# ARM® AMBA Bus

# (ARM Standard Parallel Bus)



## Introduction

Advanced Microcontroller Bus Architecture (AMBA)

- an open standard, on-chip bus specification by ARM
- describes a strategy for the interconnection and management of functional blocks that make up a System-On-Chip (SoC)
- provides systematic and efficient bus interface design specifications
- enhances portable and re-usable on-chip designs

Technology independence protocol

- only specified signal timing at the cycle level
- no electrical signals & timing requirement specifications

# Overview

AMBA 2 (i.e., rev 2) defines three distinct interface buses:

- *Advanced High-Performance Bus* (AHB)
- *Advanced System Bus* (ASB – first generation of the AMBA bus)
- *Advanced Peripheral Bus* (APB)

A typical AMBA-based system will consist of:

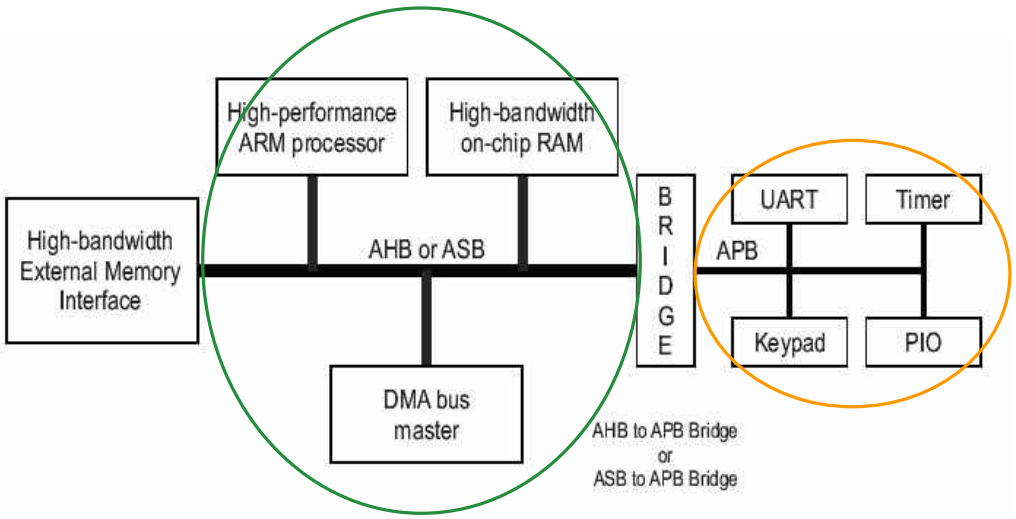
- either the AHB (preferred for a new design) or the ASB as the backbone system bus,
- and the APB secondary bus

AMBA 3 (i.e., rev 3) adds a new AXI protocol

- Advanced eXtensible Interface (AXI) protocol



# AMBA-based Microcontroller



# AMBA-based Architecture

The AHB/ASB backbone system bus is used for 'core/block' with high-bandwidth interface requirements

- CPU & co-processor
- DMA-based peripherals
- High-bandwidth peripherals with FIFO interfaces support data transfers that involve:
  - pipelined operation (two operations in one cycle)
  - multiple bus masters
  - burst transfer (AHB only)
  - split transaction (AHB only)



# AMBA-based Architecture (cont'd)

The APB secondary bus is an infrastructure to connect low-bandwidth peripherals to the higher-bandwidth pipelined main system bus

- through a 'Bridge'
- avoid loading down the high-bandwidth operation of the main system bus and reduce power consumption

Typical APB device characteristics:

- simple interface through register-mapped
- accessed under programmed control
- uses latched address and control
- ow power requirement

# AHB: Advanced High-Performance Bus



# Advanced High-Performance Bus

An AHB system bus is used to connect embedded processors such as an ARM core to high-performance peripherals, DMA controllers, on-chip memory, and interfaces.

- a high-speed, high-bandwidth bus that supports multi-master bus management to maximize system performance

Two variations of the standard AHB system bus:

- Multi-layer AHB – an interconnection scheme to optimize system bandwidth that has multiple masters
- AHB-Lite - for a system that has only one master

# AHB System

A typical AHB system will contain:

- one or more bus Master
  Processor, test interface, DMA controller that can initiate data transfer operation by providing an address and control information
- multiple bus Slaves
  APB bridge, on-chip internal memory response to read/write operation when selected by the master
- a central AHB decoder to decode the address of the transfer operation & provide control signal (HSELx) to select the individual slave

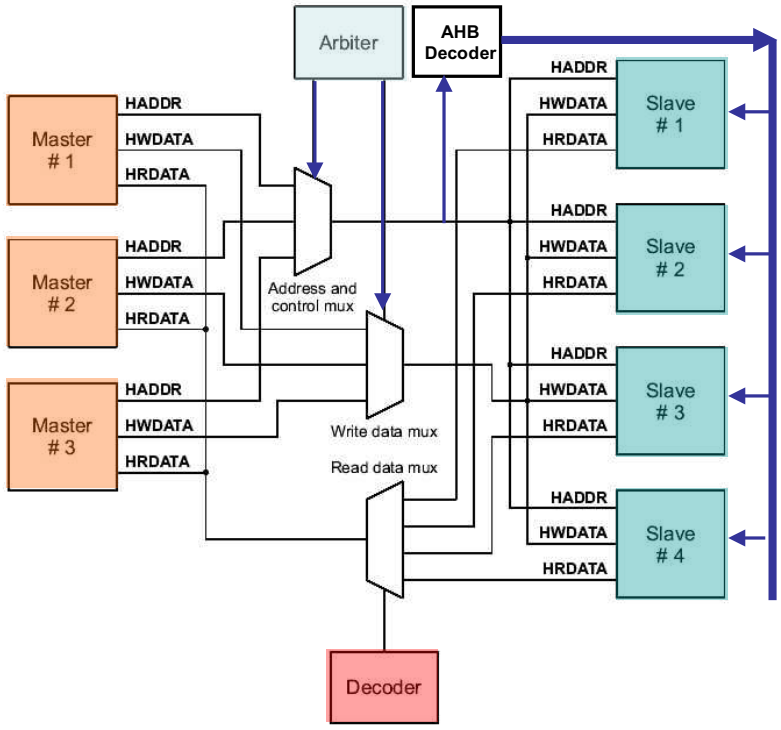


# AHB System (cont'd)

For a multi-master AHB system:

- an Arbiter is used to decide which master will be granted control of the system bus
- multiplexers controlled by the arbiter and decoder to route the buses between the master and the slave
  - Address/control bus (HADDR)
  - Write data bus (HWDATA)
  - Read data bus (HRDATA)

# Multi-Master AHB System






# AHB-Lite

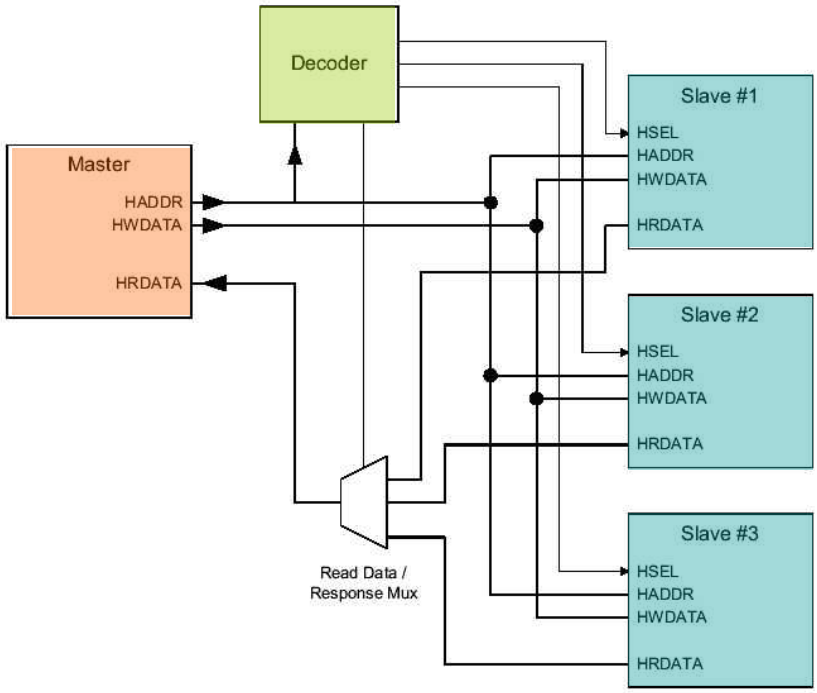
AHB-Lite is a subset of the full AHB specifications

- for design with a single master
- or multi-layer AHB where there is only one AHB master on a layer
- i.e., no arbitration is needed

Exclude the protocols required for multiple bus masters operation such as

- Request/Grant protocol with the Arbiter
- Split/Retry response from slaves

# AHB-Lite System






# AHB System Bus

The AHB signals are grouped into three buses:

a) Address + Control Signals bus
- driven by a master
- provides information on the address, direction, width, etc.

b) Write data bus – HWDATA[31:0]
- moves data from the master to a slave

c) Read data bus – HRDATA[31:0]
- moves data from a slave to the master

Separate Read and Write data buses are used
- to allow implementation of an AHB system without the use of tristate drivers

# Basic AHB Signals

Basic signals used in the AHB system:

**HCLK**: Reference clock for all bus transfers
– signals are related to the rising edge of HCLK

**HADDR[31:0]:** 32-bit system address bus

**HWDATA[31:0]:** 32-bit Write data bus
– used by the master to write data to the slave

**HRDATA[31:0]:** 32-bit Read data bus
– used by the master to read data sent by the slave



# Basic AHB Signals (cont'd)

**HWRITE**: Read/Write control signal
– HIGH for Write transfer and LOW for read transfer

**HREADY**: to indicate the status of the slave
– HIGH → the slave has completed the operation
– LOW → the slave requests for bus extension
(i.e., inserting wait states)

# AHB Data Transfer

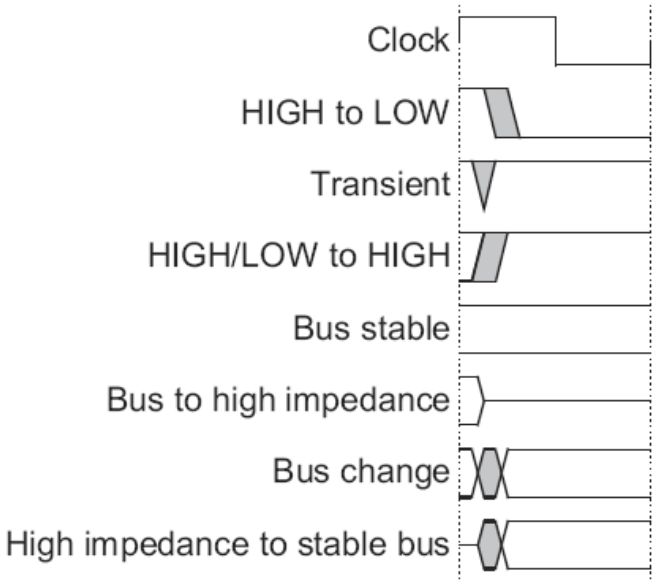
A basic AHB transfer consists of two distinct phases:

(i) Address phase
- Address and control signals are decoded and output to the selected target
- can last for one HCLK cycle

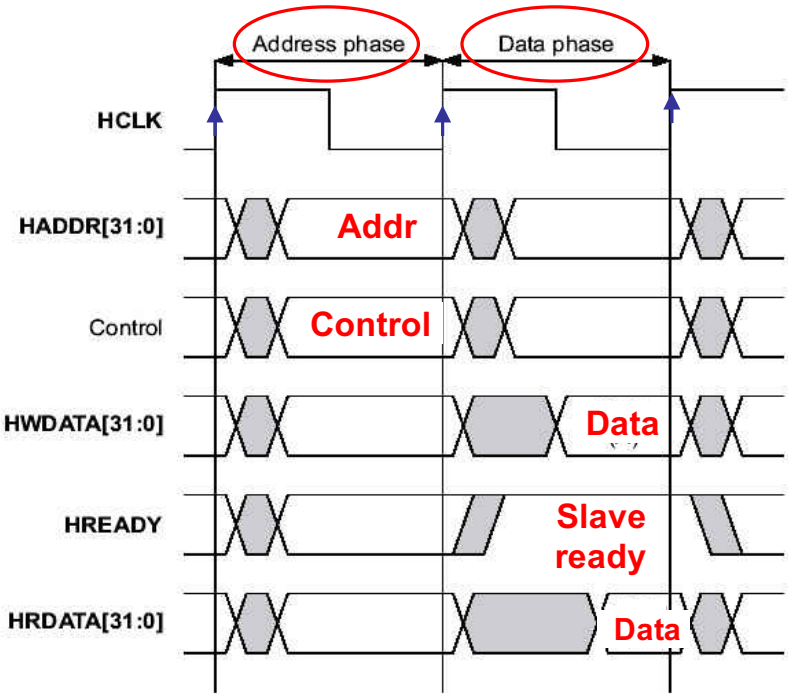
(ii) Data phase
- data are transferred (read/write) between the processor and the target
- may last for several cycles
- can be extended by inserting wait states using the HREADY signal to allow more time for the target



# Timing Diagram Conventions



# A Simple Transfer





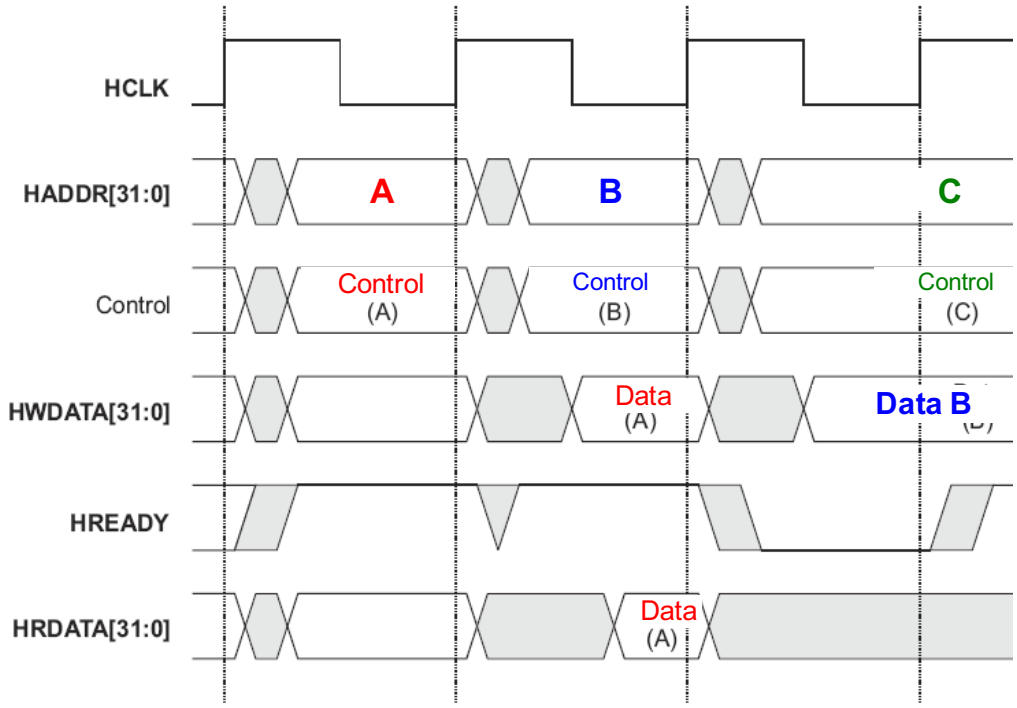

# Simple Data Transfer

- Simple data 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 slaves sample the address and control information on the next rising edge of the clock
  3. the targeted slave starts to drive the appropriate response
  4. sampled by the bus master on the third rising edge of the clock

The address phase of the next transfer can occur during the data phase of the previous transfer

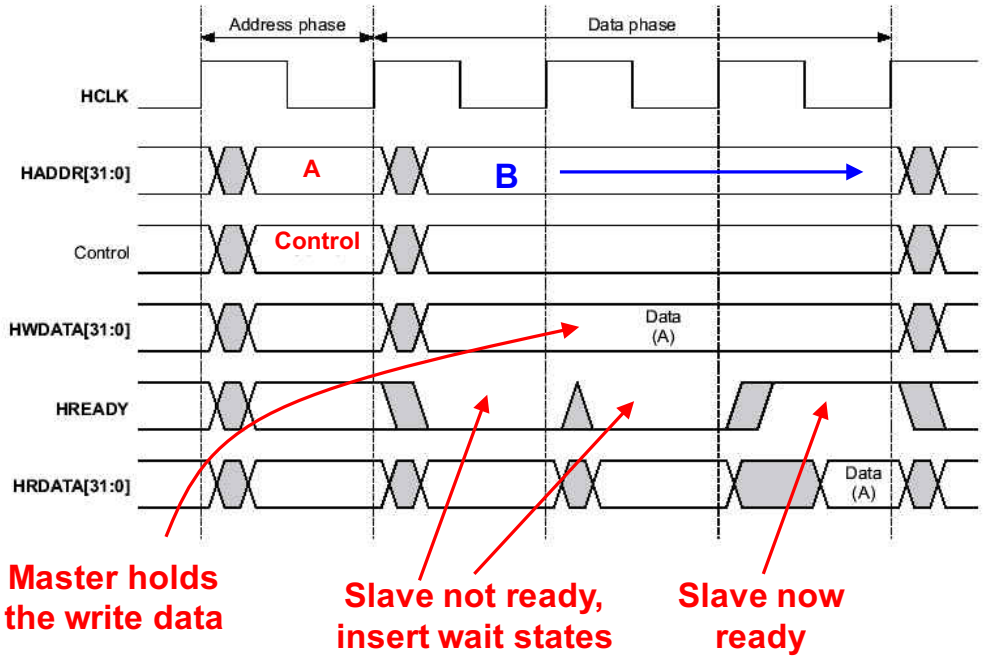
- pipelined operation of the AHB
- overlap of address phase and data phase enables higher throughput performance

# Pipelined Operation






# Transfer with Wait States



# AHB Transfer Types

For more sophisticated operations, another 2-bit signal can also be monitored to indicate the type of transfer

HTRANS[1:0]

a) 00 : Idle
   No data transfer is required, although the Master is granted the bus

b) 01 : Busy
   The master is not able to execute the next transfer immediately but want to remain connected to the slave

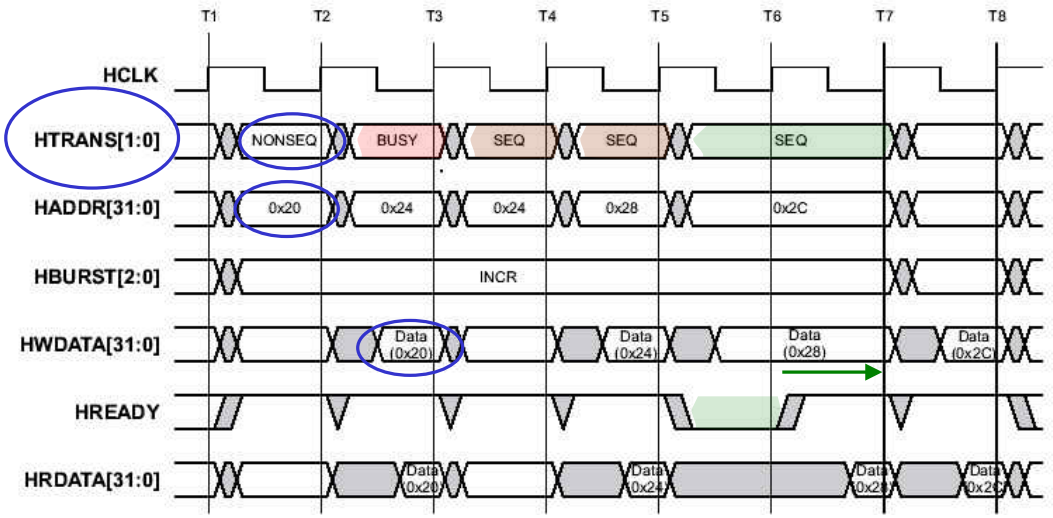


# AHB Transfer Types (cont'd)

c) 10 : NONSEQ
   A single transfer, or the first of a Burst transfer. The control information is unrelated to the previous transfer

d) 11 : SEQ
   A sequential transfer is to follow. The control information is identical to the previous transfer with appropriate address increment

# Transfer Type Example




T2-T3 : The master is not able/ready to perform a 2nd transfer

T3-T5 : Sequential transfer

T5-T6 : The slave is not ready, add a wait state to extend



# Transfer Type Example (cont'd)

T1: The first transfer is the start of a burst, hence a NONSEQUENTIAL type

T2: The master is unable to perform the second transfer of the burst immediately

- uses a BUSY transfer to delay the start of the next transfer

T3: After one cycle, the master is ready to start the next transfer in the burst, and completes with no wait states

T4: The master performs the third transfer of the burst

- but the slave is unable to complete

T5: The Wait state is due to **HREADY** negated by the slave

T6: The final transfer of the burst completes with zero wait states.

# Some Other AHB Signals

HBURST[2:0] : Burst operation (single or multiple)

HSIZE[2:0] : Transfer size (8-bit to 1024-bit)

HRESP[1:0] : Response from Slave (OKAY, ERROR, etc.)

For multiple bus masters operation:

- HBUSREQx: Master x requests the arbiter for the bus to perform a burst transfer
- HGRANTx: Master x is granted the bus by the arbiter
- HMASTLOCK: The bus is locked for the current Master



# AMBA 3 AXI protocol

AMBA Advanced eXtensible Interface (AXI) Protocol

- backward-compatible with existing AHB and APB interfaces
- extends the performance and flexibility of the AMBA-based technology
- catered for burst access

Protocol allows for

- address information to be issued ahead of the actual data transfer
- support for multiple outstanding transactions
- support for out-of-order completion of transactions

# AMBA APB



## Advanced Peripheral Bus

AMBA Advanced Peripheral Bus (APB)

- a simpler bus protocol
- design for ancillary or general-purpose peripherals (i.e., with low bandwidth)

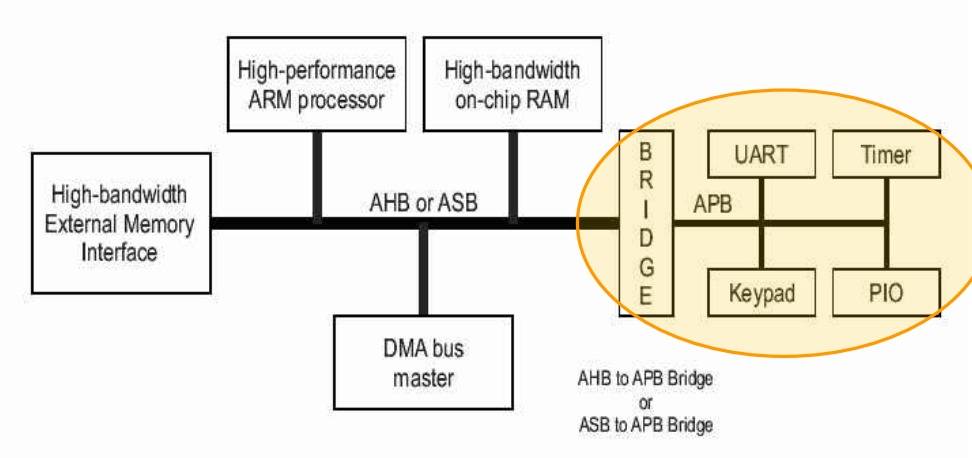
  E.g., timers, interrupt controllers, UARTs, GPIOs

Connection to the main system bus (i.e., AHB or ASB)

- through a system-to-peripheral bus bridge
- reduced loading of the main system bus by isolating peripherals behind the bridge

# APB System

Consists of a master (APB bridge) and multiple slaves (all other peripherals)






# APB Features

A simple bus structure

- non-pipelined architecture
- simple interface with the peripherals acting as slaves
- a bridge acts as the Master of APB

The bus is optimized for

- low power
- reduced interface complexity

# Bus Implementations

APB interconnection can be implemented in two ways:

i) with separate read and write data buses

- use a multiplexed bus to connect the various slaves onto the bus (i.e., like AHB)

ii) with one combined read and write data buses

- since read data and write data never occur simultaneously
- needs a tristate bus



# APB Signals

The APB signals are much less complicated than the AHB signals due to its simple operation.

- PCLK – APB bus clock
- PADDR[31:0] – APB address bus
- PRDATA[31:0] – APB read data bus (slave to bridge)
- PWDATA[31:0] – APB write data bus (bridge to slave)
- PSELx – APB select (to select slave x)
- PENABLE – APB Enable timing strobe
- PWRITE – APB Read/Write control signal
- PRESETn – APB Reset signal (active LOW)

# APB Data Transfer

The data transfer cycle on APB consists of three states:

1) IDLE: default state of APB
   PSELx = 0; PENABLE = 0

2) SETUP: Slave is selected (one cycle)
   PSELx = 1; PENABLE = 0

3) ENABLE: Data transfer occurs (one cycle)
   PSELx = 1; PENABLE = 1

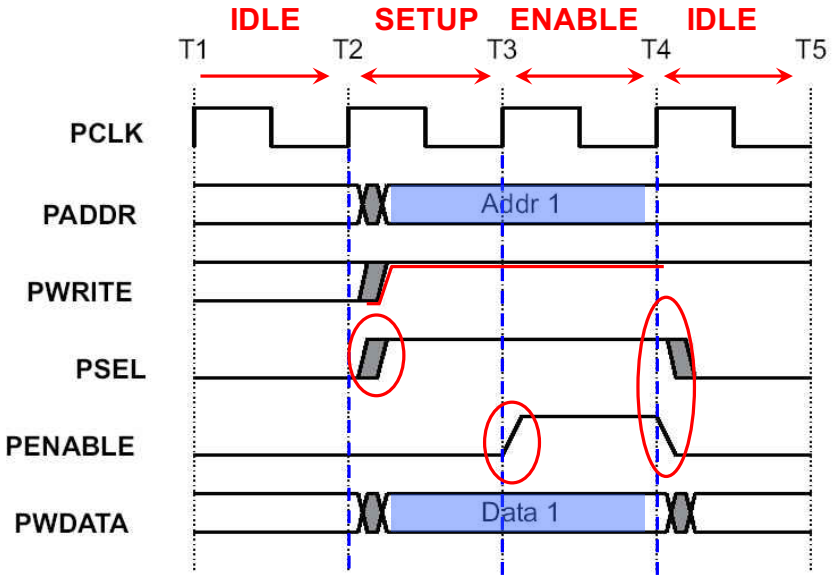
After the ENABLE cycle (i.e., transfer is completed) the bus can

- proceed to the IDLE state if no further transfer is required
- go back to the SETUP state if more transfer is needed



# APB Write Transfer

Timing diagram of a Write transfer:



# APB Write Transfer

a) Initially in the IDLE cycle: T1–T2

b) PSEL is asserted, together with the address, data, and write control signals after the rising edge of the clock at T2
   - enter the SETUP cycle: T2–T3

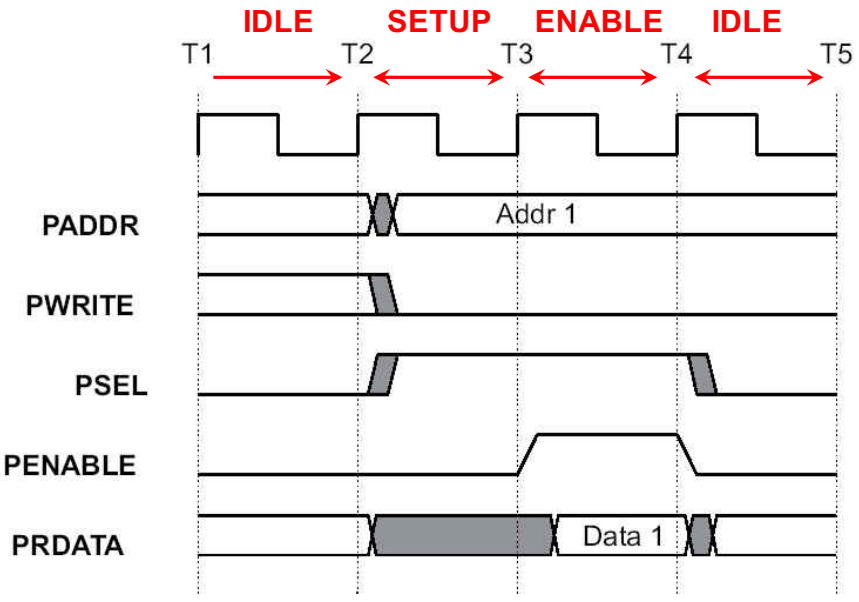
c) PENABLE is asserted at the rising edge of the clock at T3
   - enter the ENABLE cycle: T3–T4
   - address, data, and control signals all remain valid throughout the ENABLE transfer cycle

c) PENABLE is de-asserted at the end of the transfer
   - the PSEL signal will also be de-asserted
   - back to the IDLE cycle: T4–T5



# APB Read Transfer

Timing diagram of a Read transfer



# No Wait State on APB

Why is there no Wait signal on the APB?

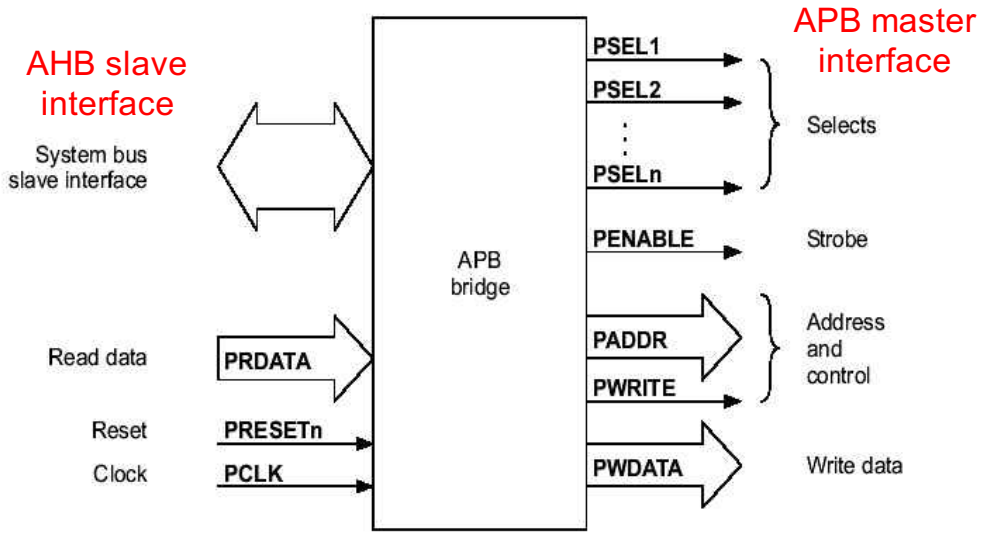
- APB is designed to implement as simple an interface as possible.
- Many APB peripherals are slow devices, e.g., UARTs, which are normally accessed via registers.
  - typically the driver software will first access a status register to determine that data is available
  - then access the data register.

  These accesses are possible without the need of additional wait state(s).

Peripherals that require wait states can/should be designed as AHB slaves.



# APB Bridge Interface




The APB bridge is the only bus master on the APB

- but appears as a local slave on its AHB interface

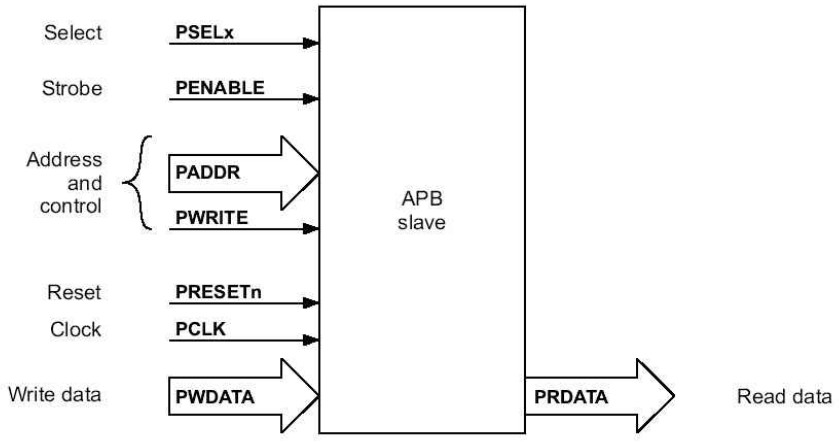
# APB Bridge

Main functions of the APB bridge:

- decodes the address to generate the PSELx peripheral select signal
- latches the address and holds it valid throughout the transfer
- drives the AHB data onto the APB for a write transfer
- drives the APB data onto the AHB system bus for a read transfer
- generates timing strobe PENABLE for the transfer



# APB Slave Interface

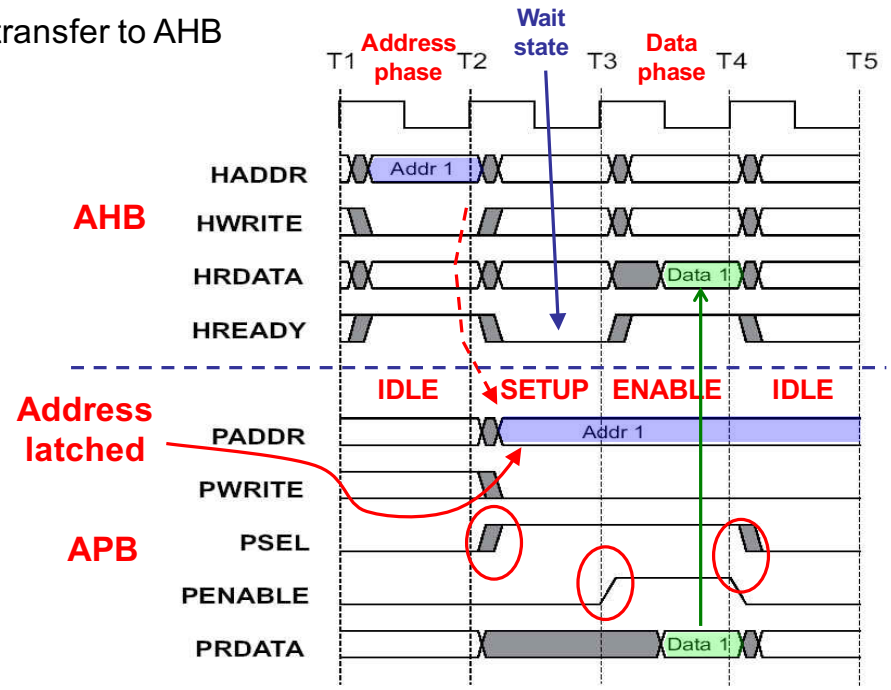



For write transfer, data can be latched on

- the rising edge of PCLK, when PSEL is HIGH
- the rising edge of PENABLE, when PSEL is HIGH

# Interfacing APB to AHB

Read transfer to AHB






# APB to AHB: Read Transfer

Read the address phase issued at AHB after T1

- The address is sampled by the APB bridge at T2
- APB enters the SETUP cycle during T2–T3
- ENABLE cycle from T3–T4
  – peripheral provides the read data

The APB bridge routes the data directly back to AHB,
the AHB bus master samples the data at the rising edge of T4.

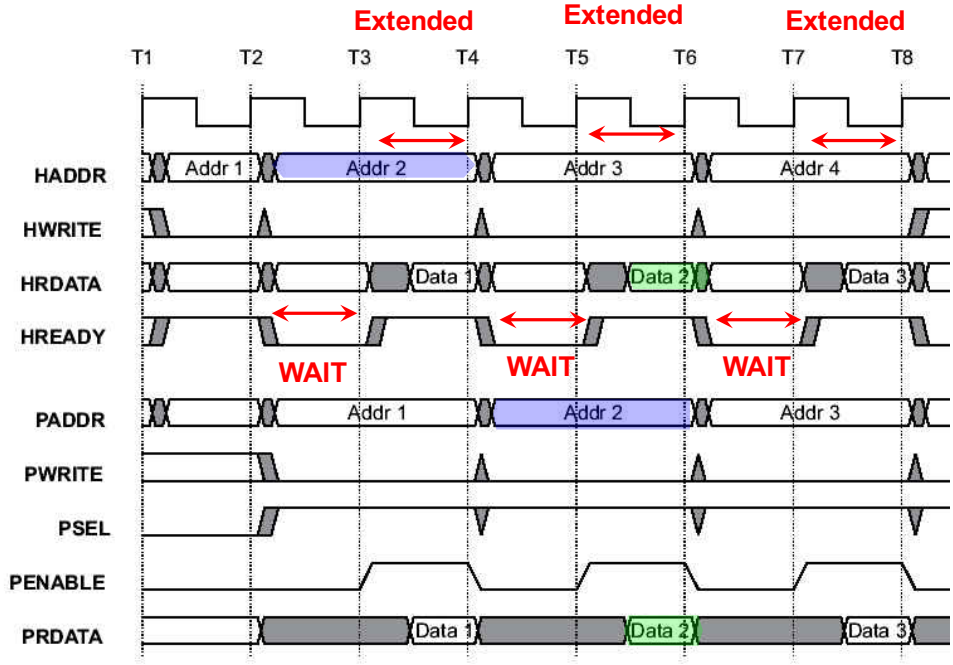
- No wait state between the APB and AHB data transfer for a single read transfer.

But one wait state incurred between the initial address phase and data phase on AHB.

- More wait states may be required for subsequent read transfers for a burst transfer operation.

# Interfacing APB to AHB

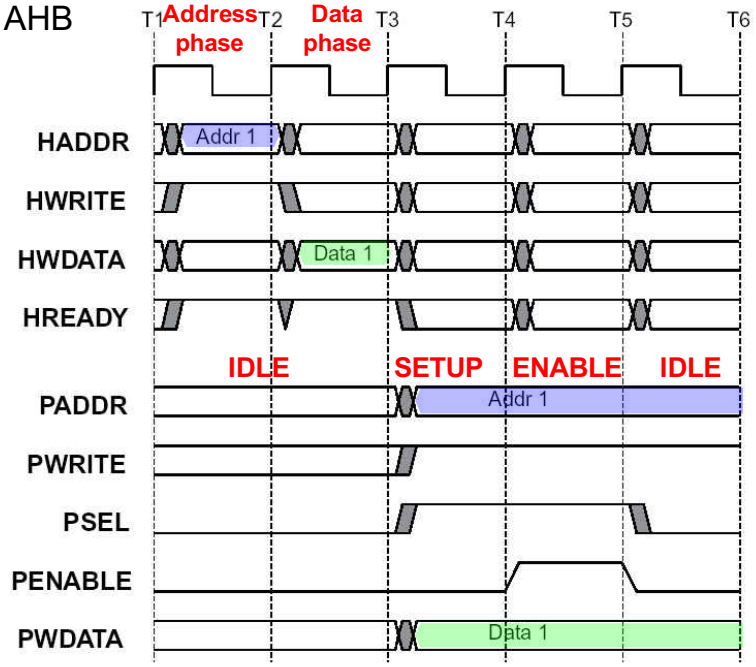
Burst of read transfer to AHB






# Interfacing APB to AHB (cont'd)

Write transfer from AHB



# AHB to APB: Write Transfer

Write transfer from AHB – transfer starts at T2

- The bridge will first sample and hold the address, followed by the data
- The address and data are latched for the duration of the write transfer on the APB

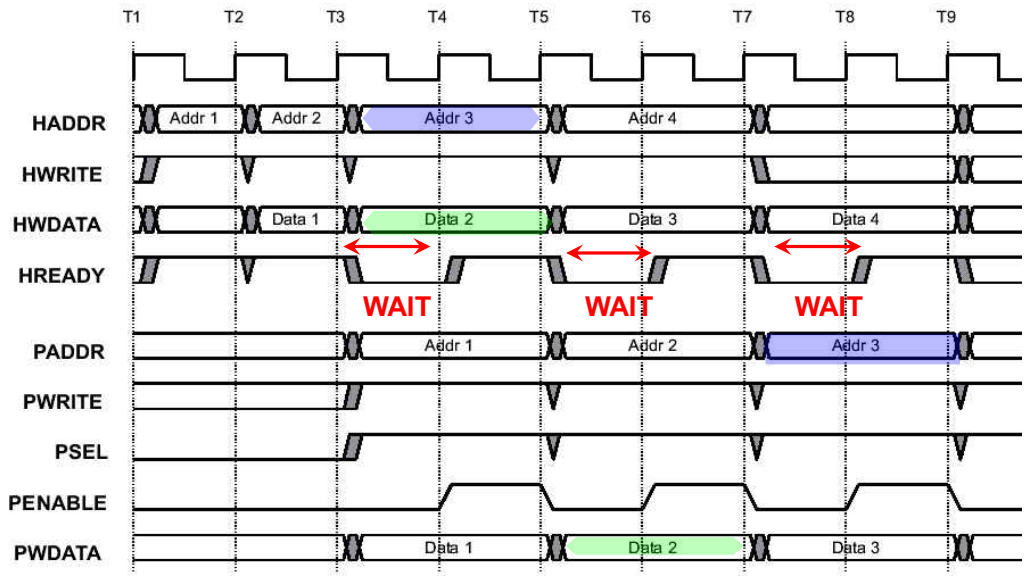
The No Wait state is incurred for a single write transfer.

- But the Wait state may be required for burst of write transfers
- The bridge needs to contain two address registers:
  - one for the current latched address
  - the other for sampling the next address



# Interfacing APB to AHB

Burst of write transfers from AHB



# Summary

AMBA provides a standard to interface subsystems together on an SoC design.

A typical AMBA design will use two buses:
- AHB (or ASB) main system bus
- APB secondary bus for peripherals

AHB provides a high bandwidth bus capable of supporting multiple bus masters.

Arbitration is needed when there are multiple bus masters on the AHB.

APB provides a low power bus, reduced interface complexity suitable for low-bandwidth general purpose peripherals.