AHB TO WISHBONE RTL Skeleton for Bridge - Status Update

By : Bhavana Shreya Padala

Date: Jan 13th, 2026

Task 1 – RTL Skeleton Definition and Interface Setup

This document describes the completion of Task 1 for the ahb2wishbone project.
The objective of this task is to define the RTL module structure, bus interfaces, reset behavior, and clock assumptions for an AHB-to-Wishbone bridge. This task establishes the foundational RTL framework upon which functional behavior will be implemented in subsequent tasks.

The bridge is intended to accept AHB-Lite transactions and later translate them into Wishbone Classic cycles while acting as a Wishbone master.

1. Scope

- Define the top-level RTL module for the AHB-to-Wishbone bridge.
- Declare and document all AHB-Lite slave interface signals.
- Declare and document all Wishbone Classic master interface signals.
- Establish clocking and reset assumptions.
- Ensure reset-safe default behavior on all outputs.
- Provide a synthesizable, lint-clean RTL skeleton without functional logic.

Out of Scope for Task 1:

- AHB transfer qualification
- Wishbone handshake logic
- FSM or wait-state logic
- Read/write datapath implementation\

2. Clock and Reset Assumptions

- The bridge operates in a single clock domain.
- Both AHB and Wishbone interfaces are synchronous to clk.
- No clock-domain crossing (CDC) logic is implemented.
- Reset is active-low and synchronous (rst_n).
- All outputs are driven to known, safe values during reset.

3. AHB-Lite Interface Definition

The bridge exposes an AHB-Lite slave interface with the following signals:

- haddr : AHB address bus
- htrans : Transfer type (IDLE, BUSY, NONSEQ, SEQ)

- hwrite : Read/write indication
- hsize : Transfer size
- hburst : Burst type
- hwdata : Write data bus
- hsel : Slave select
- hready_in : Ready input from AHB bus

AHB response signals provided by the bridge:

- hrdata : Read data bus
- hready_out : Ready output (default asserted)
- hresp : Transfer response (OKAY only)

Note: Error responses are not supported in this bridge implementation.

4. Wishbone Interface Definition

The bridge acts as a Wishbone master and defines the following Classic-cycle signals:

- wb_cyc_o : Indicates an active Wishbone bus cycle
- wb_stb_o : Indicates a valid Wishbone transfer request
- wb_we_o : Write enable (1 = write, 0 = read)
- wb_adr_o : Wishbone address
- wb_dat_o : Write data
- wb_sel_o : Byte select signals

Wishbone response signals:

- wb_ack_i : Acknowledge from slave
- wb_dat_i : Read data from slave

No advanced Wishbone features (STALL, ERR, RTY, pipelining, bursts) are included at this stage.

5. Reset and Default Behavior

On reset assertion:

- AHB outputs are driven to safe defaults:
  hready_out asserted
  hresp set to OKAY
  hrdata cleared

- Wishbone control signals are deasserted:

  wb_cyc_o = 0

wb_stb_o = 0
wb_we_o = 0

- Address, data, and select signals are cleared

This guarantees that both buses remain in an idle and safe state after reset deassertion.

6. RTL Coding Style and Quality

- Fully synthesizable RTL (no delays or unsynthesizable constructs)
- Single synchronous always @(posedge clk) block
- No inferred latches
- No unused or floating signals
- Explicit inline documentation of assumptions and intent
- Clean compilation with warnings enabled

The RTL skeleton is structured to allow straightforward extension in future tasks without refactoring.

7. Verification Status

A minimal testbench was created to support compilation and simulation using EDA Playground (Icarus Verilog).
The design compiles and simulates successfully with no errors or warnings, confirming correctness of the RTL structure and interfaces.

8. Summary and Next Steps

Task 1 is complete.
The RTL skeleton provides a stable and documented foundation for functional implementation.

Next Task:

Task 2 – AHB transfer detection and control logic
Introduction of internal state and Wishbone transaction sequencing