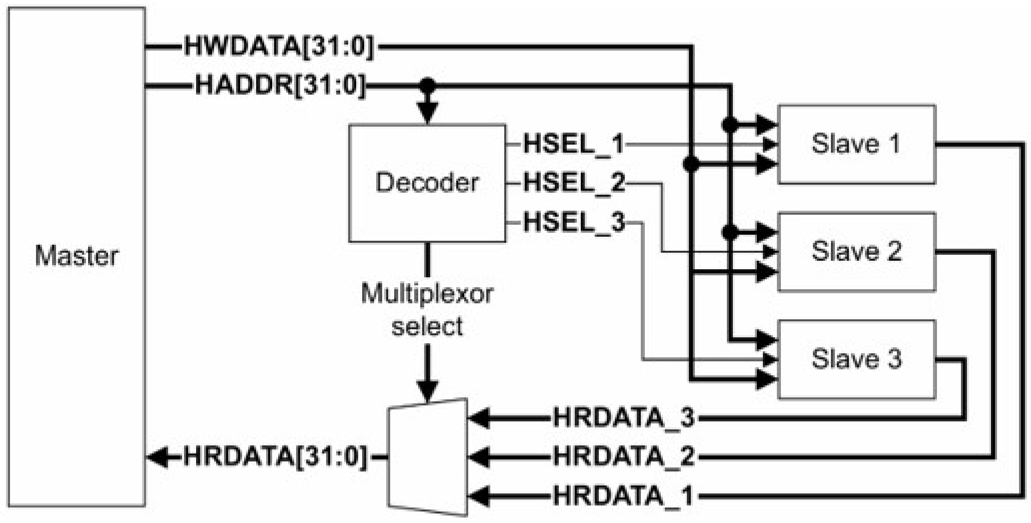
AHB Master Project Documentation

Table of Contents

About the protocol

- AMBA AHB-Lite addresses the requirements of high-performance synthesizable designs.
- It is a bus interface that supports a single bus master and provides high-bandwidth operation.



Supported and Unsupported Features

Supported Features:

- Implements AHB-Lite Master functionality, capable of generating both read and write transactions.
- Supports burst modes of operation:
  - INCR: Incremental bursts where addresses increase sequentially.
  - SINGLE: Single data transfer.
- Data widths of 32 bits supported.
- State transitions for different types of transactions (IDLE, NONSEQ, SEQ, and BUSY) are implemented.

Unsupported Features:

- Advanced burst types such as WRAP are not supported in this module.
- No support for split, retry, or error handling mechanisms.
- Only HSIZE values supporting 4-byte transfers (word size) are implemented.

- Does not support multi-layer AHB implementations.

Hardware Architecture and FSMs State Diagram

Hardware Architecture:

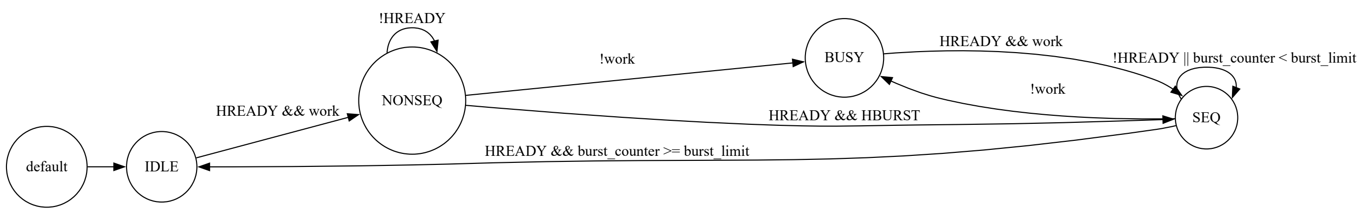
The AHB Master consists of:

- **Control Unit**: FSM responsible for controlling the transaction phases.
- **Data Path**: Handles the transfer of data between the master and the slave.
- **Burst Controller**: Manages bursts, ensuring correct sequence of address increments for burst transfers.
- **CPU Interface**: Connects the master with an external CPU, providing instruction and control information.

FSM State Diagram:

1. **IDLE State (HTRANS = 2'b00)**:
   - Default state where no transfers are taking place. The master waits for work from the CPU interface.
2. **NONSEQ State (HTRANS = 2'b10)**:
   - Non-sequential address phase, indicating a new transfer request has started.
3. **SEQ State (HTRANS = 2'b11)**:
   - Sequential state, where burst transfers take place in sequence without gaps in the address space.
4. **BUSY State (HTRANS = 2'b01)**:
   - When the master cannot continue transferring but needs to keep the bus occupied.

The FSM transitions based on the availability of work, HREADY signal, and burst conditions.



HDL Coding

The module is written in Verilog HDL, implementing AHB Master logic as per the AHB-Lite protocol.

Key RTL Components:

- **HTRANS Generation**: The FSM drives the HTRANS signal based on transaction phases.
- **Data Path**: Handles input/output data from the CPU and writes/reads to/from the AHB slave.

Code Snippet:

```
always @(posedge HCLK or negedge HRESETn) begin
    if (!HRESETn)
    begin
        HTRANS <= IDLE;
    end
    else
    begin
        case (HTRANS)
            IDLE:begin
                HADDR <= cpu_inst[63:32];
                HWDATA <= cpu_inst[31:0];
                HSIZE <= cpu_cont[6:4];
                HWRITE <= cpu_cont[0];
                HBURST <= cpu_cont[3:1];
                work <= cpu_cont[7];
                if (HREADY && work) begin
                    HTRANS <= NONSEQ;
                    burst_counter <= 0;
                end
            end
            BUSY:begin
                if (HREADY && work) begin
                    HTRANS <= SEQ;
                end
            end
            NONSEQ:begin
                if (HREADY) begin
                    if (!work)
                    HTRANS <= BUSY;
                    else if (HBURST) begin
                        HWDATA <= cpu_inst[31:0];
                        burst_counter <= burst_counter + 1;
                        HTRANS <= SEQ;
                    end else begin
                        HTRANS <= IDLE;
                        HWDATA <= cpu_inst[31:0];
                    end
                end
            end
            SEQ:begin
                if (HREADY) begin
                    HWDATA <= cpu_inst[31:0];
                    HADDR <= HADDR + (4 << HSIZE);
                        burst_counter <= burst_counter + 1;
                    if (!work)
                    HTRANS <= BUSY;
                    else if(HBURST == 3'b001 && burst_counter * HSIZE <
8'b1111111111 && burst_counter < num_beats)
                        HTRANS <= SEQ;

                        else begin
                        HTRANS <= IDLE;
                    end
                end
            end
            default:HTRANS <= IDLE;
        endcase
    end
end
```

Testing (Test Cases and Testbenches)

The `AHB_Master_tb` testbench is designed to verify the functionality of the `AHB_Master` module by simulating various AHB-Lite transactions. This includes both read and write operations, handling of burst transfers, and appropriate state transitions within the finite state machine (FSM).

Testbench Description

The testbench instantiates the `AHB_Master` module and drives its inputs to simulate different operational scenarios. It checks the outputs and internal state transitions to ensure that the `AHB_Master` behaves as expected under various conditions.

Test Case 1: Write Transaction (HSIZE = 4-byte, Incrementing Burst, Non-Sequential)

Objective:

- To verify the behavior of a write transaction with a 4-byte transfer size, incrementing burst type, and non-sequential transfer.

Procedure:

- The test initiates a write transaction by setting `cpu_inst` to `64'h000DD000_AAAAAAAA` (Address = 0xAAAAAAAA, Write Data = 0x00000000).
- Control signals (`cpu_cont`) are set to 8'b10010011, indicating:
  - HSIZE = 4-byte
  - HBURST = Incrementing Burst
  - HWRITE = 1 (Write operation)
- After some delay, the test checks if `HTRANS` transitions to `2'b10` (NONSEQ state) to indicate the start of the transaction.
- The test simulates a busy slave by setting `HREADY = 0` and later resumes with `HREADY = 1` to check if the transaction continues with a sequential transfer (`HTRANS = 2'b11`).

Result:

- **At time 30:** The transaction correctly transitioned to `NONSEQ` (non-sequential) as expected.
- **At time 80:** The transaction continued in `SEQ` (sequential) state, confirming the proper progression of the burst.

Conclusion: This test passed successfully as the transaction transitioned through the expected states.

Test Case 2: Read Transaction (Single Transfer, HSIZE = 4-byte, Non-Sequential)

Objective:

- To verify the behavior of a read transaction with a 4-byte transfer size, single burst type, and non-sequential transfer.

Procedure:

- The test initiates a read transaction by setting `cpu_inst` to `64'hBBBBBBBB_00000000` (Address = 0xBBBBBBBB).
- Control signals (`cpu_cont`) are set to 8'b10000010, indicating:
    - HSIZE = 4-byte
    - HBURST = Single
    - HWRITE = 0 (Read operation)
- The test checks if `HTRANS` transitions to `2'b10` (NONSEQ state) to indicate the start of the read transaction.
- A series of data values is simulated to be returned from the slave (`HRDATA`), ranging from `32'hDEADBEEF` to `32'hAAAAAAAA`.
- The test checks if the transaction ends properly by monitoring the transition of `HTRANS` to `2'b00`.

Result:

- **At time 120:** The transaction correctly transitioned to `NONSEQ` (non-sequential) as expected.
- **At time 180:** The transaction ended successfully, as `HTRANS` transitioned to `2'b00`.

Conclusion: This test passed successfully, as the transaction followed the correct sequence for a read operation and ended properly.

Test Case 3: Burst Transaction (Single Transfer, HSIZE = 4-byte, Sequential)

Objective:

- To verify a single transfer burst transaction with a 4-byte transfer size and sequential behavior.

Procedure:

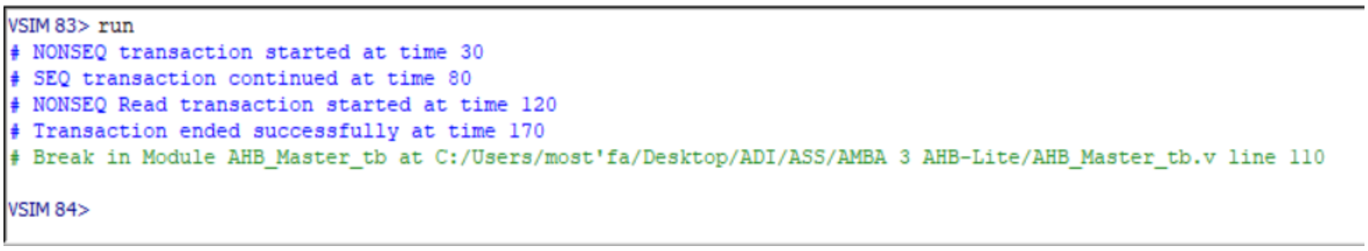
- The test initiates a burst transaction by setting `cpu_inst` to `64'hBBBBBBBB_00000000` (Address = 0xBBBBBBBB) with control signals (`cpu_cont = 8'b10000000`).
- The `HBURST` type is set to `Single`, and `HWRITE = 0` indicates that no data is being written.
- The `num_beats` is set to 5, although this value is ignored since the burst type is `Single`.

Result:

- The simulation proceeds without any errors or issues during the transaction.

Conclusion: This test checked the handling of a single transfer burst, which successfully completed without any anomalies.

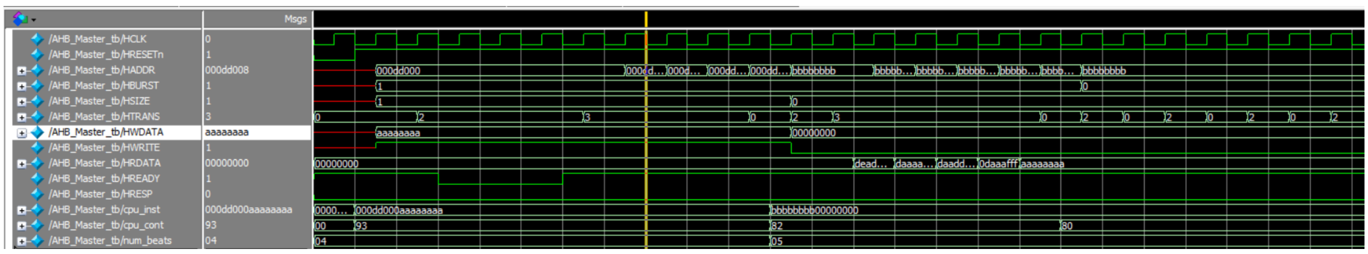
All cases were successful



```
VSIM 83> run
# NONSEQ transaction started at time 30
# SEQ transaction continued at time 80
# NONSEQ Read transaction started at time 120
# Transaction ended successfully at time 170
# Break in Module AHB_Master_tb at C:/Users/most'fa/Desktop/ADI/ASS/AMBA 3 AHB-Lite/AHB_Master_tb.v line 110

VSIM 84>
```

Wave Form



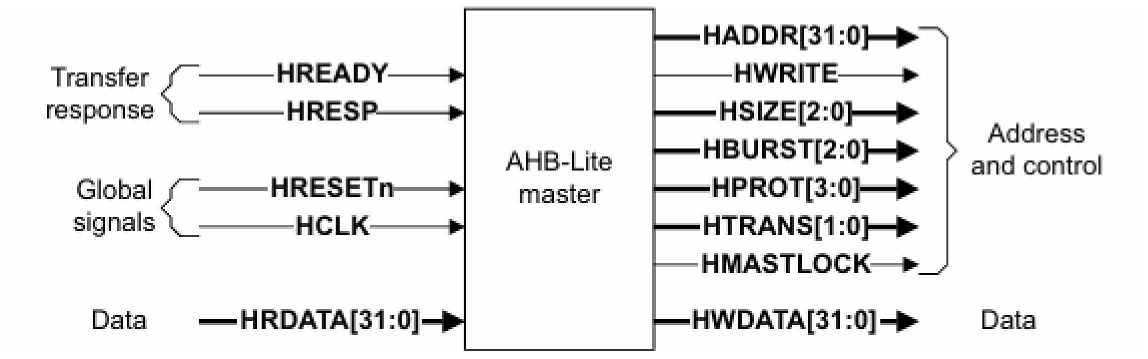
Signal Descriptions

Signal Name	Width	Direction	Description
HCLK	1	Input	Clock signal for the module.
HRESETn	1	Input	Active-low reset signal to initialize the module.
HADDR	32	Output	Address of the transaction being initiated by the master.
HBURST	3	Output	Burst type of the transaction (SINGLE, INCR, WRAP).
HSIZE	3	Output	Indicates the size of the data being transferred.
HTRANS	2	Output	Indicates the type of transfer (IDLE, BUSY, NONSEQ, SEQ).
HWDATA	32	Output	Data to be written to the slave during a write transaction.
HWRITE	1	Output	Indicates whether the current transfer is a write (1) or read (0).
HRDATA	32	Input	Data read from the slave during a read transaction.
HREADY	1	Input	Indicates whether the slave is ready to proceed with the transaction.
HRESP	1	Input	Indicates an error (1) or OK (0) response from the slave.
cpu_inst	64	Input	Contains the instruction from the CPU (address and data). " CPU signal "
cpu_cont	8	Input	Control signals from the CPU (size, burst, work). " CPU signal "

Signal Name	Width	Direction	Description
num_beats	8	Input	Number of beats that the master need to send in Incrementing burst of undefined length. " CPU signal "

Detailed Descriptions:

- **HCLK**: The main clock signal that drives the timing of the AHB Master module.
- **HRESETn**: An active-low reset signal that initializes or resets the state of the AHB Master module.
- **HADDR**: Carries the memory address for read/write transactions initiated by the master and the firet 2 bits select the slave.
- **HBURST**: Specifies the type of burst transaction, such as single, incremental (INCR), or wrapped (WRAP).
- **HSIZE**: Indicates the size of the data for the current transaction (e.g., 4 bytes for 32-bit data).
- **HTRANS**: Defines the type of transfer, including IDLE, BUSY, NON-SEQ (non-sequential), and SEQ (sequential).
- **HWDATA**: The data to be written to the slave during write transactions.
- **HWRITE**: Signals whether the current operation is a write (`1`) or a read (`0`).
- **HRDATA**: The data read from the slave during read transactions.
- **HREADY**: Indicates if the slave is ready to accept the next transaction or is still processing the current one.
- **HRESP**: Provides feedback from the slave, where `1` indicates an error and `0` indicates success.
- **cpu_inst**: Contains both the address (upper 32 bits) and data (lower 32 bits) from the CPU for the current operation.
- **cpu_cont**: Control signals from the CPU that include burst size, data size, and the direction of the transfer (read or write).
- **num_beats**: Number of beats that the CPU will send to the master and it only needed in Incrementing burst of undefined length



Downloading & Runing the Simulation

Downloading the Repository

To download the testbench and associated files, follow these steps:

1. Clone the repository from GitHub:

```
git clone https://github.com/mgma10/ahb_master_project.git
```

2. Navigate to the project directory:

```
cd ahb_master_project
```

Running the Simulation

The simulation can be run using the run.do script provided in the repository. This script automates the compilation and simulation process.

1. Open your simulation tool (e.g., ModelSim).

2. Load the run.do script:

bash

```
do run.do
```

3. The script will compile the testbench and run the simulation. You can view the results in the waveform window.