# AMBA Bus Specification

Reference

AMBA™ Specification (Rev 2.0) by ARM - http://www.arm.com

# Overview of the AMBA specification

The *Advanced Microcontroller Bus Architecture* (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers.

Three distinct buses are defined within the AMBA specification:

- the *Advanced High-performance Bus* (AHB)
- the *Advanced System Bus* (ASB)
- the *Advanced Peripheral Bus* (APB).

A test methodology is included with the AMBA specification which provides an infrastructure for modular macrocell test and diagnostic access.

## Advanced High-performance Bus (AHB)

The AMBA AHB is for high-performance, high clock frequency system modules.

The AHB acts as the high-performance system *backbone* bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques.

## Advanced System Bus (ASB)

The AMBA ASB is for high-performance system modules.

AMBA ASB is an alternative system bus suitable for use where the high-performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions.

## Advanced Peripheral Bus (APB)

The AMBA APB is for low-power peripherals.

AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.

## Terminology

**Bus cycle**

A bus cycle is a basic unit of one bus clock period and for the purpose of AMBA AHB or APB protocol descriptions is defined from rising-edge to rising-edge transitions. An ASB bus cycle is defined from falling-edge to falling-edge transitions. Bus signal timing is referenced to the bus cycle clock.

**Bus transfer**

An AMBA ASB or AHB bus transfer is a read or write operation of a data object, which may take one or more bus cycles. The bus transfer is terminated by a *completion* response from the addressed slave.

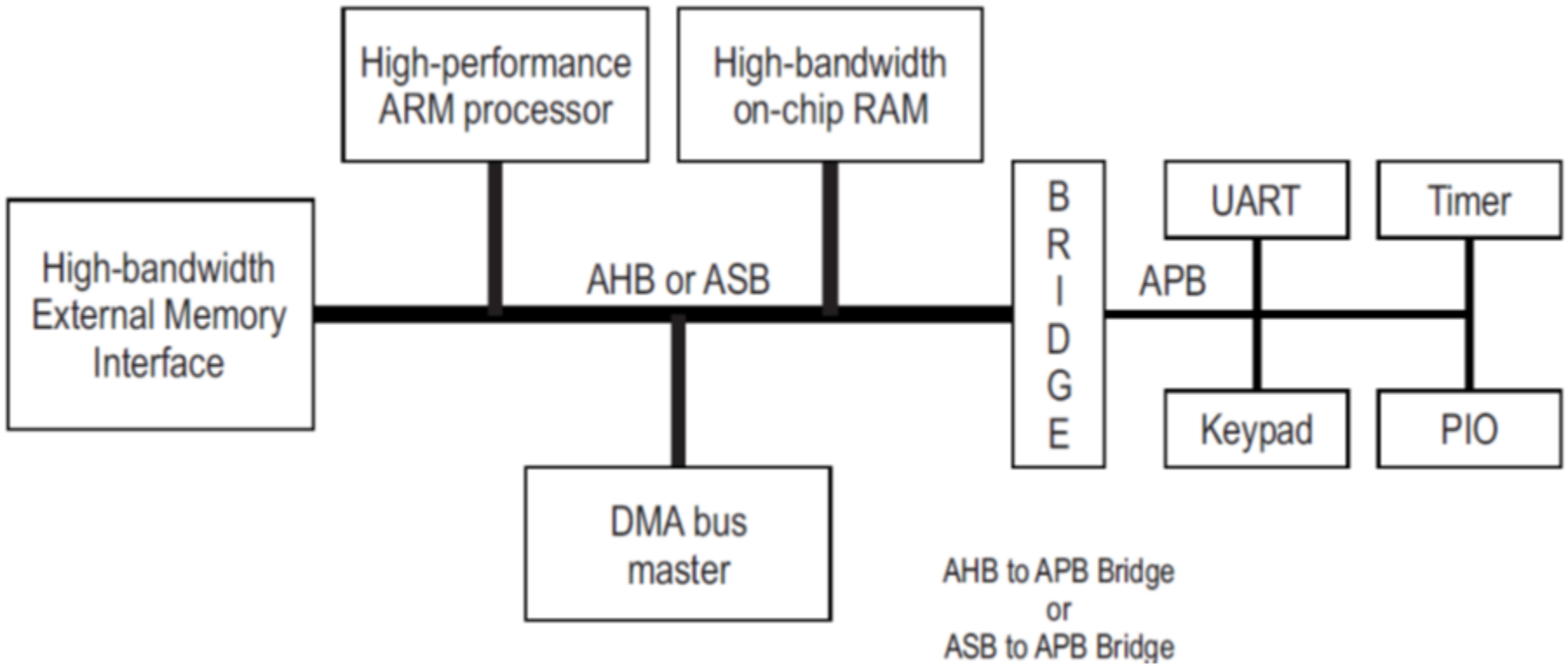
The transfer sizes supported by AMBA ASB include byte (8-bit), halfword (16-bit) and word (32-bit). AMBA AHB additionally supports wider data transfers, including 64-bit and 128-bit transfers. An AMBA APB bus transfer is a read or write operation of a data object, which always requires two bus cycles.

**Burst operation**

A burst operation is defined as one or more data transactions, initiated by a bus master, which have a consistent width of transaction to an incremental region of address space. The increment step per transaction is determined by the width of transfer (byte, halfword, word). No burst operation is supported on the APB.

# A typical AMBA-based microcontroller

An AMBA-based microcontroller typically consists of a high-performance system *backbone* bus (AMBA AHB or AMBA ASB), able to sustain the external memory bandwidth, on which the CPU, on-chip memory and other *Direct Memory Access* (DMA) devices reside. This bus provides a high-bandwidth interface between the elements that are involved in the majority of transfers. Also located on the high-performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located



## When to use AMBA AHB/ASB or APB

A full AHB or ASB interface is used for:

- bus masters
- on-chip memory blocks
- external memory interfaces
- high-bandwidth peripherals with FIFO interfaces
- DMA slave peripherals.

A simple APB interface is recommended for:

- simple register-mapped slave devices
- very low power interfaces where clocks cannot be globally routed
- grouping narrow-bus peripherals to avoid loading the system bus.

## Introducing the AMBA AHB

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation.

AMBA AHB implements the features required for high-performance, high clock frequency systems including:

- burst transfers
- split transactions
- single-cycle bus master handover
- single-clock edge operation
- non-tristate implementation
- wider data bus configurations (64/128 bits).

A typical AMBA AHB system design contains the following components:

**AHB master**

A bus master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

**AHB slave**

A bus slave responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

**AHB arbiter**

The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as *highest priority* or *fair* access can be implemented depending on the application requirements.

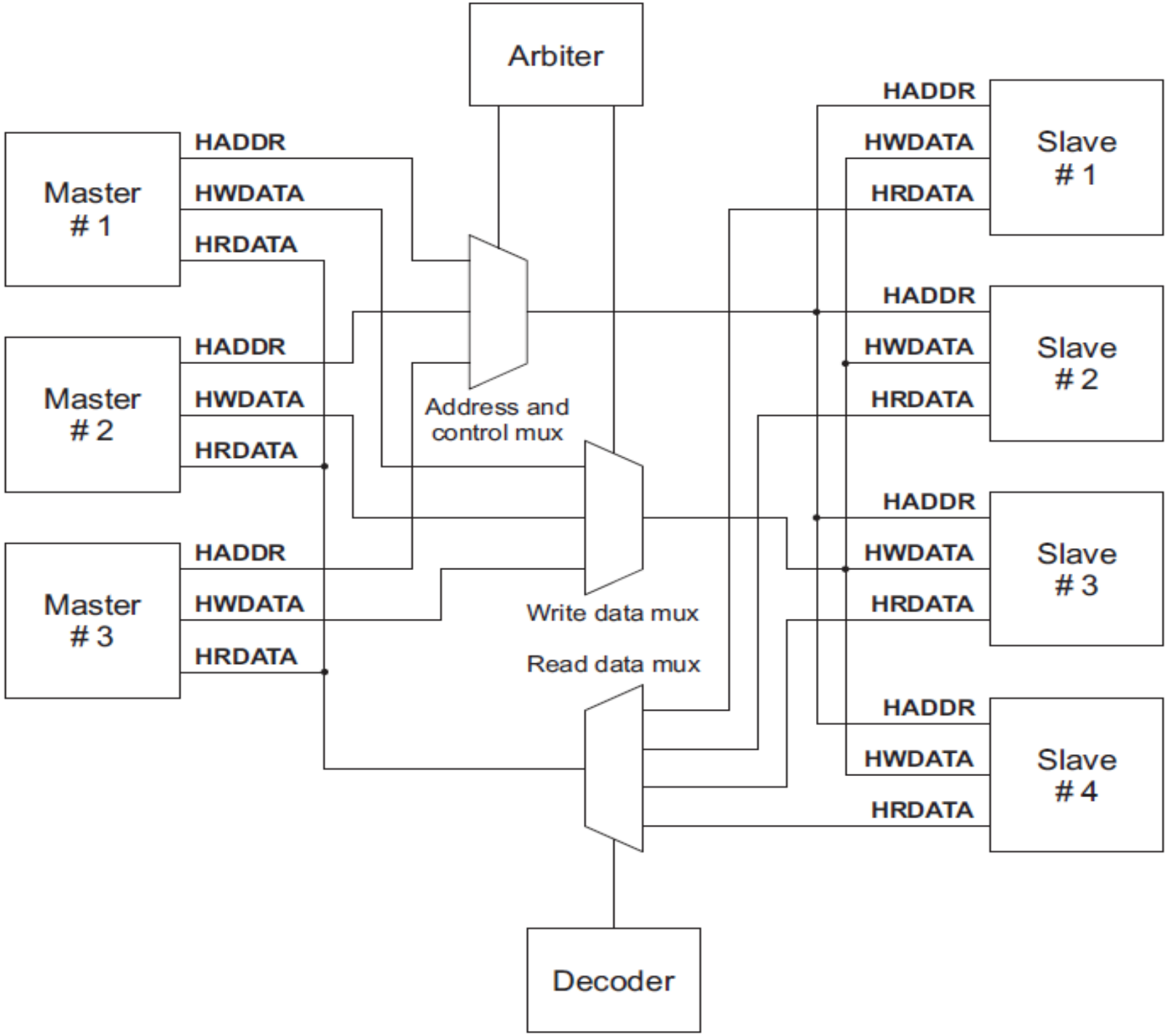
An AHB would include only one arbiter, although this would be trivial in single bus master systems.

**AHB decoder**

The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer.

A single centralized decoder is required in all AHB implementations.

## Bus interconnection



# AMBA AHB operation

Before an AMBA AHB transfer can commence the bus master must be granted access to the bus. This process is started by the master asserting a request signal to the arbiter. Then the arbiter indicates when the master will be granted use of the bus.

A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provide information on the address, direction and width of the transfer, as well as an indication if the transfer forms part of a burst. Two different forms of burst transfers are allowed:

- incrementing bursts, which do not wrap at address boundaries
- wrapping bursts, which wrap at particular address boundaries.

A write data bus is used to move data from the master to a slave, while a read data bus is used to move data from a slave to the master.

Every transfer consists of:

- an address and control cycle
- one or more cycles for the data.

The address cannot be extended and therefore all slaves must sample the address during this time. The data, however, can be extended using the **HREADY** signal. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for the slave to provide or sample data.

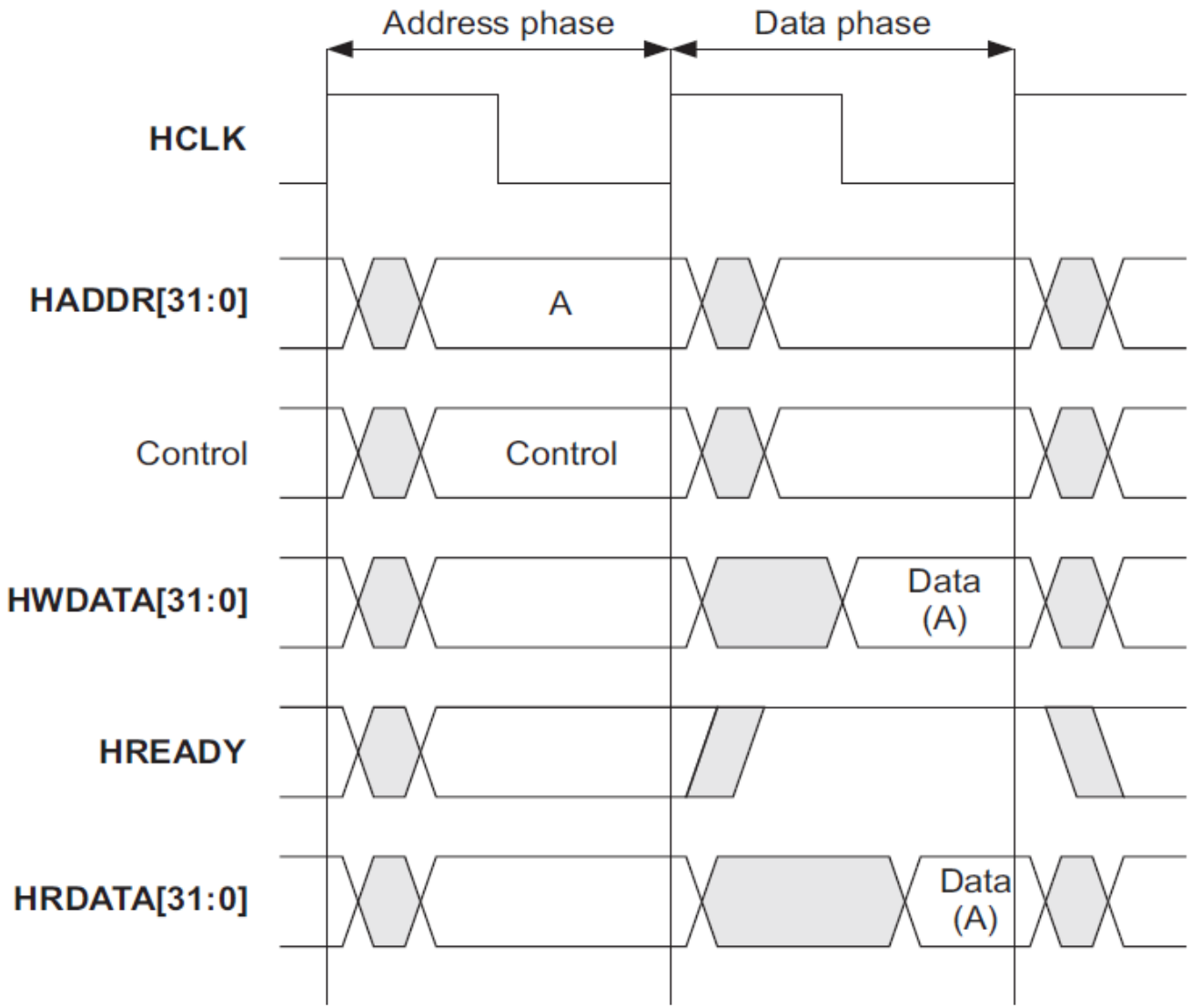
## AMBA AHB operation

During a transfer the slave shows the status using the response signals, **HRESP[1:0]**:

| | |
|---|---|
| OKAY | The OKAY response is used to indicate that the transfer is progressing normally and when **HREADY** goes HIGH this shows the transfer has completed successfully. |
| ERROR | The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful. |
| RETRY and SPLIT | Both the RETRY and SPLIT transfer responses indicate that the transfer cannot complete immediately, but the bus master should continue to attempt the transfer. |

# AMBA AHB operation

## Basic transfer

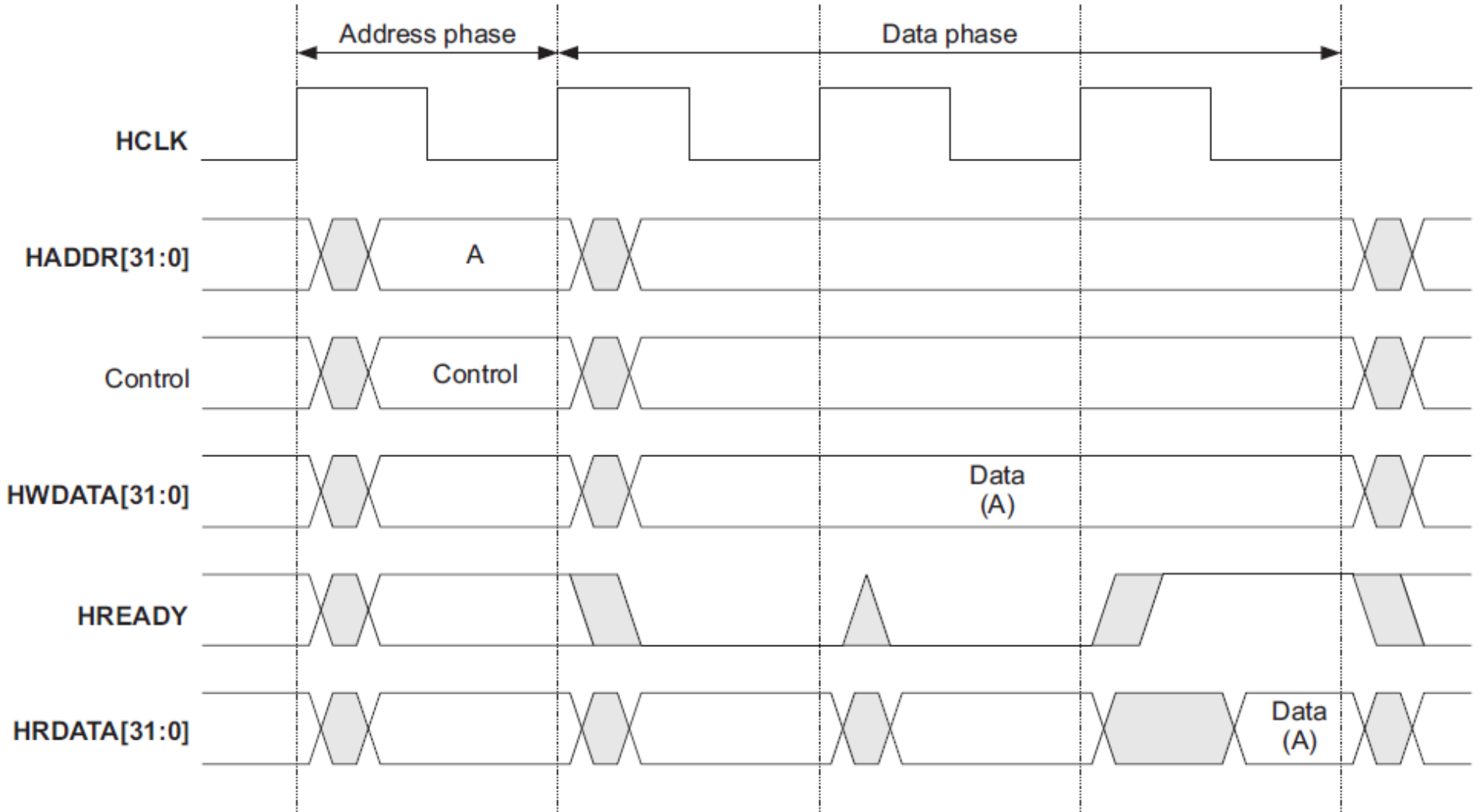



- **HREADY -** HIGH indicates that a transfer has finished on the bus.

**Simple transfer**

# AMBA AHB operation

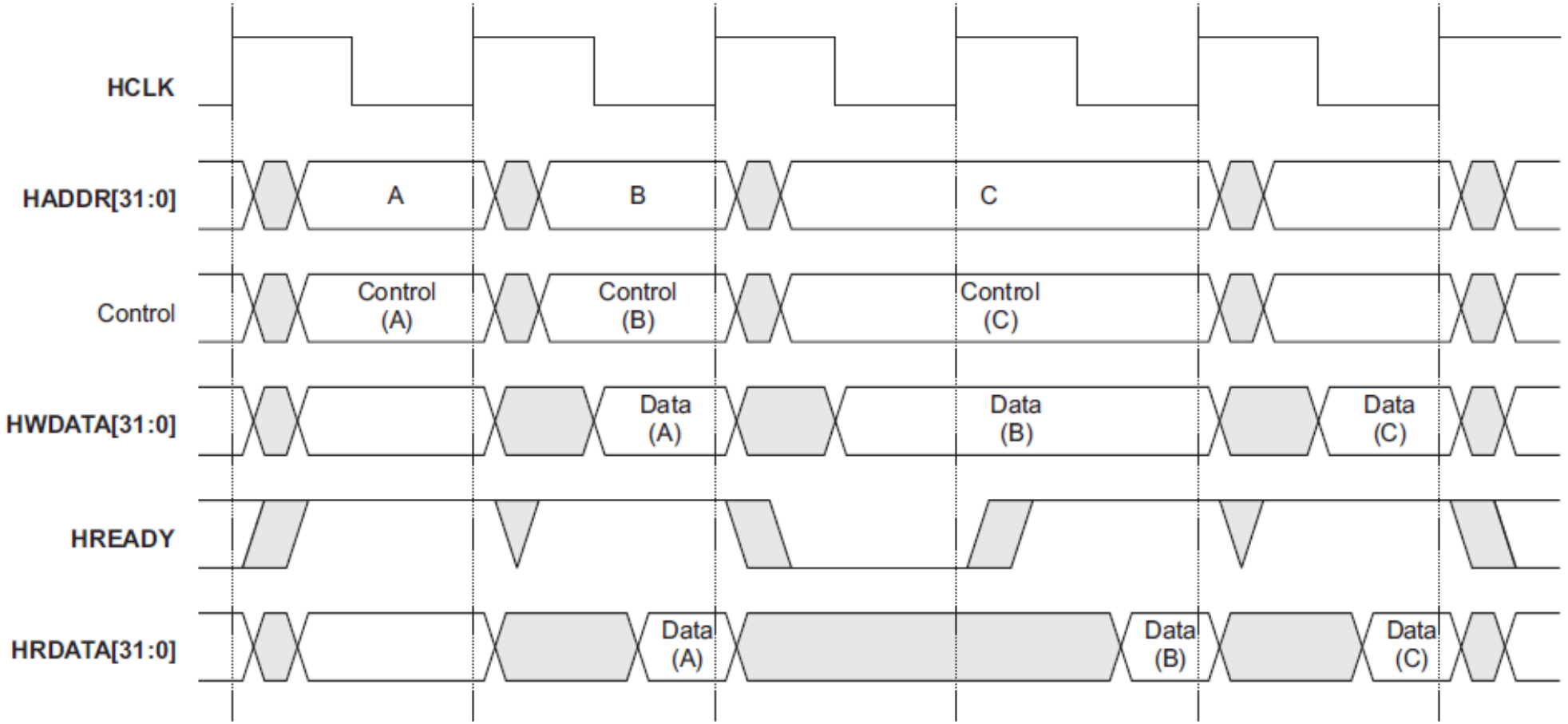
## Basic transfer




Transfer with wait states

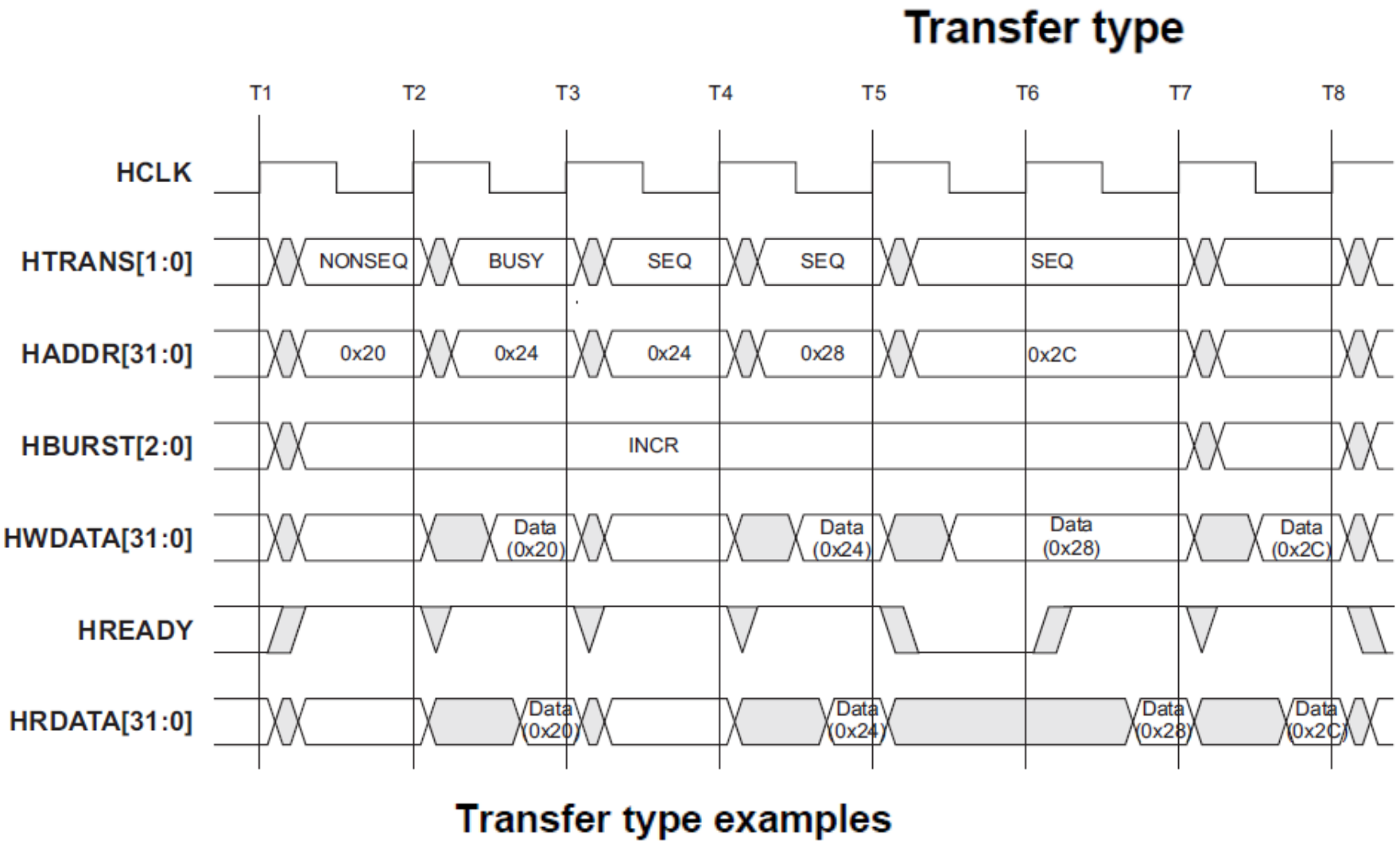
# AMBA AHB operation

## Basic transfer




## Multiple transfers

# AMBA AHB operation




Transfer type examples

- NONSEQ - indicates the first transfer of a burst or a single transfer. The address and control signals are unrelated to the previous transfer

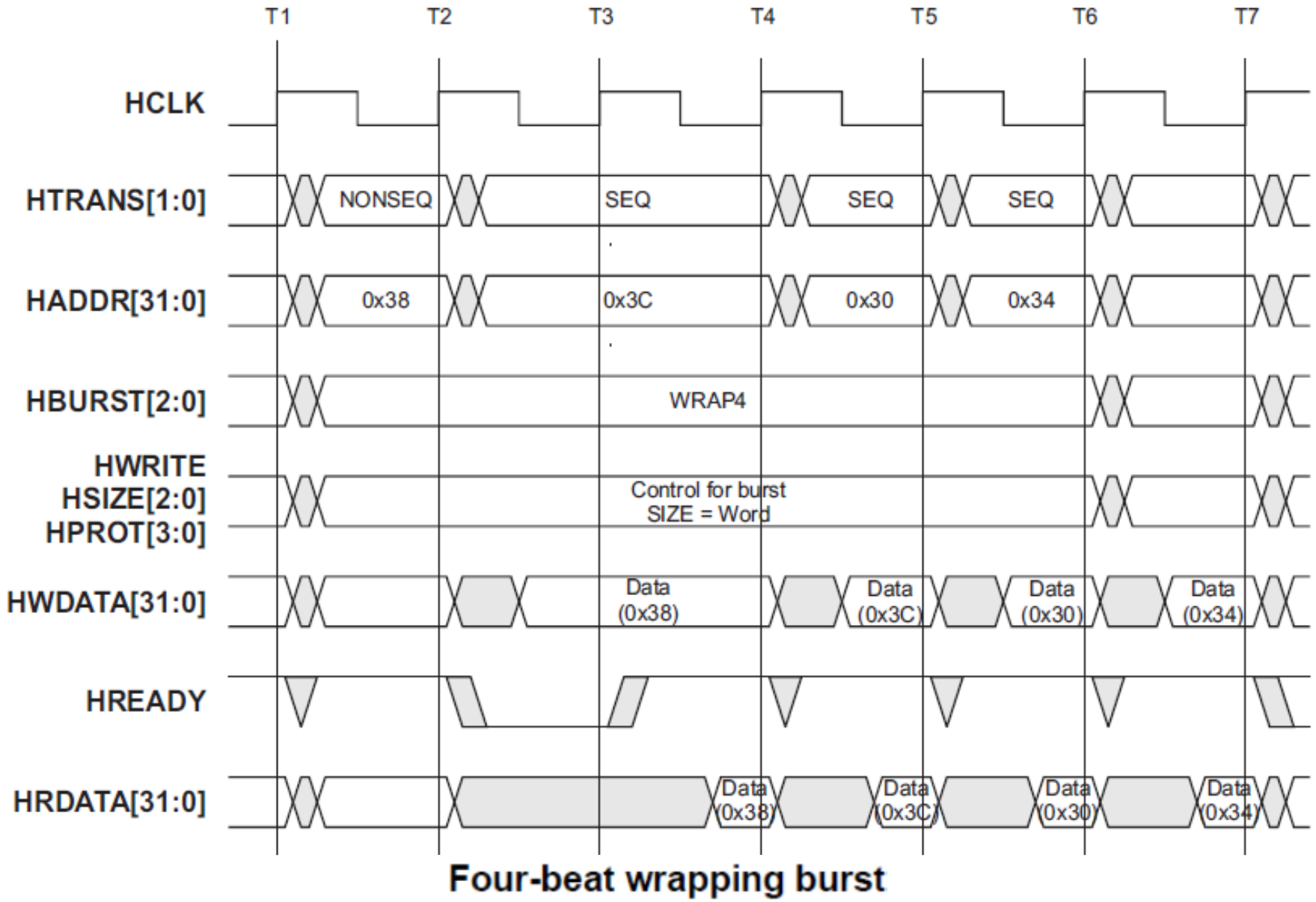
- SEQ - the remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer. The control information is identical to the previous transfer.

- IDLE- indicates that no data transfer is required

- 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.

# AMBA AHB operation

## Burst operation

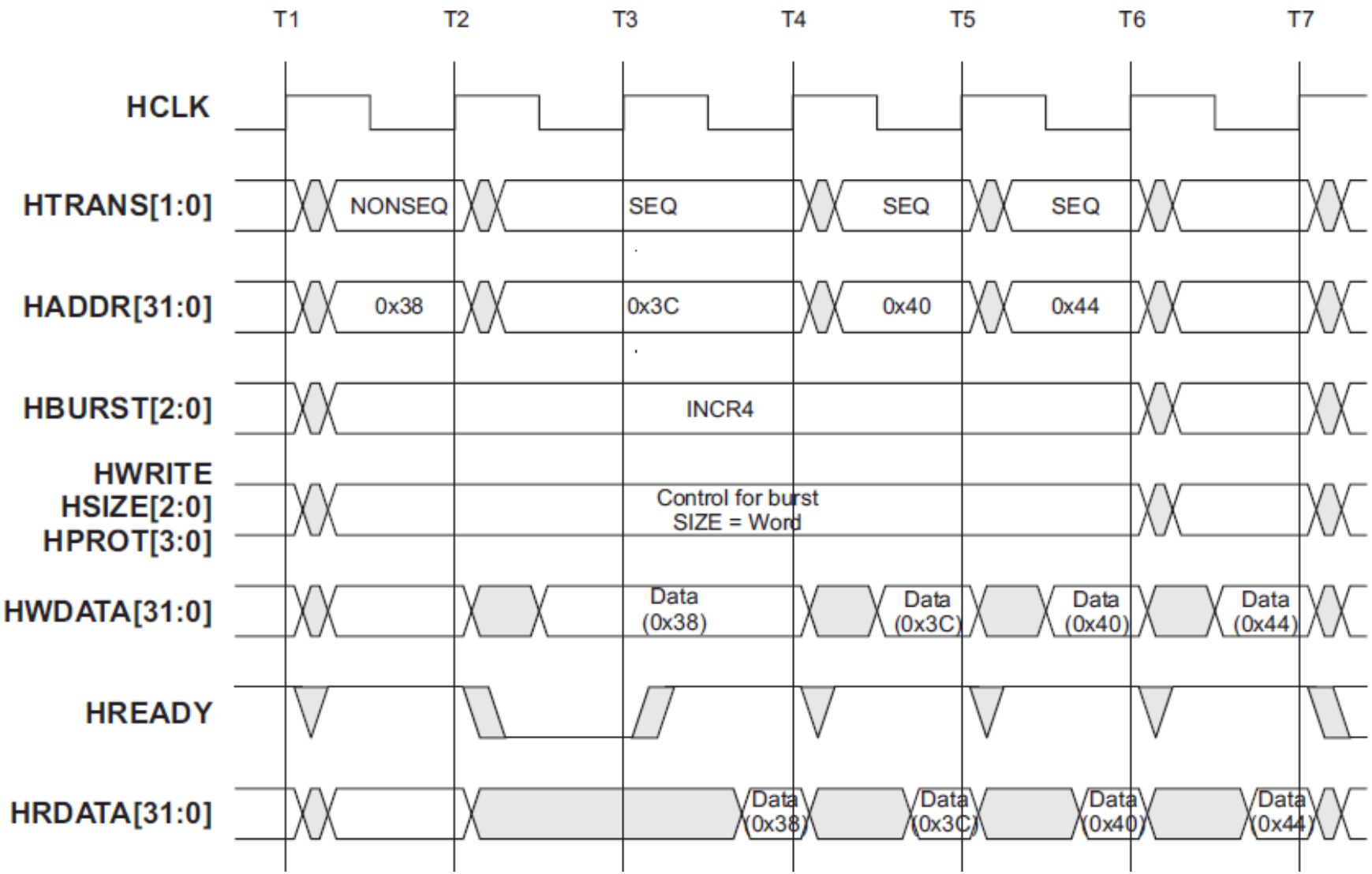



Four-beat wrapping burst

- Address of the transfers in the burst will wrap when the boundary is reached.
- A four-beat wrapping burst of word (4-byte) accesses will wrap at 16-byte boundaries.

# AMBA AHB operation

## Burst operation




Four-beat incrementing burst

- Bursts must not cross a 1kB address boundary.

# Introducing the AMBA APB

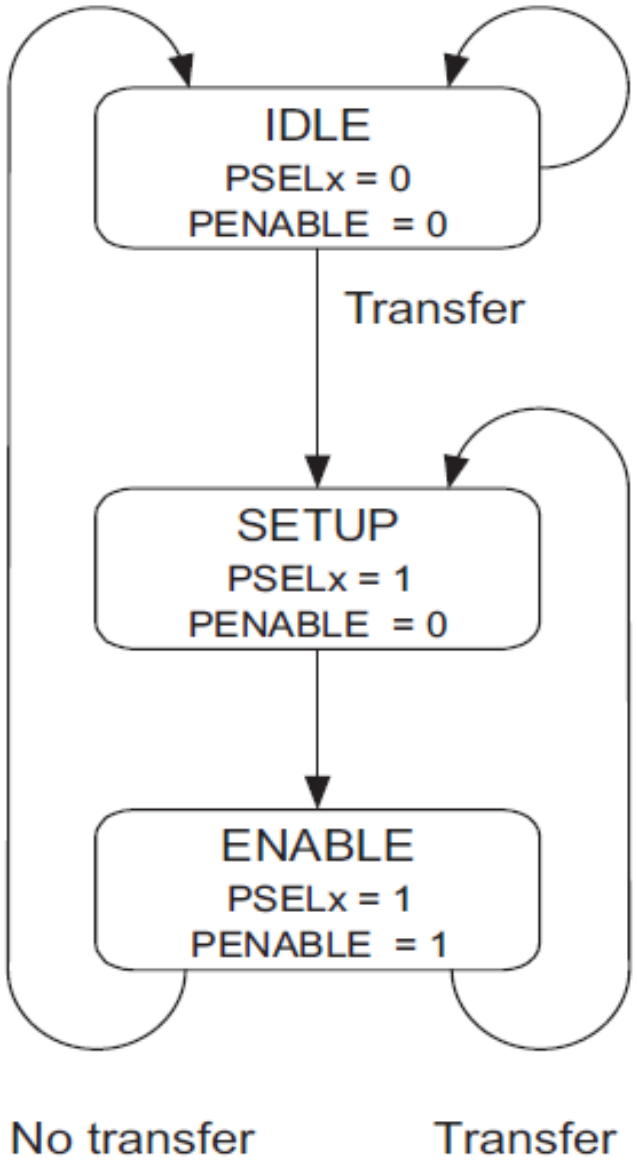
The APB is part of the AMBA hierarchy of buses

The APB bridge appears as a slave module which handles the bus handshake and control signal retiming on behalf of the local peripheral bus.

An AMBA APB implementation typically contains a single APB bridge which is required to convert AHB or ASB transfers into a suitable format for the slave devices on the APB. The bridge provides latching of all address, data and control signals, as well as providing a second level of decoding to generate slave select signals for the APB peripherals.

# AMBA APB

## State diagram

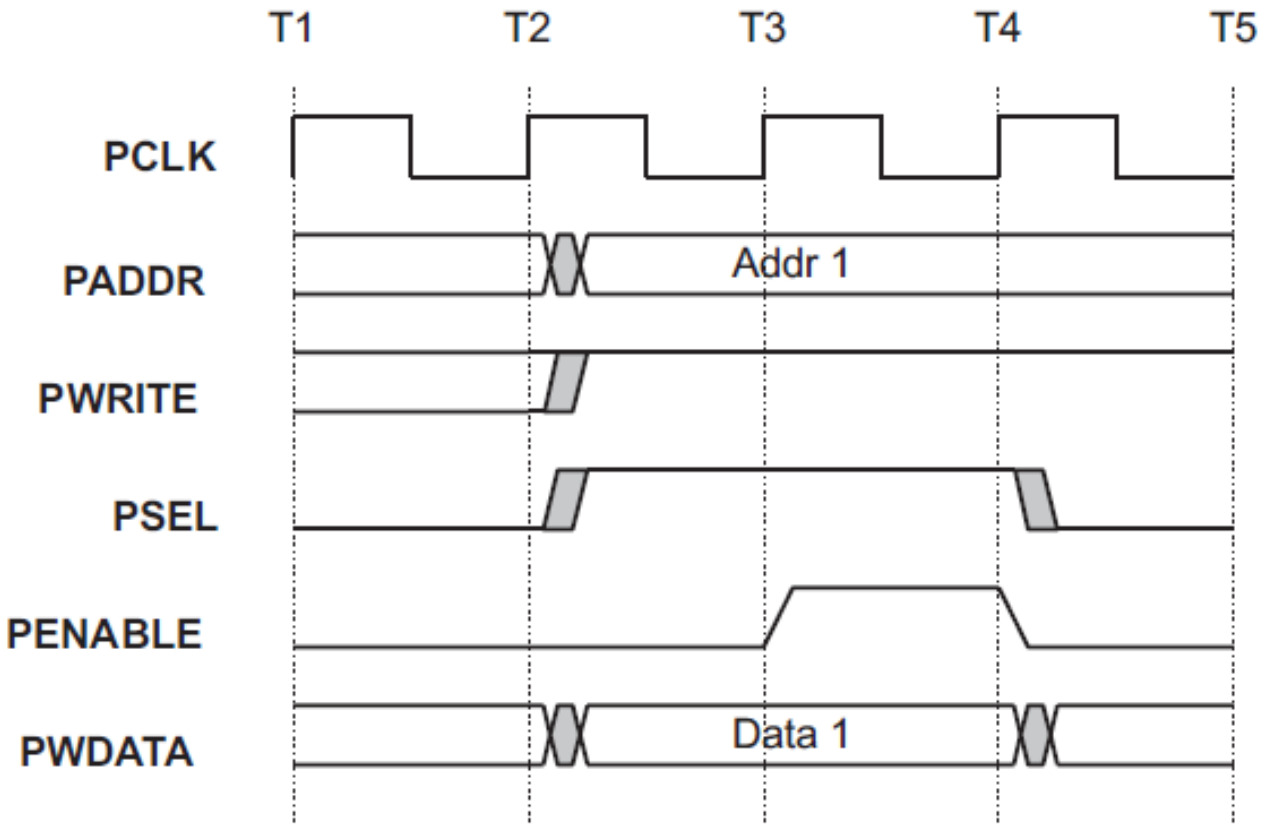



| | |
|---|---|
| IDLE | The default state for the peripheral bus. |
| SETUP | When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, **PSELx**, is asserted. The bus only remains in the SETUP state for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock. |
| ENABLE | In the ENABLE state the enable signal, **PENABLE** is asserted. The address, write and select signals all remain stable during the transition from the SETUP to ENABLE state.<br>The ENABLE state also only lasts for a single clock cycle and after this state the bus will return to the IDLE state if no further transfers are required. Alternatively, if another transfer is to follow then the bus will move directly to the SETUP state.<br>It is acceptable for the address, write and select signals to glitch during a transition from the ENABLE to SETUP states. |

- An AMBA APB bus transfer is a read or write operation of a data object, which always requires two bus cycles

# AMBA APB

## Write transfer




## Read transfer