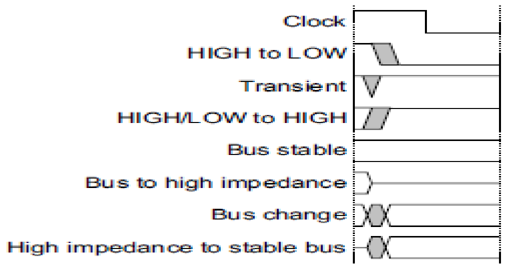
Dr. Amr Talaat
Eng. Sarah Milad
Faculty of IET

Advanced Microcontroller Bus Architecture (AMBA)



Key to timing diagram conventions

1.1 Protocol:

AMBA AHB-Lite addresses the requirements of high-performance synthesizable designs. It is a bus interface that supports a single bus master and provides high-bandwidth operation.
AHB-Lite implements the features required for high-performance, high clock frequency systems including:

• burst transfers
• single-clock edge operation
• non-tristate implementation
• wide data bus configurations, 64, 128, 256, 512, and 1024 bits.

Figure 1-1 shows a single master AHB-Lite system design with **one AHB-Lite master** and **three AHB-Lite slaves**. The bus interconnect logic **consists of one address decoder and a slave-to-master multiplexor**. The decoder **monitors the address from the master** so that the **appropriate slave is selected** and the multiplexor routes the corresponding slave output data back to the master

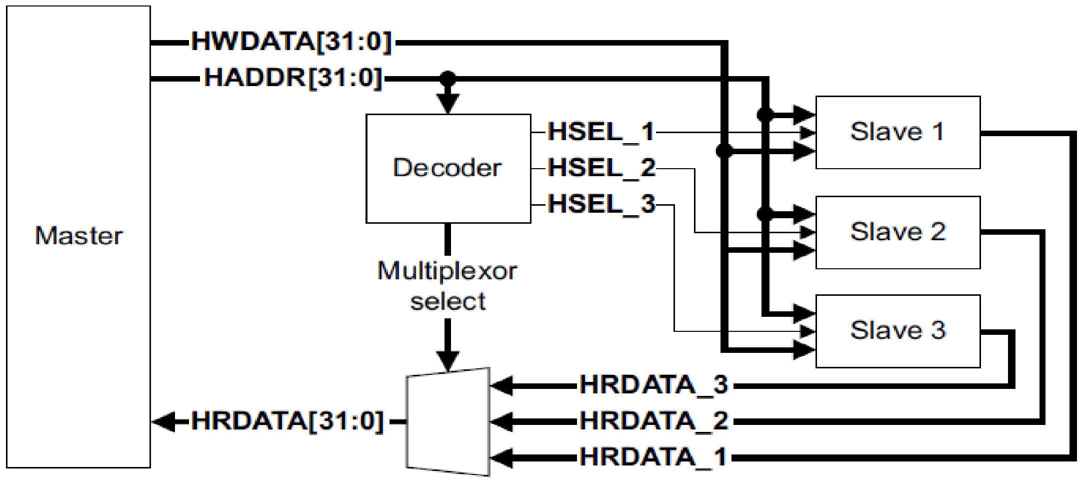


Figure 1-1 AHB-Lite block diagram

1.1.1 Master

An AHB-Lite master provides **address** and **control information** to initiate **read and write operations**. Figure 1-2 shows an AHB-Lite master interface.

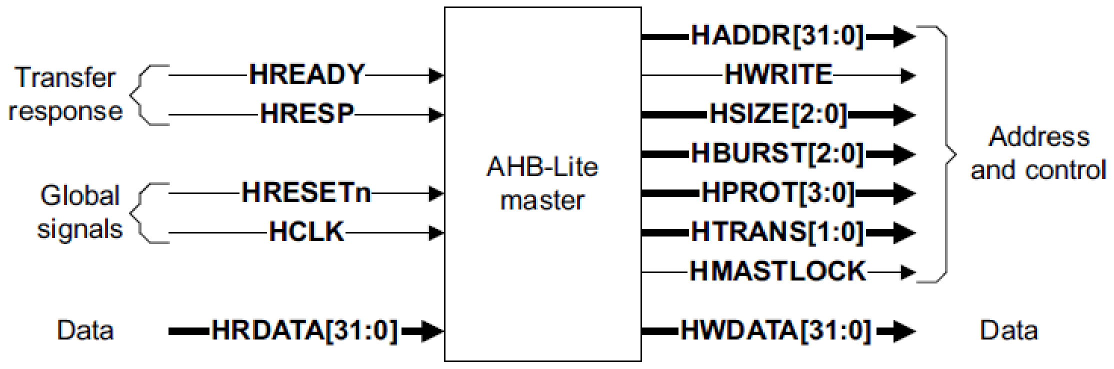


Figure 1-2 Master interface

1.1.2 Slave

An AHB-Lite slave responds to transfers initiated by masters in the system. The slave uses the **HSELx** select signal from the decoder to control when it responds to a bus transfer. The slave signals back to the master:

- the success
- failure
- or waiting of the data transfer.

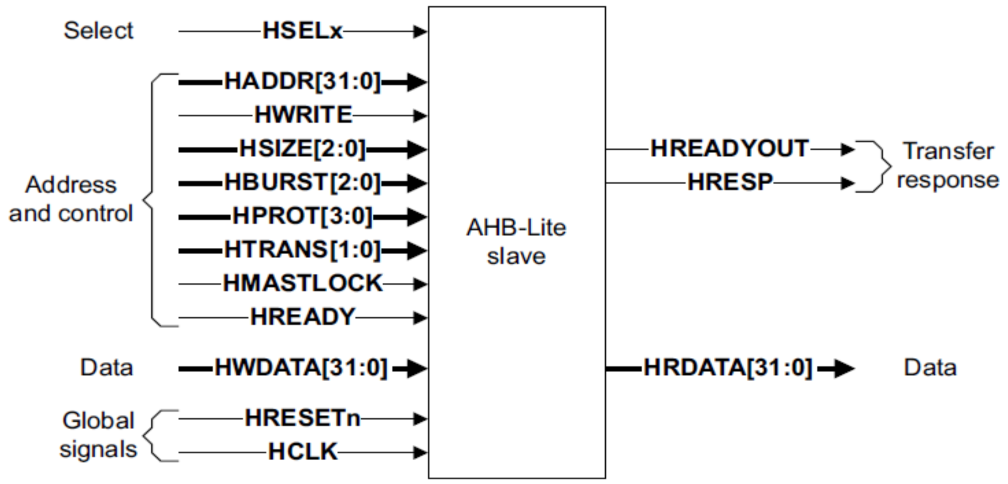


Figure 1-3 Slave interface

1.1.3 Decoder

This component decodes the address of each transfer and provides a select signal for the slave that is involved in the transfer. It also provides a control signal to the multiplexor. A single centralized decoder is required in all AHB-Lite implementations that use two or more slaves.

In multi-layer AHB-Lite implementations, the decoder function is usually included in the multi-layer interconnect component.

1.1.4 Multiplexer

A slave-to-master multiplexor is required to multiplex the read data bus and response signals from the slaves to the master. The decoder provides control for the multiplexor. A single centralized multiplexor is required in all AHB-Lite implementations that use two or more slaves.

In multi-layer AHB-Lite implementations, the multiplexor function is typically included in the multi-layer interconnect component.

1.2 Operation:

The master starts a transfer by **driving the address and control signals.** These signal provide information about **the address, direction, width of the transfer, and indicate if the transfer forms part of a burst**.
Transfers can be:

- **single**
- **incrementing bursts that do not wrap at address boundaries**
- **wrapping bursts that wrap at particular address boundaries.**

The write data bus moves data from the master to a slave, and the read data bus moves data from a slave to the master.

Every transfer consists of:
Address phase$\longrightarrow$one address and control cycle
Data phase $\longrightarrow$ one or more cycles for the data.
A slave cannot request that the address phase is extended and therefore all slaves must be capable of sampling the address during this time. However, a slave can request that the master extends the data phase by using **HREADY.** This signal, when LOW, causes wait states to be inserted into the transfer and enables the slave to have extra time to provide or sample data.
The slave uses HRESP to indicate the success or failure of a transfer.

1.3 Multi-layer AHB lite

Because AHB-Lite is a single master bus interface then if a multi-master system is required, the system designer must include a component that isolates all masters from each other. To achieve this isolation function, each master can be considered to be on its own layer, therefore the component must create a multi-layer interconnect where all masters are isolated from each other, but can share access to the slaves. Slave arbitration must be performed by the multi-layer interconnect component.

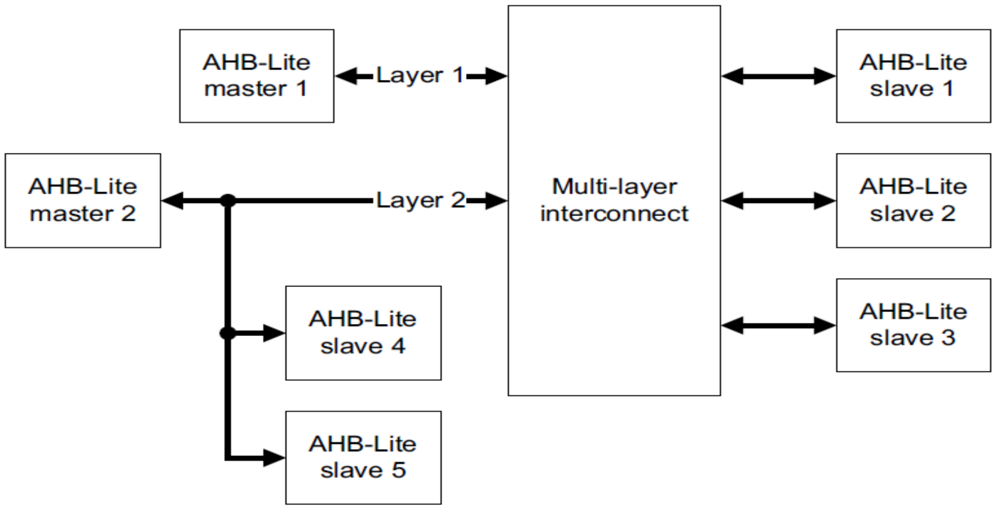


Figure 1-4 Example multi-layer AHB-Lite block diagram

In Figure 1-4, master 1 and master 2 each have access to slaves 1, 2, and 3. The multi-layer interconnect must prevent simultaneous access to a single slave by implementing an arbitration scheme for the three shared slaves. Master 1 does not require access to slaves 4 and 5, so these two slaves are kept local to master 2. This reduces the complexity of the multi-layer interconnect component.

2 Signal Description:

2.1 Global signals

Name	Source	Description
HCLK	Clock source	The bus clock times all bus transfers. All signal timings are related to the rising edge of **HCLK**.
HRESETn	Reset controller	The bus reset signal is active LOW and resets the system and the bus. This is the only active LOW AHB-Lite signal.

2.2 Master signals

Name	Destination	Description
HADDR[31:0]	Slave and decoder	The 32-bit system address bus.
HBURST[2:0]	Slave	The burst type indicates if the transfer is a single transfer or forms part of a burst. Fixed length bursts of 4, 8, and 16 beats are supported. The burst can be incrementing or wrapping. Incrementing bursts of undefined length are also supported.
HMASTLOCK	Slave	When HIGH, this signal indicates that the current transfer is part of a locked sequence. It has the same timing as the address and control signals.
HPROT[3:0]	Slave	The protection control signals provide additional information about a bus access and are primarily intended for use by any module that wants to implement some level of protection. The signals indicate if the transfer is an opcode fetch or data access, and if the transfer is a privileged mode access or user mode access. For masters with a memory management unit these signals also indicate whether the current access is cacheable or bufferable.
HSIZE[2:0]	Slave	Indicates the size of the transfer, that is typically byte, halfword, or word. The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
HTRANS[1:0]	Slave	Indicates the transfer type of the current transfer. This can be: • IDLE • BUSY • NONSEQUENTIAL • SEQUENTIAL.
HWDATA[31:0][a]	Slave	The write data bus transfers data from the master to the slaves during write operations. A minimum data bus width of 32 bits is recommended. However, this can be extended to enable higher bandwidth operation.
HWRITE	Slave	Indicates the transfer direction. When **HIGH** this signal indicates a write transfer and when **LOW** a read transfer. It has the same timing as the address signals, however, it must remain constant throughout a burst transfer.

2.3 Slave signals

Name	Destination	Description
HRDATA[31:0]	Multiplexor	During read operations, the read data bus transfers data from the selected slave to the multiplexor. The multiplexor then transfers the data to the master. A minimum data bus width of 32 bits is recommended. However, this can be extended to enable higher bandwidth operation.
HREADYOUT	Multiplexor	When HIGH, the **HREADYOUT** signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESP	Multiplexor	The transfer response, after passing through the multiplexor, provides the master with additional information on the status of a transfer. When LOW, the **HRESP** signal indicates that the transfer status is OKAY. When HIGH, the **HRESP** signal indicates that the transfer status is **ERROR.**

2.4 Decoder signals

Name	Destination	Description
HSELxa	Slave	Each AHB-Lite slave has its own slave select signal **HSELx** and this signal indicates that the current transfer is intended for the selected slave. When the slave is initially selected, it must also monitor the status of **HREADY** to ensure that the previous bus transfer has completed, before it responds to the current transfer. The **HSELx** signal is a combinatorial decode of the address bus.

Note:

The letter x used in **HSELx** must be changed to a unique identifier for each AHB-Lite slave in a system. For example, **HSEL_S1**, **HSEL_S2**, and **HSEL_Memory**.

Usually the decoder also provides the multiplexor with the **HSELx** signals, or a signal/bus derived from the **HSELx** signals, to enable the multiplexor to route the appropriate signals, from the selected slave to the master. It is important that these additional multiplexor control signals are retimed to the data phase.

2.5 Multiplexor Signals

Name	Destination	Description
HRDATA[31:0]	Master	Read data bus, selected by the decoder. Because the **HRDATA[31:0]** and **HRESP** signals pass through the multiplexor and retain the same signal naming.
HREADY	Master and slave	When HIGH, the **HREADY** signal indicates to the master and all slaves, that the previous transfer is complete.
HRESP	Master	Transfer response, selected by the decoder (same reason)

Dr. Amr Talaat
Eng. Sarah Milad
Faculty of IET

3 Transfer

3.1 Basic Transfer

Address: Lasts for a single **HCLK** cycle unless its extended by the previous bus transfer.
Data: That might require several **HCLK** cycles. Use the **HREADY** signal to control the number of clock cycles required to complete the transfer.
HWRITE controls the direction of data transfer to or from the master. Therefore, when:
HWRITE is **HIGH**, it indicates a write transfer and the master broadcasts data on the write data bus, **HWDATA[31:0]**
HWRITE is **LOW**, a read transfer is performed and the slave must generate the data on the read data bus, **HRDATA[31:0]**.

The simplest transfer is one with no wait states, so the transfer consists of one address cycle and one data cycle. Figure 3-1 shows a simple read transfer and Figure 3-2 shows a simple write transfer.

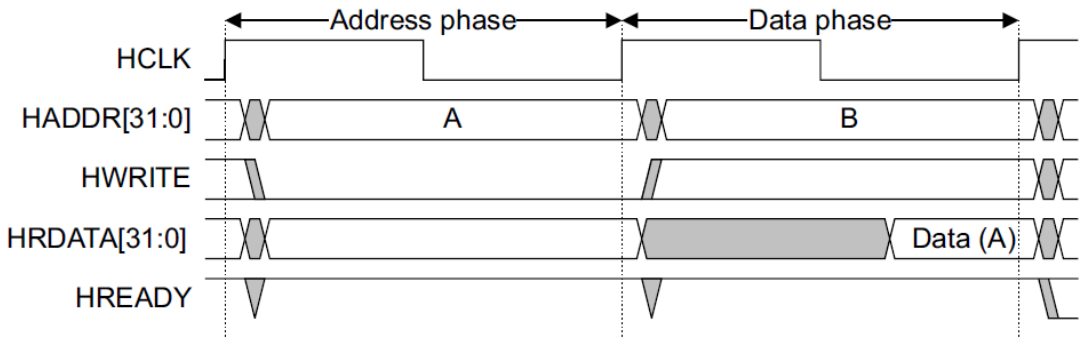


Figure 3-1 Read transfer

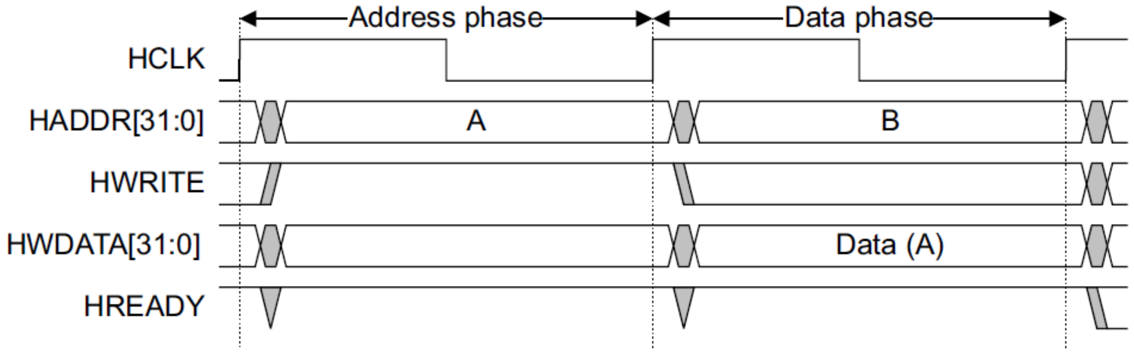


Figure 3-2 Write transfer

In a simple transfer with no wait states:
1. The master drives the address and control signals onto the bus after the rising edge of **HCLK**.
2. The slave then samples the address and control information on the next rising edge of **HCLK**.
3. After the slave has sampled the address and control it can start to drive the appropriate **HREADY** response. This response is sampled by the master on the third rising edge of **HCLK**.

This simple example demonstrates how the address and data phases of the transfer occur during different clock cycles. The address phase of any transfer occurs during the data phase of the previous transfer. This overlapping of address and data is fundamental to the pipelined nature of the bus and enables high performance operation while still providing adequate time for a slave to provide the response to a transfer.
A slave can insert wait states into any transfer to enable additional time for completion.

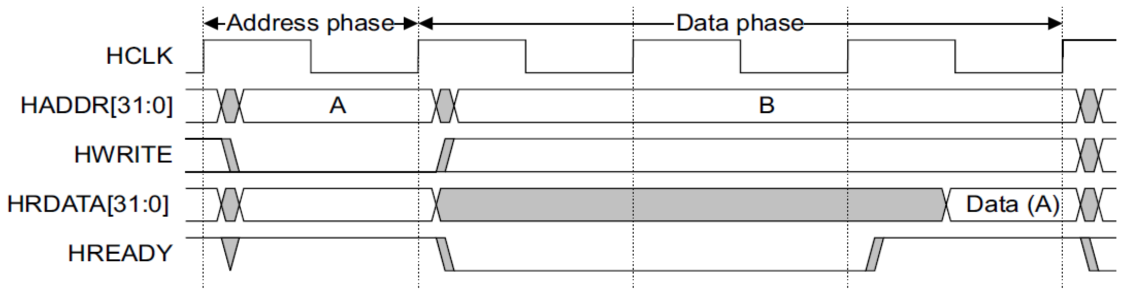


Figure 3-3 Read transfer with wait states

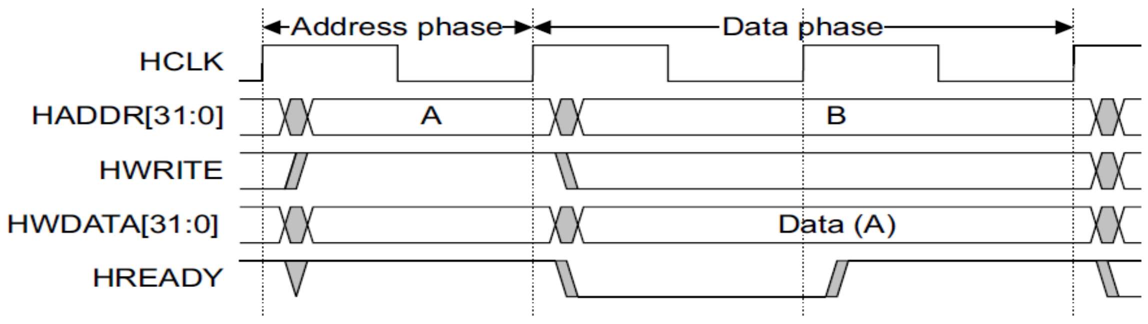


Figure 3-4 Write transfer with wait state

For **write operations** the master holds the data stable throughout the extended cycles.
For **read transfers** the slave does not have to provide valid data until the transfer is about to complete.

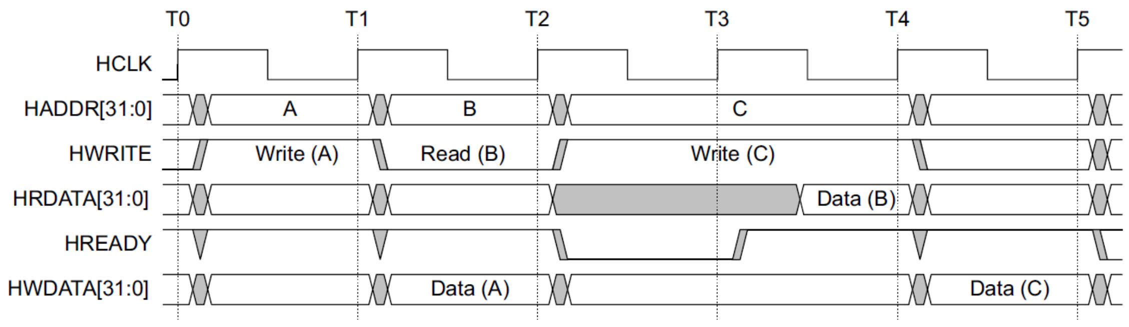


Figure 3-5 Multiple transfers

In Figure 3-5:

- The transfers to addresses A and C are zero wait state
- 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

3.2 Transfer Types

HTRANS[1:0]	Type	Description
b00	IDLE	Indicates that no data transfer is required. A master uses an IDLE transfer when it does not want to perform a data transfer. It is recommended that the master terminates a locked transfer with an IDLE transfer. Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer must be ignored by the slave.
b01	BUSY	The BUSY transfer type enables masters to insert idle cycles in the middle of a burst. This transfer type indicates that the master is continuing with a burst 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. Only undefined length bursts can have a BUSY transfer as the last cycle of a burst. Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer must be ignored by the slave.
b10	NONSEQ	Indicates a single transfer or the first transfer of a burst. The address and control signals are unrelated to the previous transfer. Single transfers on the bus are treated as bursts of length one and therefore the transfer type is NONSEQUENTIAL.
b11	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 transfer size, in bytes, with the transfer size being signaled by the **HSIZE[2:0]** signals. **In the case of a wrapping burst the address of the transfer wraps at the address boundary.**

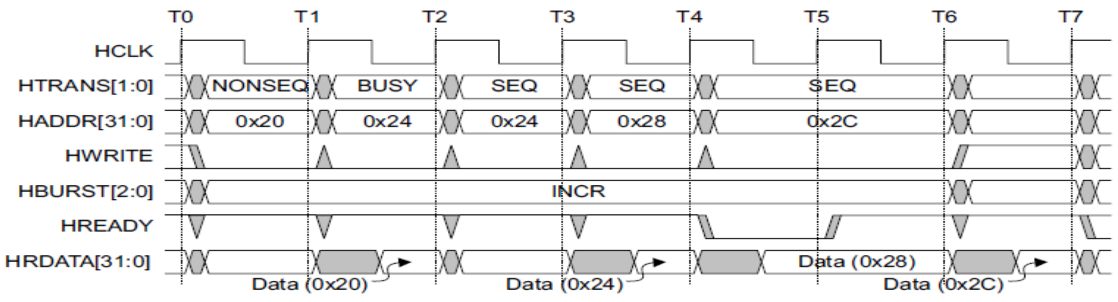


Figure 3-6 Transfer type examples

In Figure 3-6:

T0-T1	The 4-beat read starts with a NONSEQ transfer.
T1-T2	The master is unable to perform the second beat and inserts a BUSY transfer to delay the start of the second beat. The slave provides the read data for the first beat.
T2-T3	The master is now ready to start the second beat, so a SEQ transfer is signaled. The master ignores any data that the slave provides on the read data bus.
T3-T4	The master performs the third beat. The slave provides the read data for the second beat.
T4-T5	The master performs the last beat. The slave is unable to complete the transfer and uses **HREADY** to insert a single wait state.
T5-T6	The slave provides the read data for the third beat.
T6-T7	The slave provides the read data for the last beat.

3.3 Locked Transfer

If the master requires locked accesses then it must also assert the **HMASTLOCK** signal. This signal indicates to any slave that the current transfer sequence is indivisible and must therefore be processed before any other transactions are processed.
Figure 3-7 shows the **HMASTLOCK** signal with a microprocessor SWP instruction.

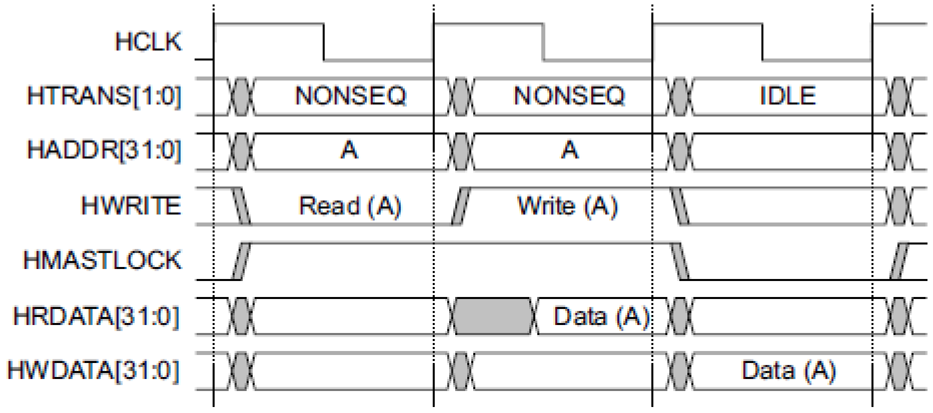


Figure 3-7 Locked transfer

Note:
After a locked transfer, it is recommended that the master inserts an IDLE transfer.
Most slaves have no requirement to implement **HMASTLOCK** because they are only capable of performing transfers the order they are received. Slaves that can be accessed by more than one master, for example, a Multi Port Memory Controller(MPMC) must implement the **HMASTLOCK** signal.

3.4 Transfer Size

HSIZE[2:0] indicates the size of a data transfer. Table 3-2 lists the possible transfer sizes.

Table 3-2 Transfer size encoding

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

Note
The transfer size set by **HSIZE** must be less than or equal to the width of the data bus. For example, with a 32 bit data bus, **HSIZE** must only use the values b000, b001, or b010.

Use **HSIZE** in conjunction with **HBURST**, to determine the address boundary for wrapping bursts.
The **HSIZE** signals have exactly the same timing as the address bus. However, they must remain constant throughout a burst transfer.

3.5 Burst Operation

Bursts of 4, 8, and 16-beats, undefined length bursts, and single transfers are defined in this protocol. It supports **incrementing** and **wrapping bursts.**

- Incrementing bursts access sequential locations and the address of each transfer in the burst is an increment of the previous address.
- Wrapping bursts wrap when they cross an address boundary. The address boundary is calculated as the product of the numbers of beats in a burst and the size of the transfer. The number of beats are controlled by **HBURST** and the transfer size is controlled by **HSIZE**.

For example, a four-beat wrapping burst of word (4-byte) accesses wraps at 16-byte boundaries. Therefore, if the start address of the transfer is 0x34, then it consists of four transfers to addresses 0x34, 0x38, 0x3C, and 0x30.

HBURST[2:0] controls the burst type. Table 3-3 lists the possible burst types.

Table 3-3 Burst signal encoding

HBURST[2:0]	Type	Description
b000	SINGLE	Single burst
b001	INCR	Incrementing burst of undefined length
b010	WRAP4	4-beat wrapping burst
b011	INCR4	4-beat incrementing burst
b100	WRAP8	8-beat wrapping burst
b101	INCR8	8-beat incrementing burst
b110	WRAP16	16-beat wrapping burst
b111	INCR16	16-beat incrementing burst

Masters must not attempt to start an incrementing burst that crosses a 1KB address boundary.
Masters can perform single transfers using either:

- SINGLE burst
- Undefined length burst that has a burst of length one

Note

- The **burst size** indicates the **number of beats in the burst** and **not the number of bytes transferred**. Calculate the total amount of data transferred in a burst by **multiplying the number of beats** by the **amount of data in each beat**, as indicated by **HSIZE[2:0]**.
- All transfers in a burst must be aligned to the address boundary equal to the size of the transfer.

3.5.1 Burst Termination after a BUSY transfer

After a burst has started, the master uses **BUSY** transfers if it requires more time before continuing with the next transfer in the burst.
During an **undefined length burst, INCR**, the master might insert **BUSY** transfers and then decide that no more data transfers are required. Under these circumstances, it is acceptable for the master to then perform a **NONSEQ or IDLE transfer** that then effectively terminates the undefined length burst.

The protocol **does not permit** a **master** to end a burst with a **BUSY** transfer for **fixed length** bursts of type:

- incrementing INCR4, INCR8, and INCR16
- Wrapping WRAP4, WRAP8, and WRAP16.

These fixed length burst types **must terminate** with a **SEQ transfer.**
The master is **not permitted** to perform a **BUSY** transfer immediately after **a SINGLE burst. SINGLE** bursts must be followed by an **IDLE** transfer or a **NONSEQ** transfer.

3.5.2 Early Burst Termination

Bursts can be terminated by either:

- Slave error response
- Multi-layer interconnect termination

Slave Error Response:

- If a slave provides an **ERROR** response then the **master can 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.
- If the **master does not complete** that burst then there is no requirement for it **to rebuild the burst when it next accesses that slave**. For example**,** if a master only completes three beats of an eight-beat burst then it does not have to complete the remaining five transfers when it next accesses that slave.

Multi- layer interconnect termination:

- Although masters are not permitted to terminate a burst request early, **slaves must be designed to work correctly if the burst is not completed**.
- When a multi-layer interconnect component is used in a multi-master system then it can terminate a burst so **that another master can gain access to the slave.** The slave must terminate the burst from the original master and then respond appropriately to the new master if this occurs.

3.5.3 Burst Examples

- Four-beat wrapping burst, WRAP4
- Four-beat incrementing burst, INCR4
- Eight-beat wrapping burst, WRAP8
- Eight-beat incrementing burst, INCR8
- Undefined length bursts, INCR

Four-beat wrapping burst, WRAP4

Figure 3-8 shows a write transfer using a four-beat wrapping burst, with a wait state added for the first transfer.

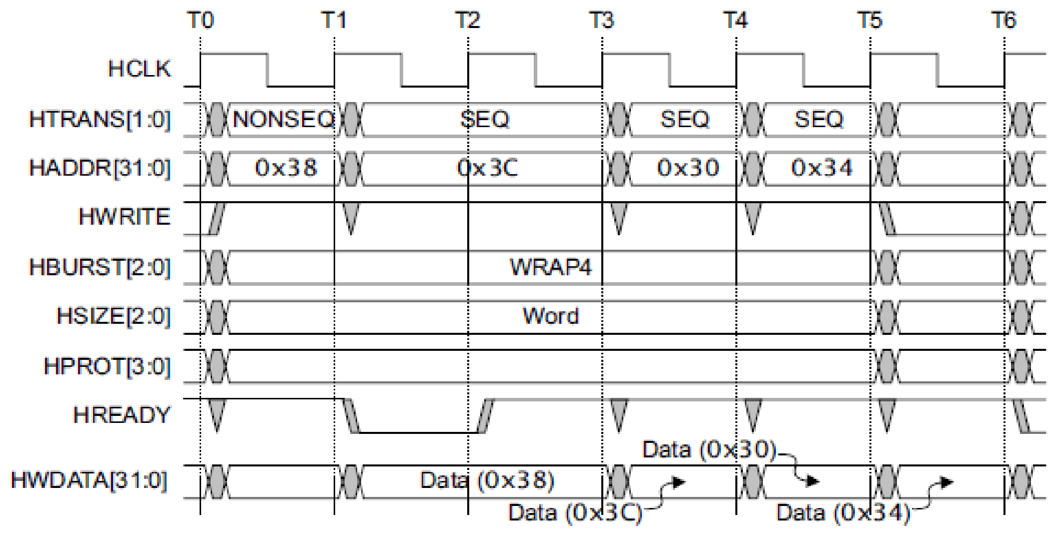


Figure 3-8 Four-beat wrapping burst

Because the burst is **a four-beat burst** of word transfers, the address wraps at **16-byte boundaries**, and the transfer to address **0x3C is followed by a transfer to address 0x30.**

Four-beat incrementing burst, INCR

Figure 3-9 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the **address does not wrap at a 16-byte** boundary and the **address 0x3C** is followed by a transfer to address **0x40**.

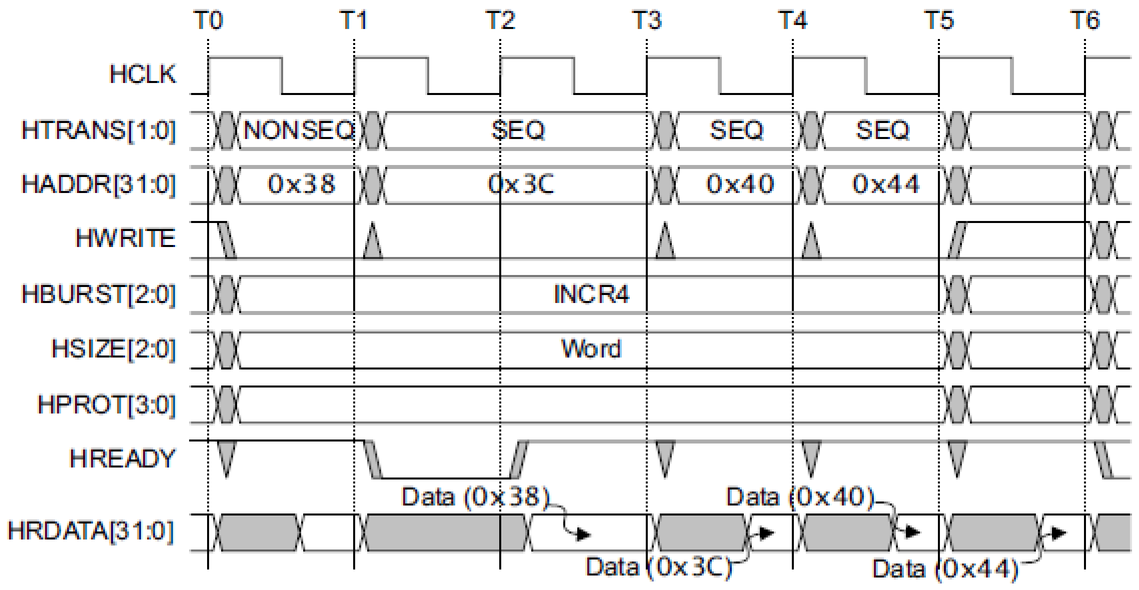


Figure 3-9 Four-beat incrementing burst

Eight-beat wrapping burst, WRAP8

Figure 3-10 shows a read transfer using an eight-beat wrapping burst.

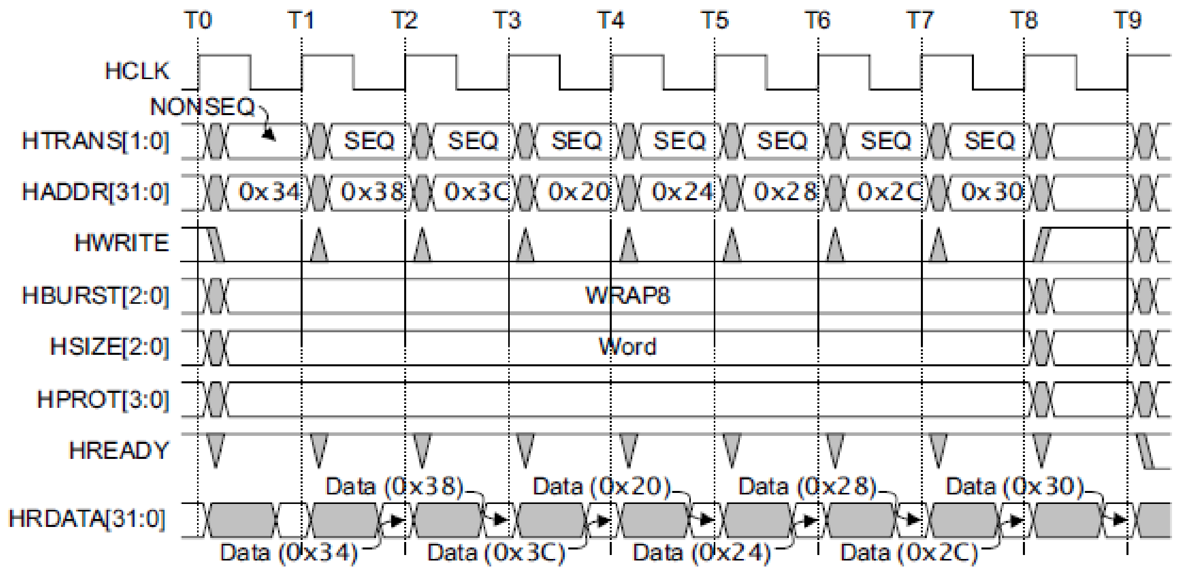


Figure 3-10 Eight-beat wrapping burst

Because the burst is an eight-beat burst of word transfers, the address wraps at 32-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x20. (Because the rule is the low 5 bits of address can changes since it is 32 byte boundaries)

Eight-beat incrementing burst, INCR8

Figure 3-11 shows a write transfer using an eight-beat incrementing burst.

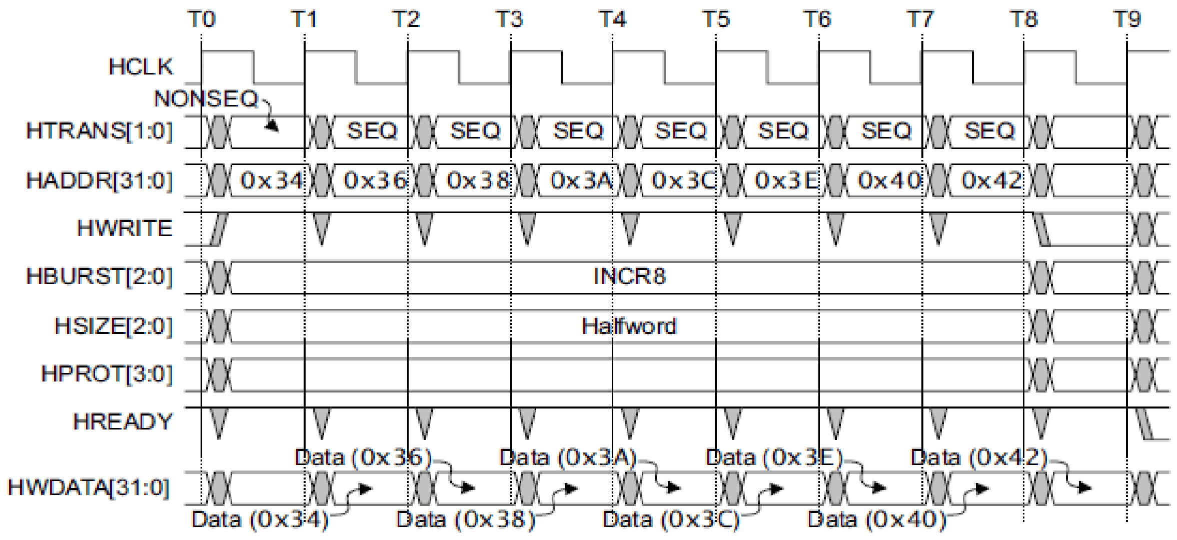


Figure 3-11 Eight-beat incrementing burst

This burst uses **half word** transfers, therefore the addresses increase by two. Because the burst is incrementing, the addresses continue to increment beyond the 16-byte address boundary.

Dr. Amr Talaat
Eng. Sarah Milad
Faculty of IET

Undefined length bursts, INCR
Figure 3-12 shows incrementing bursts of undefined length.

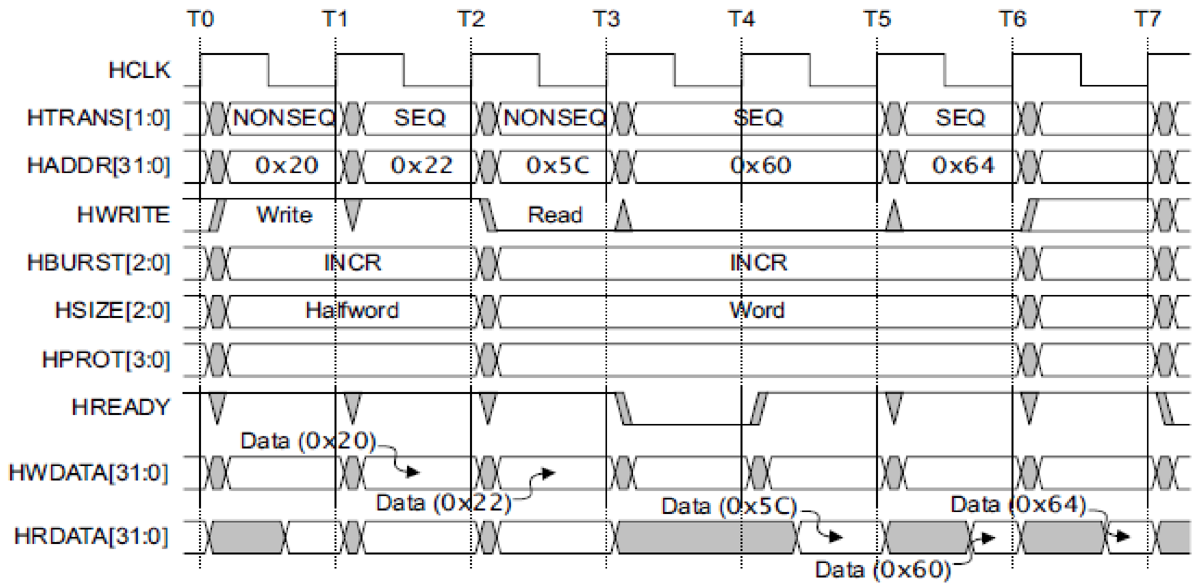


Figure 3-12 Undefined length bursts

Figure 3-12 shows two bursts:

- The first burst is a write consisting of two half word transfers starting at address 0x20. These transfer addresses increment by two.
- The second burst is a read consisting of three word transfers starting at address0x5C. These transfer addresses increment by four.

3.6 Waited Transfers

Slaves use **HREADY** to insert wait states if they require more time to provide or sample the data. During a waited transfer, the master is restricted to what changes it can make to the transfer type and address. These restrictions are described in the following sections:

- Transfer type changes during wait states
- Address changes during wait states

3.6.1 Transfer type changes during wait states

When the slave is requesting wait states, the master must not change the transfer type, **except:**

- ✓ IDLE transfer
- ✓ BUSY transfer, fixed length burst
- ✓ BUSY transfer, undefined length burst

Reference:

2006 ARM “AMBA 3 AHB-Lite Protocol”