Documentation for AXI DMA Master Design

Design Choices

This module implements a Direct Memory Access (DMA) controller using the AXI protocol. The design choices include:

1. **State Machine-Based Control:** Two separate state machines manage **read and write** operations to ensure data integrity and proper handshaking with the AXI bus.
2. **FIFO for Intermediate Storage:** A synchronous FIFO buffers the data between AXI read and write transactions to decouple the two operations.
3. **AXI-Lite Protocol:** The design uses AXI-Lite handshaking for reliable data transfers, ensuring compatibility with standard AXI interfaces**.**
4. **Configurable Transfer Length:** The length of the data transfer is controlled via an input signal (length), making the module flexible.
5. **Trigger-Based Execution:** The DMA transaction begins upon receiving a trigger signal, allowing external control of transfers.
6. **Synchronous Reset: T**he module supports a synchronous reset mechanism to ensure a clean start for the read and write operations.
7. **Byte-Address Calculation**: Since AXI expects byte addresses, word addresses must be converted before sending to the slave memory.

Byte Address Conversion

AXI uses byte addressing, while memory operations might be based on word addressing. To ensure proper access, the word address must be converted to a byte address before being sent to the AXI bus. A dedicated function is implemented in Verilog to perform this conversion:

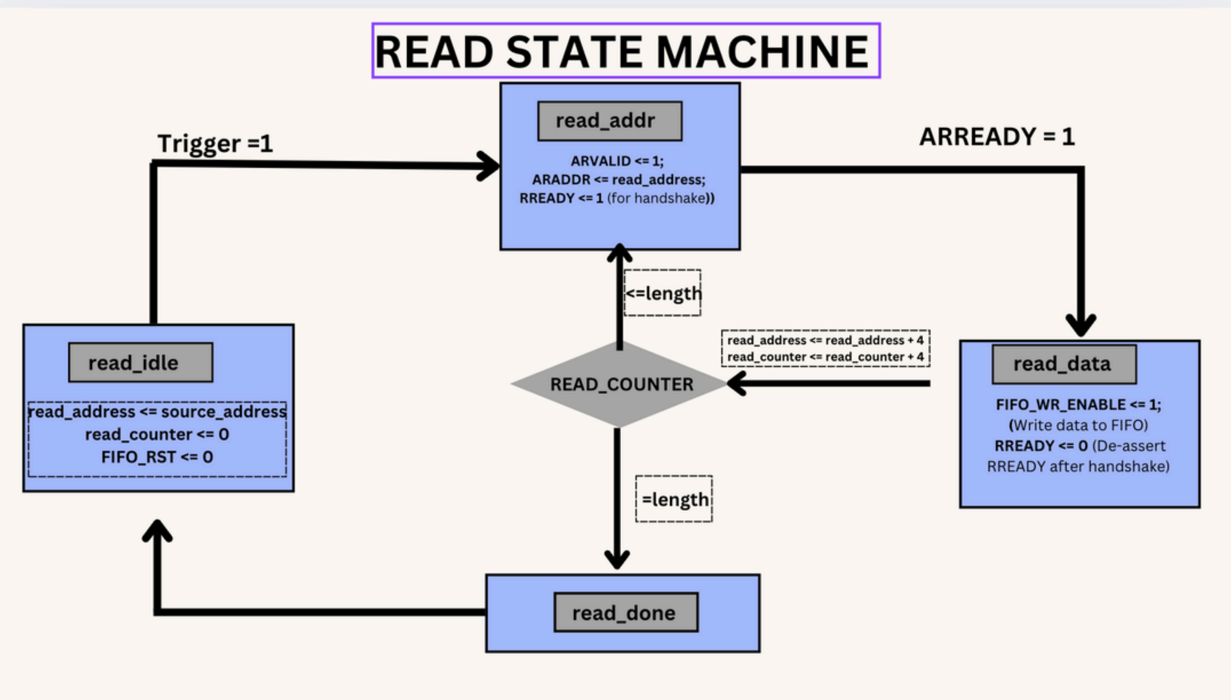
```
function [31:0] convert_to_byte_address;
    input [31:0] word_address;
    input [3:0] word_size; // Typically 4 bytes for 32-bit
begin
    convert_to_byte_address = word_address * word_size;
end
endfunction
```

State Machine Logic

Read State Machine

The read state machine controls fetching data from the source address via AXI.

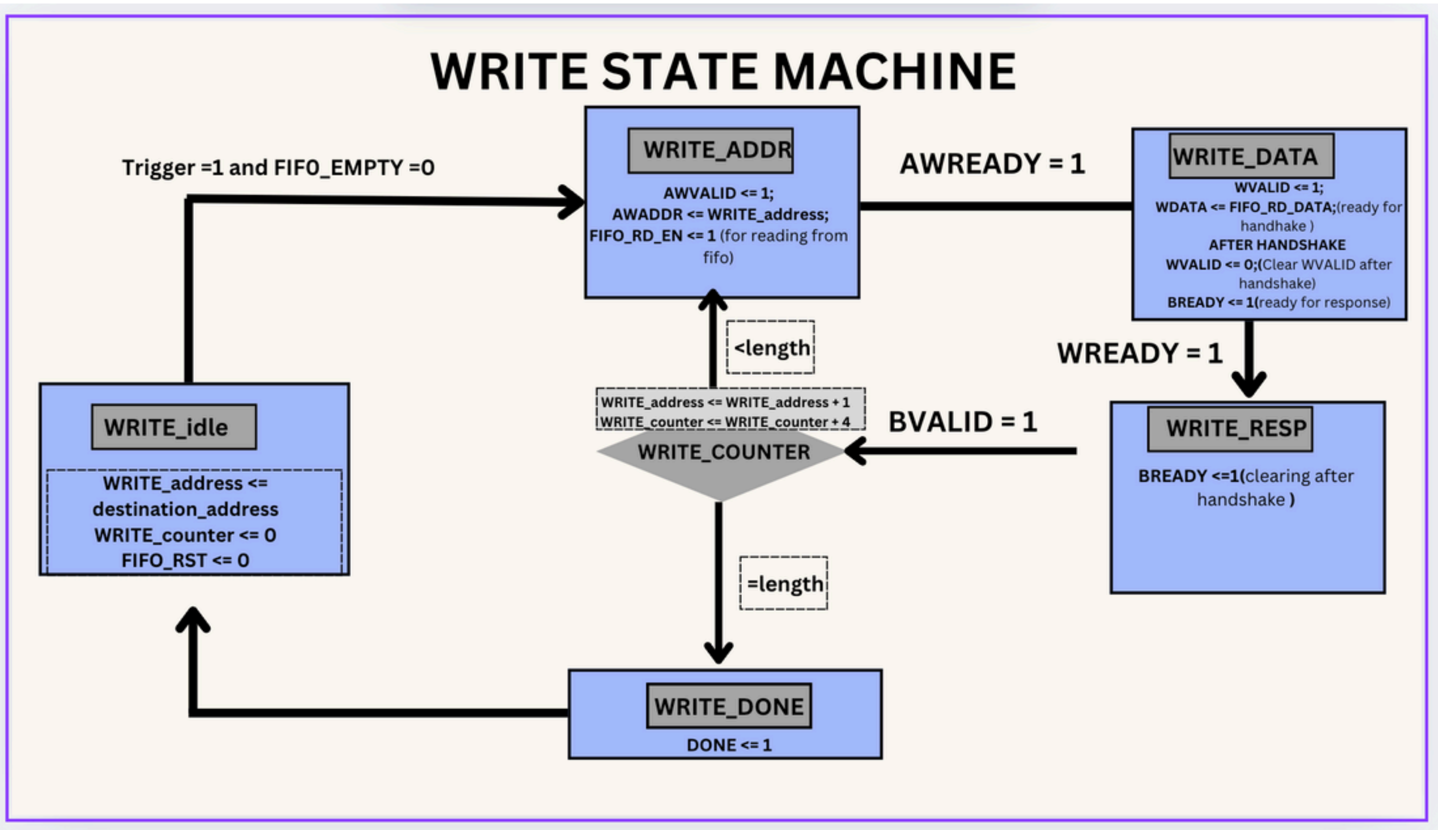
State	Description
READ_IDLE	Waits for the trigger signal to start the transaction.
READ_ADDR	Sends the read address to the AXI bus and waits for address acceptance.
READ_DATA	Waits for the data to be available from the AXI bus and writes it to the FIFO.
READ_DONE	Completes the read operation and transitions to READ_IDLE.



Write State Machine

The write state machine manages storing data into the destination address via AXI.

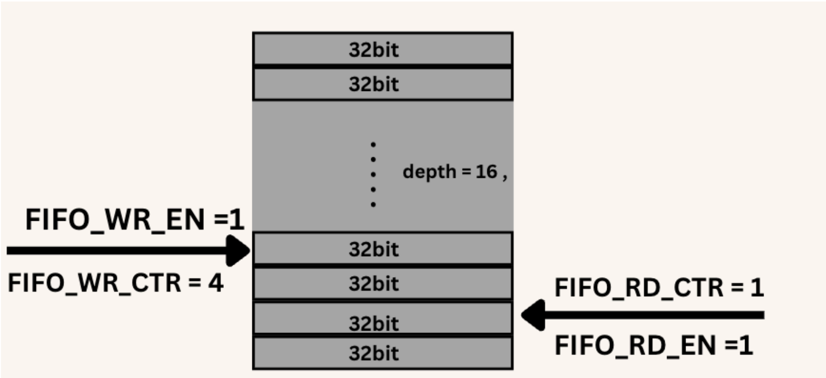
State	Description
WRITE_IDLE	Waits for data in the FIFO and a trigger signal to start the transaction.
WRITE_ADDR	Sends the write address to the AXI bus and waits for address acceptance.
WRITE_DATA	Reads data from the FIFO and writes it to the AXI bus.
WRITE_RESP	Waits for write response from the AXI bus.
WRITE_DONE	Completes the write operation and transitions to WRITE_IDLE.



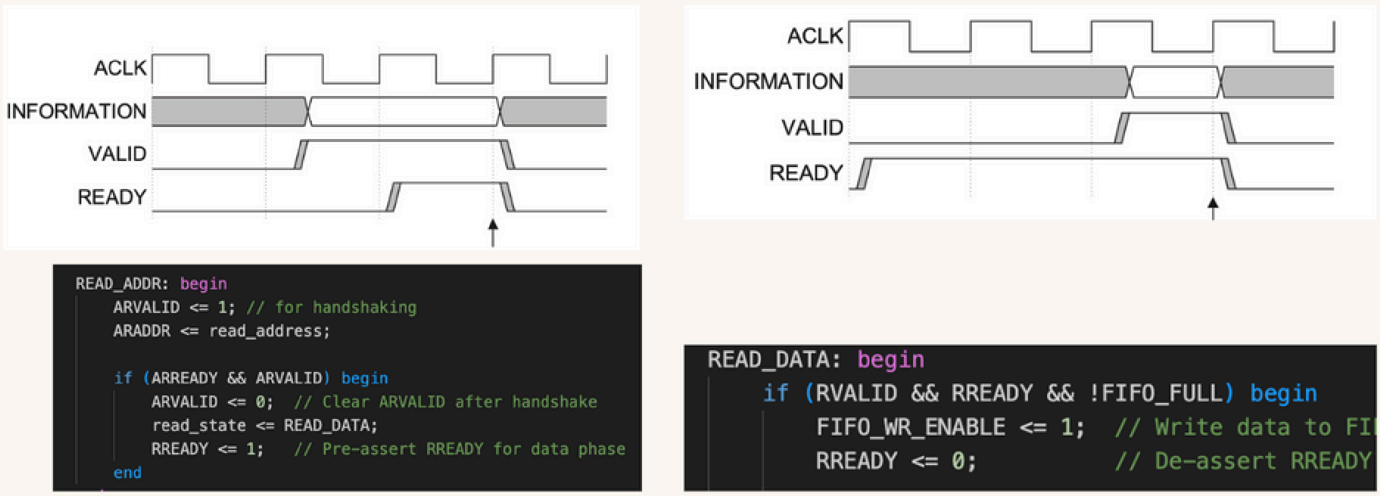
FIFO Usage

A 16-depth synchronous FIFO is used to buffer data between read and write transactions. The FIFO allows read and write operations to proceed independently, improving efficiency. The FIFO maintains:

- **FIFO_WR_ENABLE:** Signals when data should be written into the FIFO.
- **FIFO_RD_EN:** Signals when data should be read from the FIFO.
- **FIFO_EMPTY & FIFO_FULL:** Flags indicating FIFO status to prevent overflow or underflow.
- **FIFO_WR_PTR & FIFO_RD_PTR:** Pointers for write and read operations to manage FIFO entries.



AXI-Lite Handshaking Methodology



ARVALID signal waits for ARREADY signal for handshaking

RREADY signal wait for the RVALID signa for handshaking

The module follows the AXI-Lite protocol for reliable data transfers:

1. **Read Address Channel (ARADDR, ARVALID, ARREADY)**
   - The module asserts ARVALID with the read address (ARADDR).
   - It waits for ARREADY from the AXI slave before proceeding to the read data phase.
2. **Read Data Channel (RDATA, RVALID, RREADY)**
   - The slave asserts RVALID when data is ready.
   - The module asserts RREADY to receive data and stores it in FIFO if not full.
3. **Write Address Channel (AWADDR, AWVALID, AWREADY)**
   - The module asserts AWVALID with the write address (AWADDR).
   - It waits for AWREADY before proceeding to the data phase.

4. **Write Data Channel (WDATA, WVALID, WREADY)**
   - The module asserts WVALID with data (WDATA) read from the FIFO.
   - It waits for WREADY before proceeding to the write response phase.
5. **Write Response Channel (BVALID, BREADY, BRESP)**
   - The slave asserts BVALID with a response.
   - The module asserts BREADY to acknowledge the response and transition the state machine.

This handshaking ensures that data transfers occur correctly without collisions or data loss.

SIMULATION: