APB Basics

1. APB Overview

APB (Advanced Peripheral Bus) is a **simple, low-speed bus** used to connect a master (e.g., CPU or bridge) to peripherals (UART, GPIO, Timer).

Key characteristics:

- One master talks to one slave at a time → **point-to-point communication**
- No arbitration, no bursts, no pipelines → easy to verify
- Works in **two phases**: **SETUP** and **ENABLE**
- Common versions: **APB2**, **APB3** (most common), **APB4** (adds extra features)

2. Master and Slave

Role	**Description**	**Example in UART Project**
Master	Initiates transactions (read/write)	CPU or AXI→APB Bridge
Slave	Responds to transactions	UART (APB Slave)

Bridge Role: AXI→APB bridge is a slave on AXI side, master on APB side. Bridge converts AXI transactions into APB transactions

Master = the block that initiates bus transactions.

- Starts a read or write operation
- Provides **address, control signals, and data (if write)**

Examples: CPU, DMA, AXI→APB Bridge

Notes for project: CPU talks AXI**,** AXI→APB Bridge acts as **APB master**

Slave = the block that responds to master requests.

- Waits for **PSEL=1** and address
- Accepts writes → updates registers
- Responds to reads → drives PRDATA

Examples: UART, GPIO, Timer, ADC

In UART project, the **APB slave logic is part of the UART module**

3. Peripherals

Peripheral = a hardware block connected to the bus that does some task.

Examples: UART, GPIO, Timer, ADC

- Peripherals are usually APB slaves in SoCs.
- They have internal registers that control operation (e.g., BAUD register in UART).
- **In your UART project:**

  CPU → AXI/AXI-Lite → APB Bus → UART Peripheral

- **Inside UART Peripheral:**
  - **TX Engine** → transmits data
  - **RX Engine** → receives data
  - **APB Slave interface** → communicates with APB bus, handles read/write to registers

✅ Both TX and RX are part of UART, controlled via APB registers.

4. Point-to-Point Communication

- APB is **simple**: one master talks to **one slave at a time**
- No multi-master arbitration required
- Sequence: Master → Slave (write/read, address + data)
- Only one PSEL=1 at a time for the selected slave

5. APB Signals

Signal	Description
PSEL	Select the slave (only one slave active)
PADDR	Register address
PWRITE	1=write, 0=read
PWDATA	Data to write (master → slave)
PRDATA	Data read from slave (slave → master)
PENABLE	Indicates enable phase of transfer
PREADY	Slave ready / handshake signal
PSLVERR	Optional: slave error signal

6. APB Transfer Phases

6.1 Setup Phase

- Master drives signals: PSEL=1, PADDR=address, PWRITE=0/1, PWDATA=data (if write), PENABLE=0
- Slave just “prepares” to respond

6.2 Enable Phase

- Master sets PENABLE=1 → “perform operation”
- Slave responds:
    - For write: updates register
    - For read: drives PRDATA
    - Sets **PREADY=1** when done

6.3 Completion

- Master samples data (if read)
- Master deasserts **PSEL=0, PENABLE=0**
- Transaction finished

Notes About Handshake

- **PENABLE + PREADY = handshake**
- If slave is fast → PREADY=1 immediately → low latency
- If slave is slow → PREADY=0 → master waits in enable phase
- Ensures **safe and reliable transfers**

7. Address Decoding

- Each slave decodes **PADDR** to figure out which register to access
- Only the **selected slave** with **PSEL=1** responds
- Example (UART registers):
  0x00 → CONTROL

  0x04 → STATUS

  0x08 → BAUD

  0x0C → TXDATA

  0x10 → RXDATA
    - Only these addresses are valid for this slave

8. Setup Time & Metastability (Why APB is Reliable)

- **Setup time:** data must be stable **before clock edge**
- **Hold time:** data must remain stable **after clock edge**
- **Metastability:** happens if setup/hold violated → flip-flop output unstable
- APB handshake + separate setup/enable phases **prevents metastability**

9. For your UART project, **APB3 is sufficient**

10. Summary – What We Verify in APB

- **Write operation:** data correctly written to registers
- **Read operation:** data returned correctly
- **Address decoding:** only targeted register updated
- **Handshake:** PSEL/PENABLE/PREADY sequence correct
- **PREADY timing:** slave responds correctly (fast/slow)
- **Multiple transactions:** back-to-back operations work
- **Edge cases:** min/max values, busy slave, empty/full flags

◆ Example APB UART Transaction (Setup → Enable → Complete)

Cycle 1: Setup

PSEL=1 (UART selected)

PADDR=BAUD register

PWRITE=1

PWDATA=0x55

PENABLE=0

Cycle 2: Enable

PENABLE=1

Slave captures data

PREADY=1 → transaction done

Cycle 3: Complete

PSEL=0

PENABLE=0