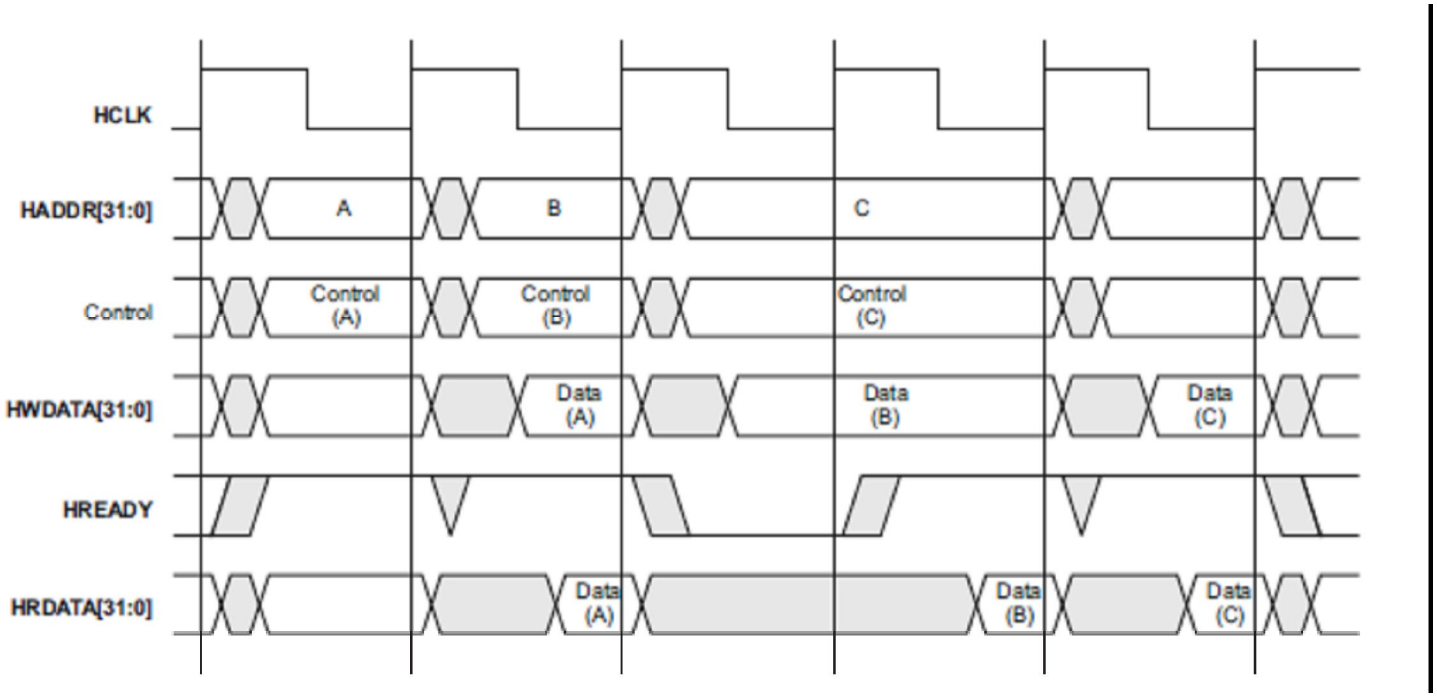
Transfer with wait states

- For Write Operations bus master will hold the data stable throughout the extended cycles.
- For read transfers the slave does not have to provide valid data untill the transfer is about to complete



Multiple Transfers

- Transfer to addresses A & C are both zero wait states .
- The transfer to address B is one wait state.
- Extending the data phase of the transfer to address B has the effect of extending the address phase of the transfer to address C.



Control Signals

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

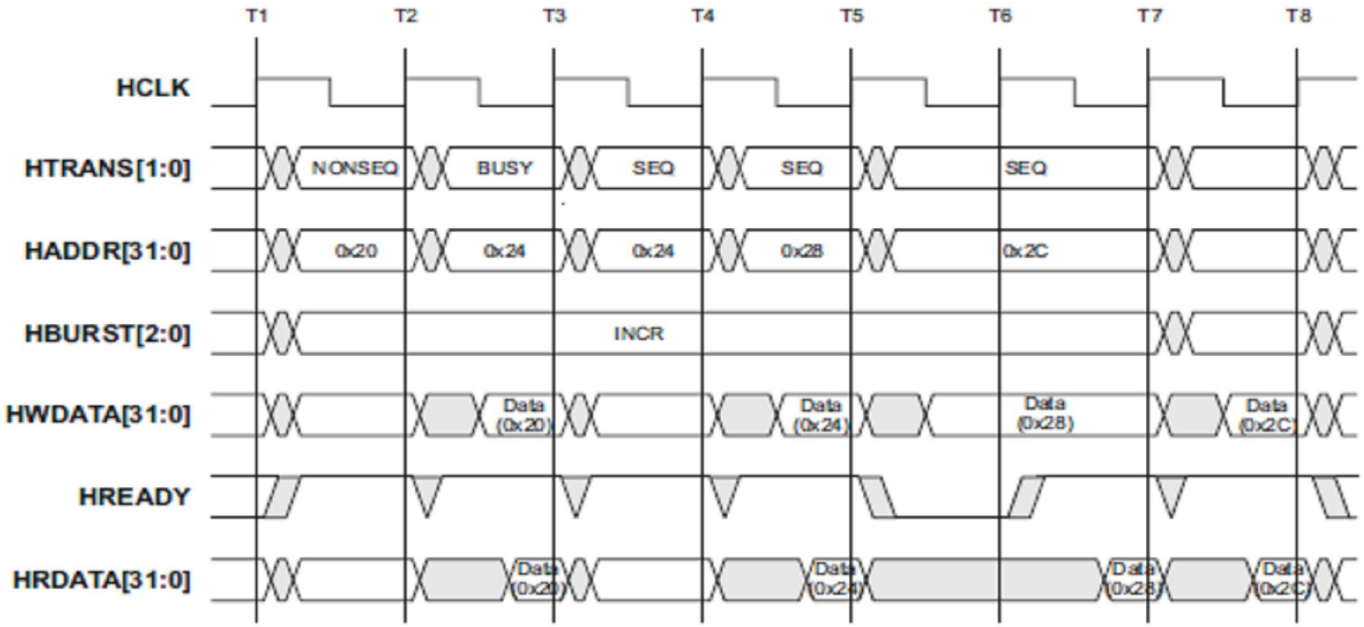
- Transfer Direction – HWRITE
  - High – Write Transfer, Master will broadcast the data on HWDATA[31:0].
  - Low – Read Transfer, Slave must generate the data on HRDATA[31:0].

Transfer Type

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).

Transfer Type Examples

- The first transfer is the start of a burst and therefore is NONSEQUENTIAL.
- The master is unable to perform the second transfer of the burst immediately and therefore the master uses a BUSY transfer to delay the start of the next transfer.
- The master performs the third transfer of the burst immediately, but this time the slave is unable to complete and uses **HREADY** to insert a single wait state.
- The final transfer of the burst completes with zero wait states.



Burst Type

- The burst size indicates the number of beats in the burst, not the number of bytes transferred.
- Incrementing Bursts access sequential locations and the address of each transfer in the burst is just an increment of the previous address.
- For wrapping burst, the address of the transfers in the burst will wrap when the boundary is reached.
- Burst must not cross 1 KB boundary.

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

Transfer Size

- HSIZE indicates the size of the transfers.
- The size is used in conjunction with the **HBURST[2:0]** signals to determine the address boundary for wrapping bursts.

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	-

Protection Control

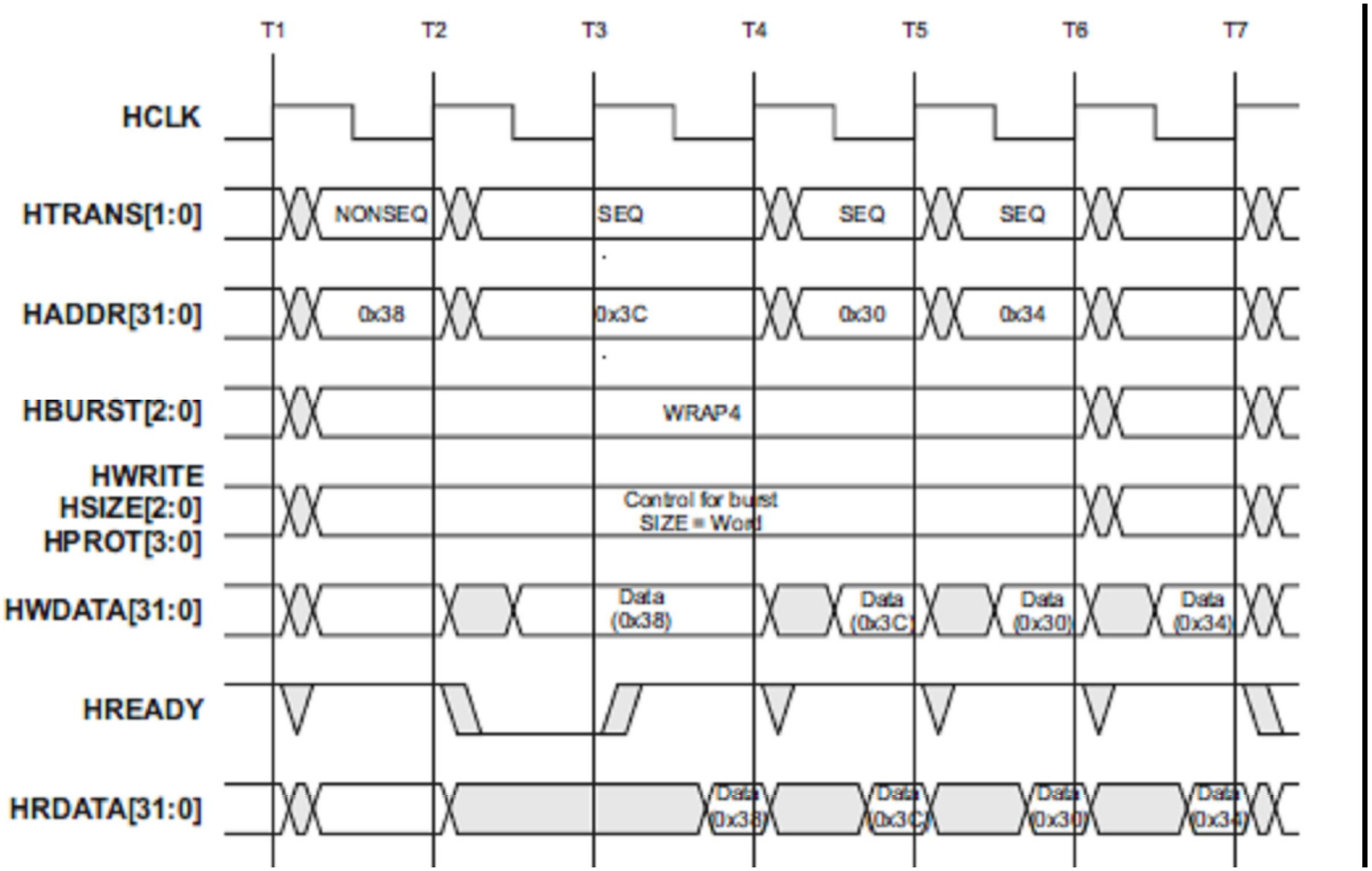
- provide additional information about a bus access and are primarily intended for use by any module that wishes to implement some level of protection.
- Indicates :
  - An opcode fetch or data access
  - A privileged mode access or user mode access.
  - Bufferable or Not Bufferable.
  - Cacheable or Not Cacheable.

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

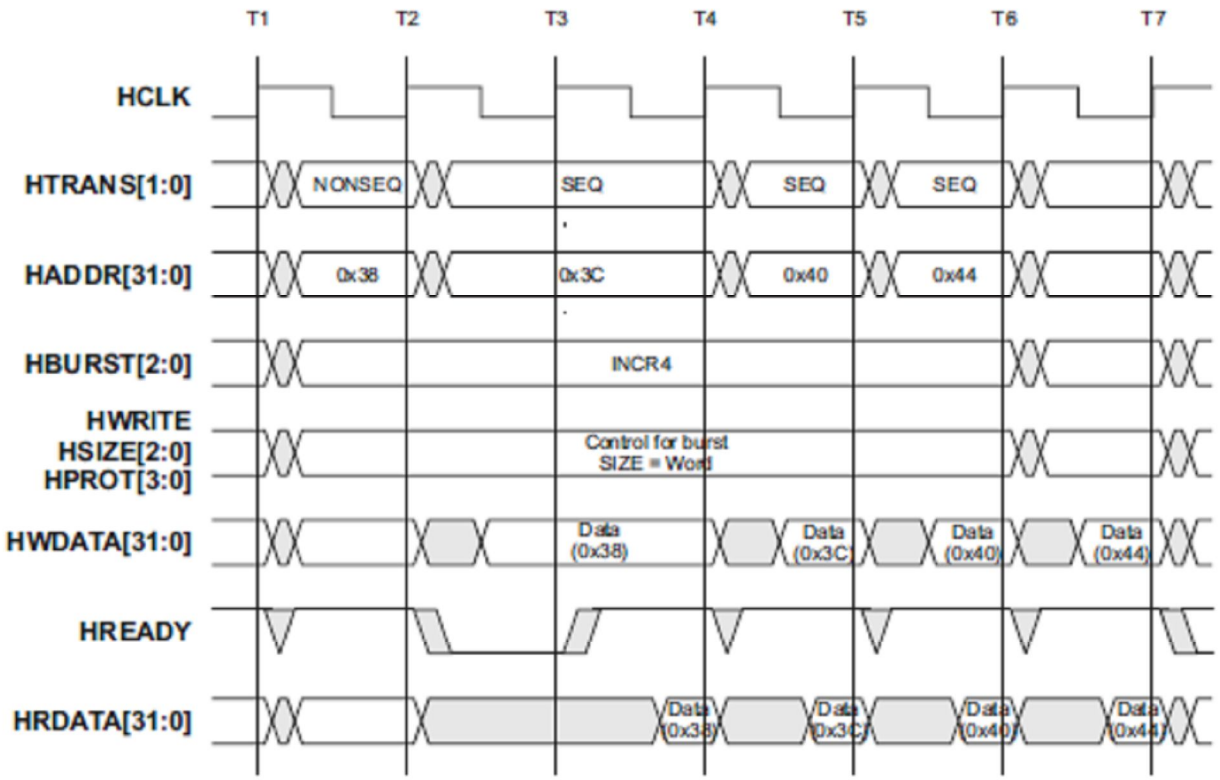
AHB Burst Operation

- Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol,
- Undefined-length bursts and single transfers.
- Both incrementing and wrapping bursts are supported in the protocol.
- The total amount of data transferred in a burst is calculated by multiplying the number of beats (as indicated by HBURST) by the amount of data in each beat(as indicated by **HSIZE[2:0])**.
- All transfers within a burst must be aligned to the address boundary equal to the size of the transfer.

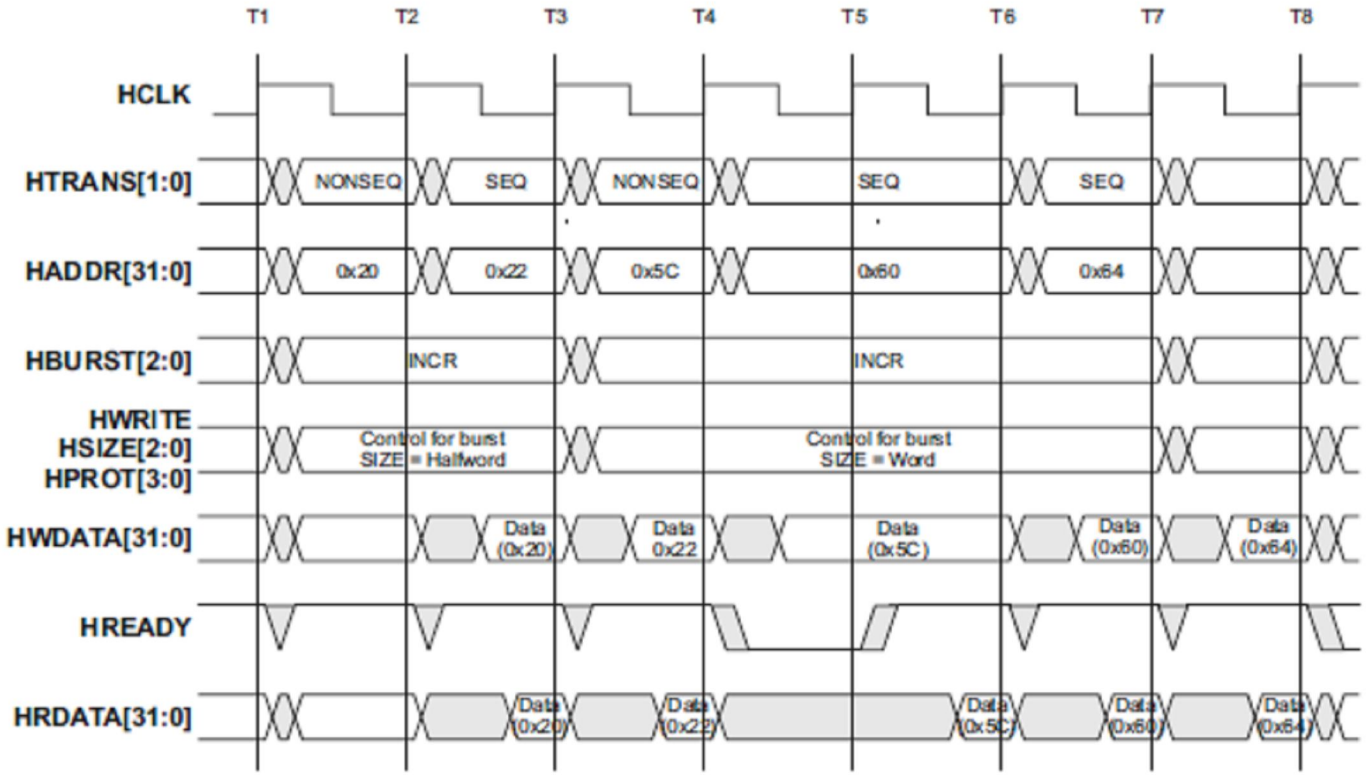
Burst Examples: 4 beat Wrapping Burst



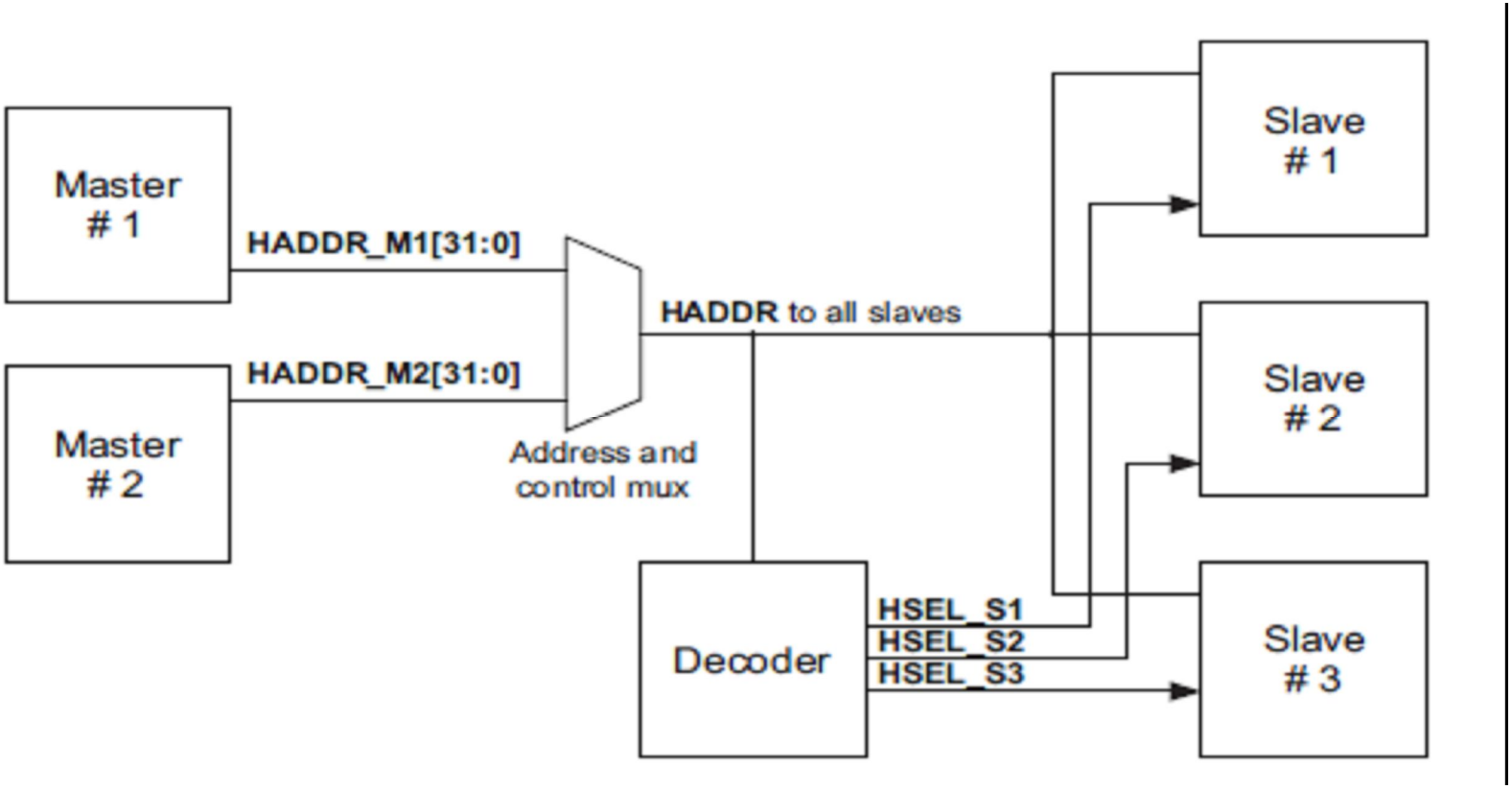
Burst Examples: 4 beat Incrementing Operation



Burst Example: Undefined length Bursts



Address Decoding



Address Decoding

- A slave must only sample the address and control signals and **HSELx** when **HREADY** is HIGH.
- The minimum address space that can be allocated to a single slave is 1kB.
- All bus masters are designed such that they will not perform incrementing transfers over a 1kB boundary, thus ensuring that a burst never crosses an address decode boundary.
- A default slave should be implemented to provide a response when any of the nonexistent address locations are accessed.
- If a NONSEQUENTIAL or SEQUENTIAL transfer is attempted to a nonexistent address location then the default slave should provide an ERROR response.
- IDLE or BUSY transfers to nonexistent locations should result in a zero wait state OKAY response.

Slave Responses

- HRESP description
- Slave Response Rules
- Two Cycle Responses
- Transfer with a Retry Response
- Transfer with a Error Response
- SPLIT & RETRY

HRESP Description

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.

Slave Response rules

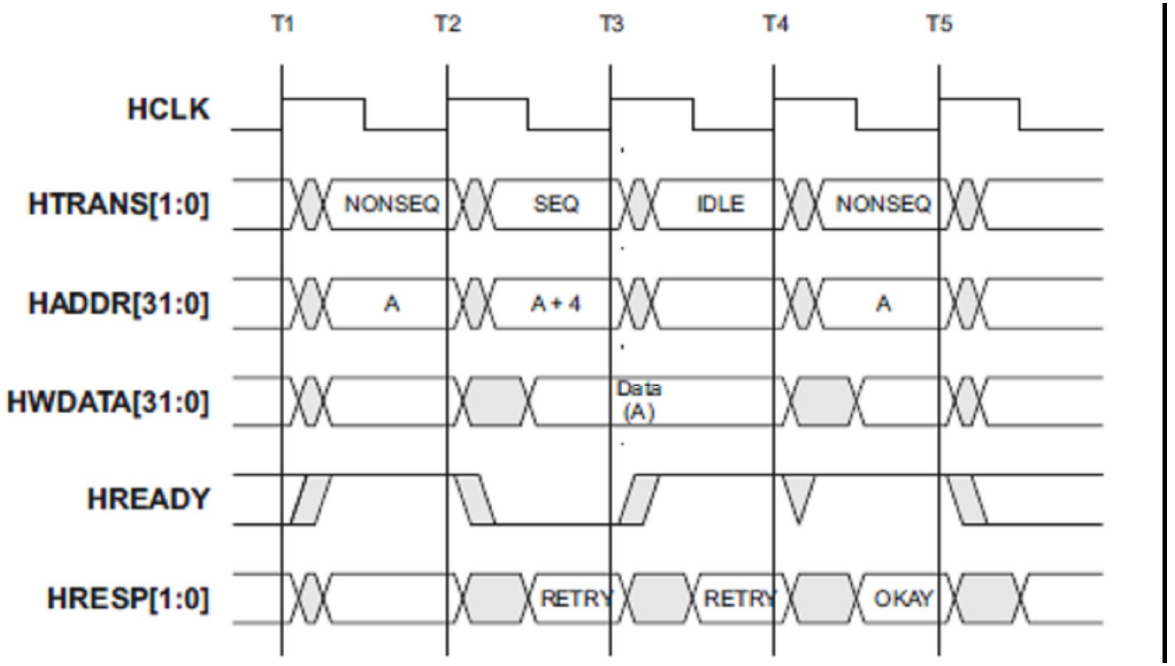
- A master can not cancel a transfer once it has commenced.
- Slave must provide a response which indicates the status of the transfer
- HREADY signal is used to extend the transfer.
- HRESP provides the status of the transfer.
- Slave can complete the transfer by:
  - Completing the transfer immediately; HREADY=1, HRESP=OKAY.
  - Inserting one or more wait states to allow time to complete the transfer.
  - Giving Error response indicating failure of transfer.
  - Delaying the completion of the transfer(not more than Time out value), but allow the master and slave to back off.

Two Cycle Responses

- The ERROR, SPLIT and RETRY responses require at least two cycles.
- If the slave needs more than two cycles to provide the ERROR, SPLIT or RETRY response then additional wait states may be inserted at the start of the transfer.
- The two-cycle response is required because of the pipelined nature of the bus. By the time a slave starts to issue either an ERROR, SPLIT or RETRY response then the address for the following transfer has already been broadcast onto the bus.

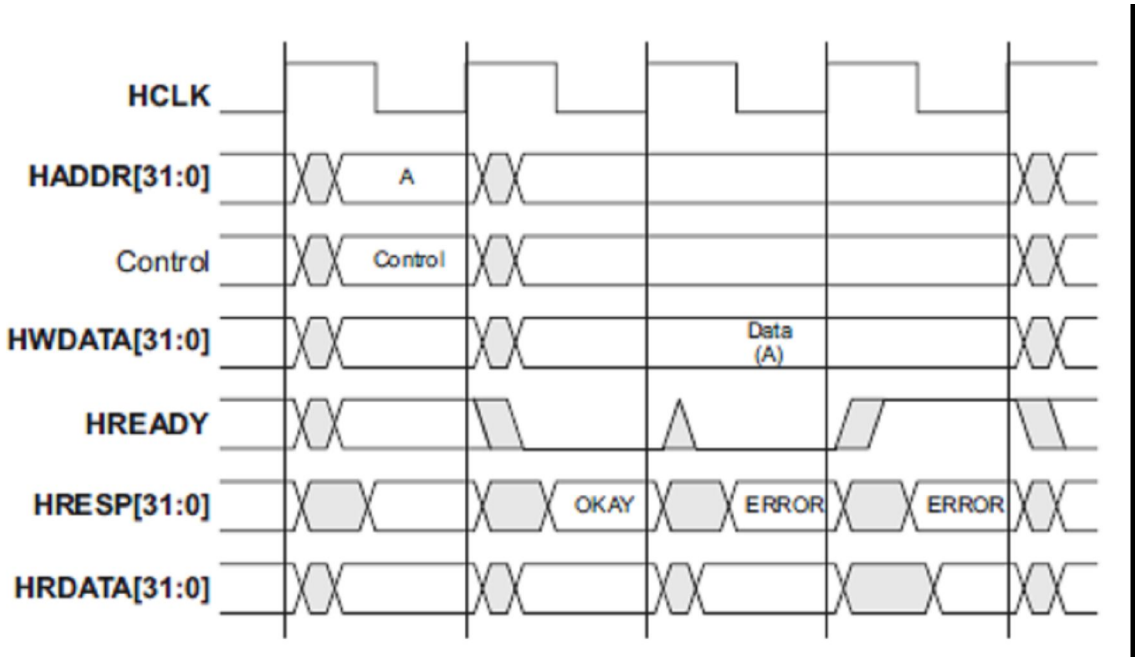
Transfer with a Retry Response

- If a slave provides an ERROR response then the master may choose to cancel the remaining transfers in the burst.
- However, this is not a strict requirement and it is also acceptable for the master to continue the remaining transfers in the burst.
- A slave which issues RETRY responses must only be accessed by one master at a time. It is not enforced by Protocol but should be ensured by system architecture.



Transfer with a Error Response

- If a slave provides an ERROR response then the master may choose to cancel the remaining transfers in the burst.
- However, this is not a strict requirement and it is also acceptable for the master to continue the remaining transfers in the burst.



Split & Retry

- 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.
- Difference between Split & Retry:
  - For RETRY the arbiter will continue to use the normal priority scheme and therefore only masters having a higher priority will gain access to the bus.
  - For a SPLIT transfer the arbiter will adjust the priority scheme so that any other master requesting the bus will get access, even if it is a lower priority. In order for a SPLIT transfer to complete the arbiter must be informed when the slave has the data available.
- The SPLIT transfer requires extra complexity in both the slave and the arbiter, but has the advantage that it completely frees the bus for use by other masters, whereas the RETRY case will only allow higher priority masters onto the bus.

Data Buses

- The minimum data bus width is specified as 32 bits.
- For transfers that are narrower than the minimum width of the bus, for example a 16-bit transfer on a 32-bit bus, then the bus master only has to drive the appropriate byte lanes.
- The slave is responsible for selecting the write data from the correct byte lanes.
- Burst transfers which have a transfer size less than the width of the data bus will have different active byte lanes for each beat of the burst.
- The active byte lane is dependent on the endianness of the system.
- AHB does not specify the required endianness. Therefore, it is important that all masters and slaves on the bus are of the same endianness.

Data Buses : Active Byte lanes for Little endian data bus

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	✓	-	-	-

Data Buses : Active Byte lanes for Big endian data bus

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	-	-	-	✓

Arbitration

- Requesting Bus Access
- Granting Bus Access
- Bus Handover after Burst
- Early Burst Termination
- Locked Transfers
- Default Master

Arbitration

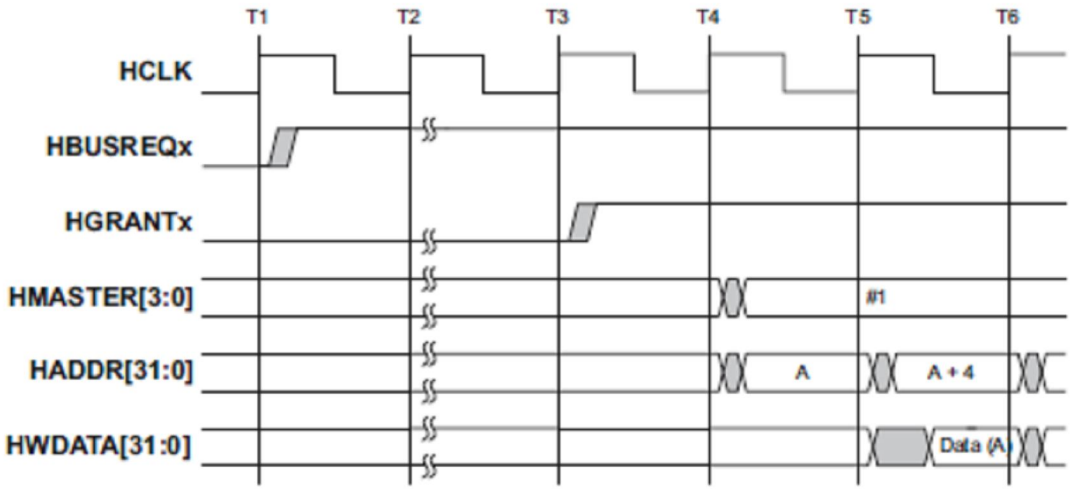
- Normally the arbiter will only grant a different bus master when a burst is completing.
- However, if required, the arbiter can terminate a burst early to allow a higher priority master access to the bus.
- If the master requires locked accesses then it must also assert the **HLOCKx** signal to indicate to the arbiter that no other masters should be granted the bus.

Requesting Bus Access

- A bus master uses the **HBUSREQx** signal to request access to the bus and may request the bus during any cycle.
- The arbiter will sample the request on the rising of the clock and then use an internal priority algorithm to decide which master will be the next to gain access to the bus.
- When a master is granted the bus and is performing a fixed length burst it is not necessary to continue to request the bus in order to complete the burst.
- The arbiter observes the progress of the burst and uses the **HBURST[2:0]** signals to determine how many transfers are required by the master.
- If the master wishes to perform a second burst after the one that is currently in progress then it should re-assert the request signal during the burst.
- If a master loses access to the bus in the middle of a burst then it must re-assert the **HBUSREQx** request line to regain access to the bus.
- For undefined length bursts the master should continue to assert the request until it has started the last transfer.
- if a master does not require access to the bus it drives the transfer type **HTRANS** to indicate an IDLE transfer.

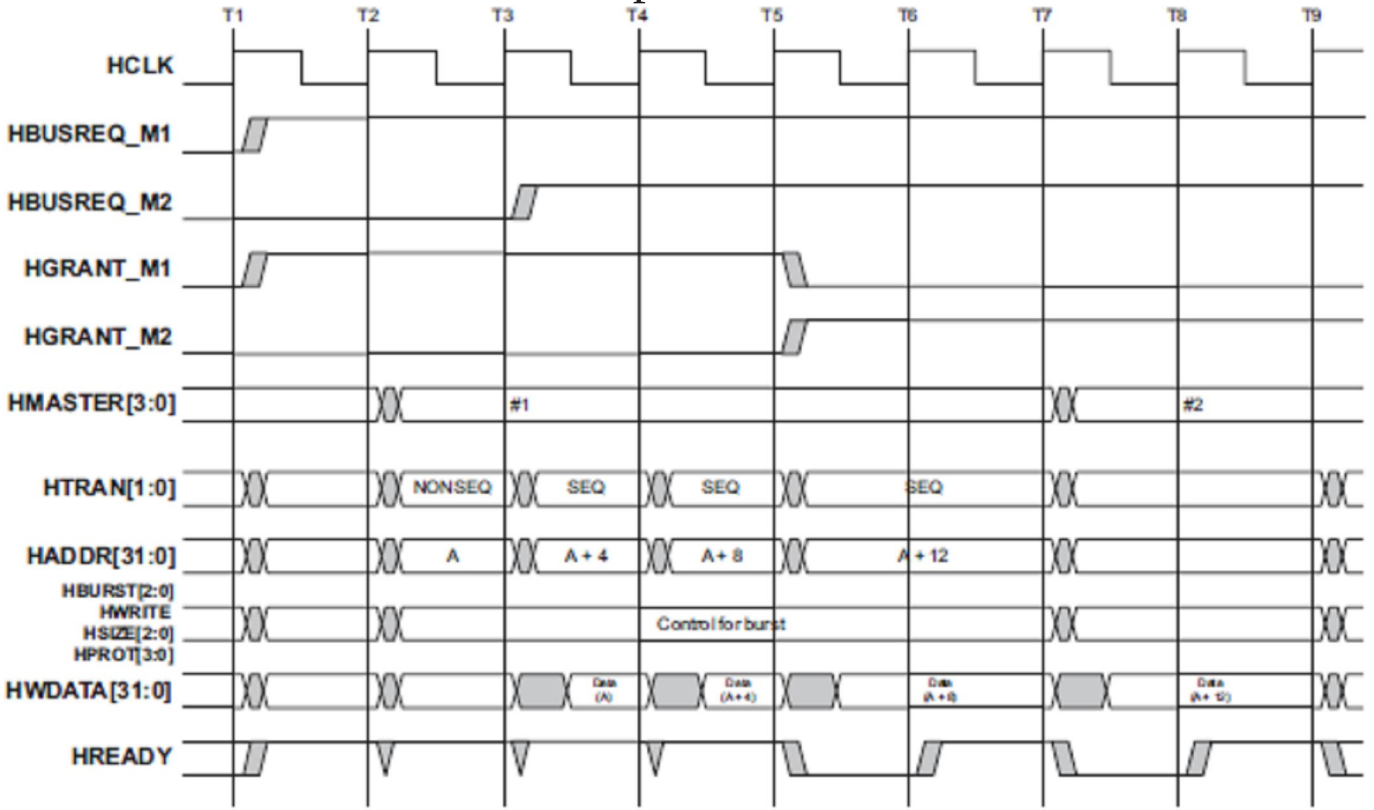
Granting Bus Access

- The arbiter indicates which bus master is currently the highest priority requesting the bus by asserting the appropriate **HGRANTx** signal.
- When the current transfer completes, as indicated by **HREADY** HIGH, then the master will become granted and the arbiter will change the **HMASTER[3:0]** signals to indicate the bus master number.



Bus Handover after Burst

- The arbiter changes the **HGRANTx** signals when the penultimate (one before last) address has been sampled.
- The new **HGRANTx** information will then be sampled at the same point as the last address of the burst is sampled.



Early Burst Termination

- The slave should keep monitoring the **HTRANS** signals.
- After the start of the burst every transfer should be labeled as SEQUENTIAL or BUSY.
- If a NONSEQUENTIAL or IDLE transfer occurs then this indicates that a new burst has started and therefore the previous one must have been terminated.
- If a bus master cannot complete a burst because it loses ownership of the bus then :
  - It must rebuild the burst appropriately when it next gains access to the bus.
  - For example, if a master has only completed one beat of a four-beat burst then it must use an undefined-length burst to perform the remaining three transfers.

Locked Transfers

- HLOCKx is used by master to indicate a locked transfer.
- The arbiter is then responsible for ensuring that no other bus masters are granted the bus until the locked sequence has completed.
- After a sequence of locked transfers the arbiter will always keep the bus master granted for an additional transfer to ensure that the last transfer in the locked sequence has completed successfully and has not received either a SPLIT or RETRY response.
- Therefore it is recommended, but not mandatory, that the master inserts an IDLE transfer after any locked sequence to provide an opportunity for the arbitration to change before commencing another burst of transfers.
- The arbiter is also responsible for asserting the **HMASTLOCK** signal. This signal indicates to any slave that the current transfer is locked and therefore must be processed before any other masters are granted the bus.

Default Master

- Every system must include a default bus master which is granted the bus if all other masters are unable to use the bus.
- When granted, the default bus master must only perform IDLE transfers.
- If no masters are requesting the bus then the arbiter may either grant the default master or alternatively it may grant the master that would benefit the most from having low access latency to the bus.
- Granting the default master access to the bus also provides a useful mechanism for ensuring that no new transfers are started on the bus and is a useful step to perform prior to entering a low-power mode of operation.
- The default master must be granted if all other masters are waiting for SPLIT transfers to complete.

Split Transfers

- SPLIT transfers improve the overall utilization of the bus by separating (or splitting) the operation of the master.
- When a transfer occurs the slave can decide to issue a SPLIT response if it believes the transfer will take a large number of cycles to perform.
- This signals to the arbiter that the master which is attempting the transfer should not be granted access to the bus until the slave indicates it is ready to complete the transfer.
- During every transfer the arbiter broadcasts a number, or tag, showing which master is using the bus using HMASTER[3:0].
- Therefore the arbiter is responsible for observing the response signals and internally masking any requests from masters which have been SPLIT.
- Later, when the slave can complete the transfer, it asserts the appropriate bit, according to the master number, on the **HSPLITx[15:0]** signals from the slave to the arbiter.
- Eventually the arbiter will grant the master so it can re-attempt the transfer. This may not occur immediately if a higher priority master is using the bus.
- When the transfer eventually takes place the slave finishes with an OKAY transfer response.

Split Transfer