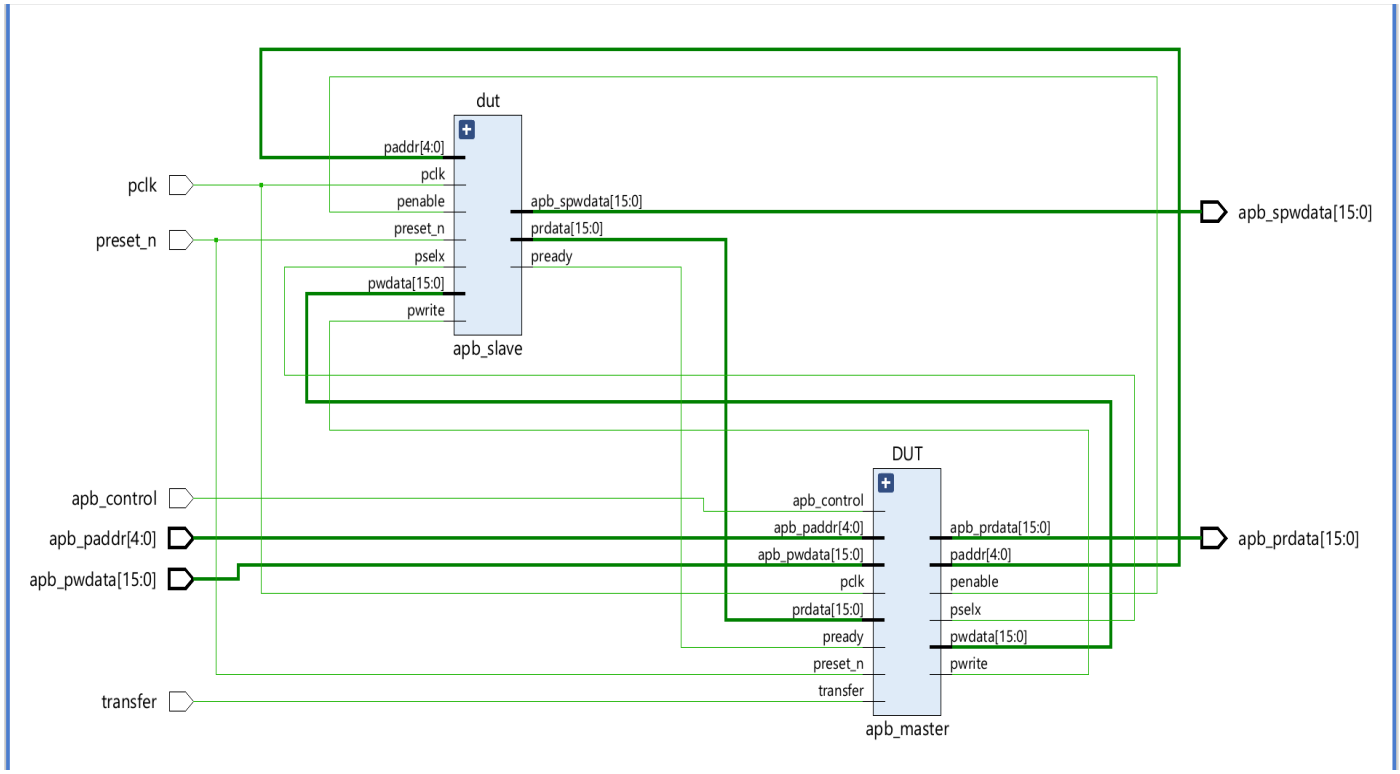
APB PROTOCOL

This project implements a basic Advanced Peripheral Bus (APB) interface using three SystemVerilog modules: an APB wrapper, an APB slave, and an APB master. The design is parameterizable (using `addr_width` and `data_width`) and follows the APB protocol by defining clear state machines for handling transactions.



1. APB Wrapper Module

Purpose:
The APB wrapper module integrates the APB master and APB slave modules into a single top-level unit. It facilitates the routing of control, address, and data signals between a higher-level interface and the slave memory, effectively serving as the bridge for APB transactions.

Parameters:

- `addr_width` (default = 5): Determines the width of address signals.
- `data_width` (default = 16): Determines the width of data signals.

Ports:

- **Inputs:**
  - `pclk`: APB clock signal (rising edge triggered).
  - `preset_n`: Active-low reset signal that resets the state machines.

    - `transfer`: Control signal indicating when a transaction is requested.
    - `apb_paddr`: Address input from the higher-level interface.
    - `apb_pwdata`: Write data input from the higher-level interface.
    - `apb_control`: Transaction control signal (1 for write, 0 for read).
- **Outputs:**
    - `apb_spwdata`: Data output from the slave during write transactions.
    - `apb_prdata`: Data output from the slave during read transactions.

Functionality:

- **Instantiation:**
  The wrapper instantiates both the APB master and the APB slave. The master is responsible for generating APB control signals, while the slave handles the memory operations.
- **Signal Routing:**
  It routes the signals between the master and the slave:
    - The master generates signals such as `pselx`, `penable`, `paddr`, `pwrite`, `pwdata`, and `pstrb`.
    - The slave uses these signals to perform read/write operations on an internal memory array and returns data (via `prdata`) and a ready signal (`pready`) to indicate completion.
- **Overall Operation:**
  This integration enables seamless communication where higher-level commands are translated into APB transactions, processed by the slave, and the results are then sent back through the wrapper.

---

2. APB Slave Module

Purpose:
The APB slave module implements a basic slave device that handles read and write transactions. It contains an internal memory array where data is stored or retrieved based on the APB transactions initiated by the master.

Parameters:

- `addr_width` (default = 5): Determines the width of the address bus.
- `data_width` (default = 16): Determines the width of the data bus.

Ports:

- **Inputs:**
    - `pclk`: APB clock signal.
    - `preset_n`: Active-low reset signal.
    - `pselx`: Slave select signal from the master.
    - `penable`: APB enable signal (used to latch data).
    - `pwrite`: Write control signal (indicates write when high).
    - `paddr`: Address for accessing the memory.

    - `pwdata`: Data to be written into memory.
- **Outputs:**
    - `apb_spwdata`: Output reflecting the data written during a write transaction.
    - `prdata`: Data output for read transactions.
    - `pready`: Signal indicating the slave is ready to complete the transaction.

Internal Implementation:

- **Memory Array:**
  An internal memory `mem` is defined as an array of `data_width`-bit registers with a depth of 2^`addr_width`.
- **State Machine:**
  The slave utilizes an FSM with three states:
    - **IDLE:** No active transaction.
    - **SETUP:** The address and control signals are being established.
    - **ACCESS:** Data transfer is occurring.
- **State Transitions:**
    - **IDLE → SETUP:** Occurs when `pselx` is asserted without `penable` (indicating the start of a transaction).
    - **SETUP → ACCESS:** When both `pselx` and `penable` are asserted.
    - **ACCESS → IDLE/SETUP:** The FSM returns to IDLE if `pselx` is deasserted, or reverts to SETUP if `pselx` remains high.
- **Data Handling:**
    - **Write Operation:**
      If `pready` is high and `pwrite` is asserted, the module writes `pwdata` into the memory at the location specified by `paddr` and outputs this data on `apb_spwdata`.
    - **Read Operation:**
      If `pready` is high and `pwrite` is low, the data from the memory at `paddr` is presented on `prdata`.
- **Ready Signal:**
  The `pready` signal is asserted only during the ACCESS state to signal the completion of a transaction.

---

3. APB Master Module

Purpose:
The APB master module is responsible for generating the required control and data signals to perform read or write transactions with the APB slave. It serves as the interface between the higher-level control logic and the APB slave device.

Parameters:

- `addr_width` (default = 5): Determines the width of the address signals.
- `data_width` (default = 16): Determines the width of the data signals.

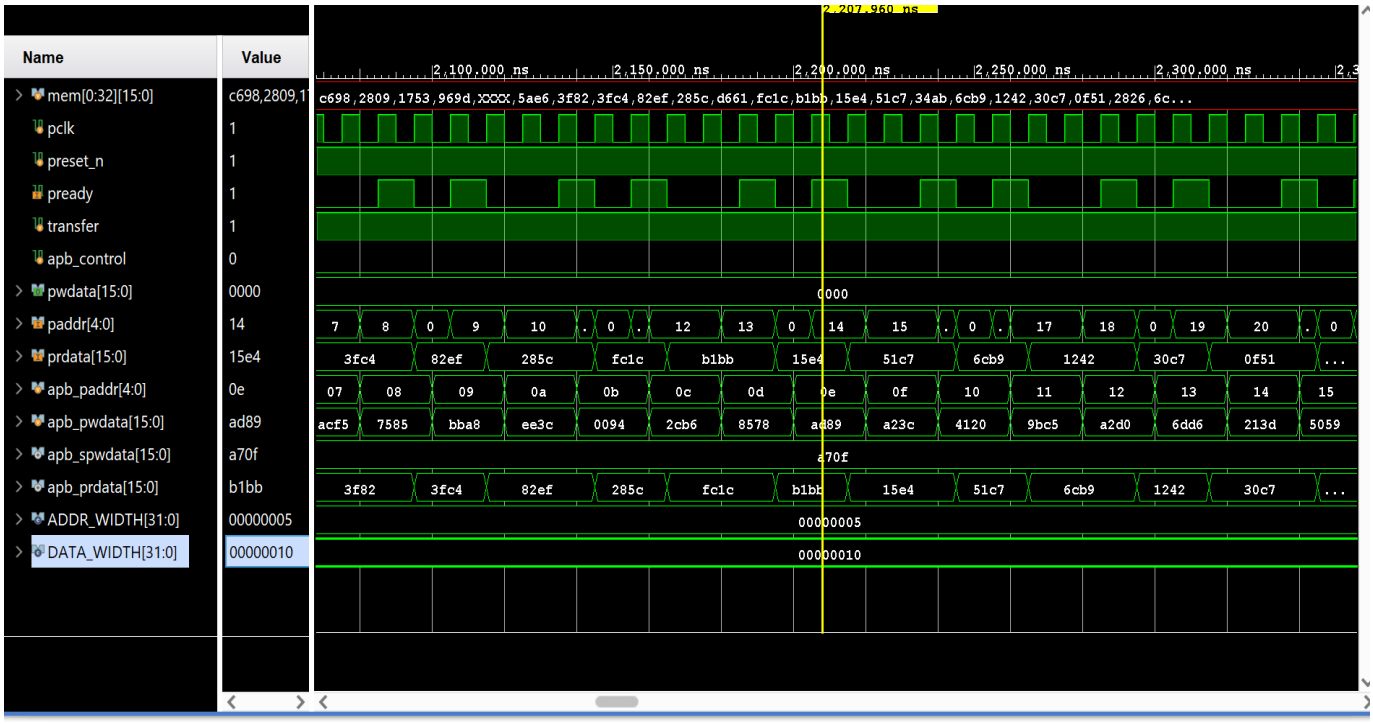
Ports:

- **Inputs:**
  - `pclk`: APB clock signal.
  - `preset_n`: Active-low reset signal.
  - `pready`: Signal from the slave indicating transaction readiness.
  - `transfer`: Signal from the higher-level interface to initiate a transaction.
  - `prdata`: Data from the slave during read transactions.
  - `apb_paddr`: Address provided by the higher-level interface.
  - `apb_pwdata`: Write data provided by the higher-level interface.
  - `apb_control`: Control signal (1 for write, 0 for read).
- **Outputs:**
  - `paddr`: Address driven to the slave.
  - `pselx`: Slave select signal.
  - `penable`: Enable signal used during the ACCESS state.
  - `pwrite`: Write control signal, directly derived from `apb_control`.
  - `pwdata`: Write data output to the slave.
  - `pstrb`: Byte strobe signals for write transactions (one per byte lane).
  - `apb_prdata`: Read data output that latches data from the slave.

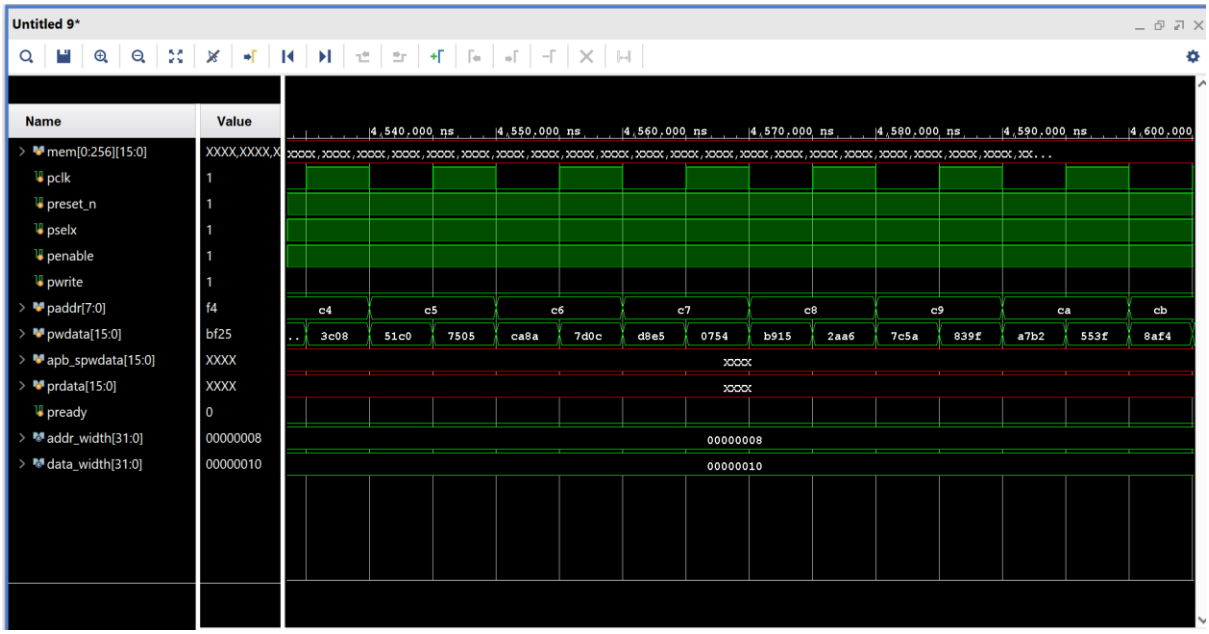
Internal Implementation:

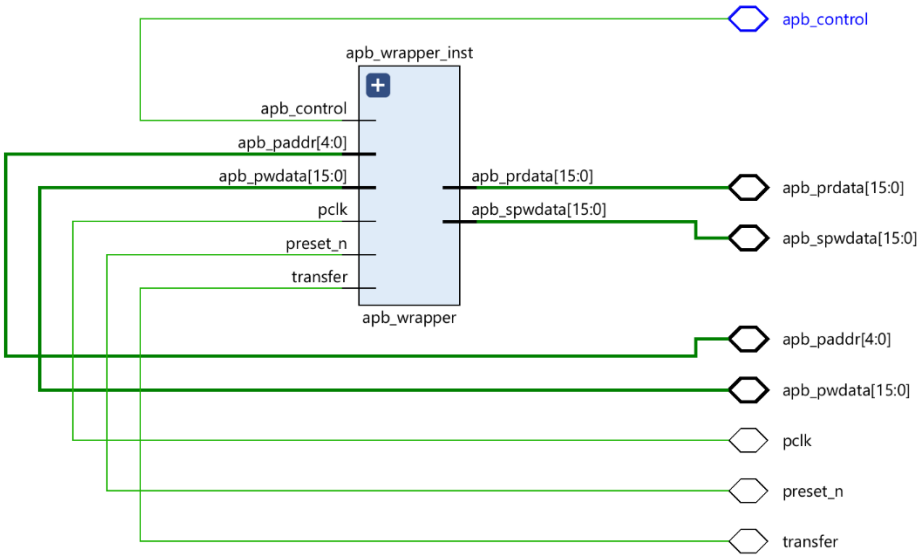
- **Finite State Machine (FSM):**
  The master uses an FSM with three states:
  - **idle:** No active transaction.
  - **setup:** Address and control signals are being set up.
  - **access:** Data transfer is active.
- **State Transitions:**
  - **idle → setup:** Triggered by a transfer request from the higher-level interface.
  - **setup → access:** Proceeds if the transfer signal remains asserted.
  - **access → idle:** Occurs when the slave asserts `pready`, indicating the transaction is complete.
- **Signal Generation:**
  - **Address and Data Assignment:**
    During both the setup and access states, the master assigns the higher-level provided address (`apb_paddr`) to the slave interface and routes the write data (`apb_pwdata`) if the operation is a write.
  - **Control Signals (`pselx` and `penable`):**
    - In **idle**, both signals are deasserted.
    - In **setup**, `pselx` is asserted while `penable` remains deasserted.
    - In **access**, both `pselx` and `penable` are asserted.
  - **Byte Strobe (pstrb):**
    For write transactions, the module generates a `pstrb` signal covering all byte lanes based on the `data_width`.
- **Read Data Capture:**
  A temporary register (`temp_apb_prdata`) is used to latch the read data from the slave when `pready` is asserted in a read transaction.

Read transactions



Write transactions





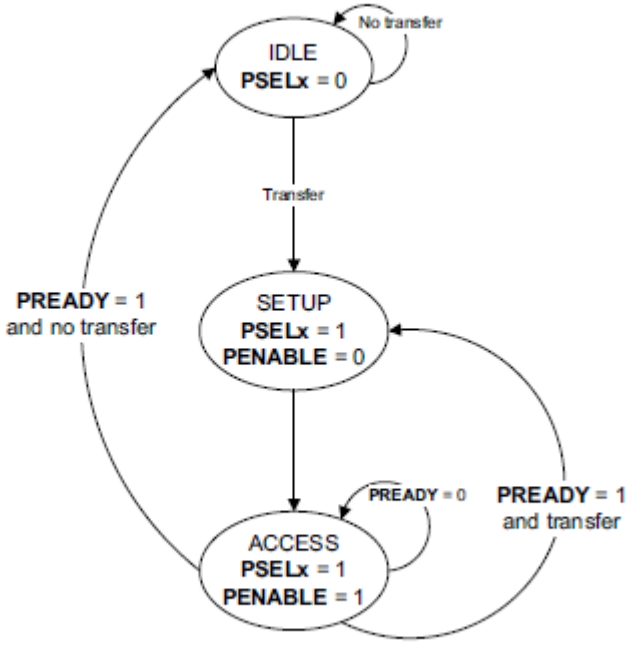


Figure 4-1 State diagram