MASTER DMA CONTROLLER IMPLEMENTATION

Design Overview:

This design implements an AXI-Lite-based data transfer system, where data is read from a source address and written to a destination address using AXI-Lite transactions. The module uses a finite state machine (FSM) to control the read and write operations, ensuring proper handshaking with AXI-Lite signals. A FIFO buffer is used to temporarily store data during the transfer process.

State Machine Logic:

The design includes two FSMs:

1. Read FSM:

- READ_NONE: Idle state. When the trigger is high, the read process starts.
- READ_ADDR: Sends the read address (ARADDR) and asserts ARVALID. Waits for ARREADY from the AXI-Lite interface.
- READ_DATA: Once ARREADY is received, waits for RVALID to get data (RDATA).
- READ_STORE: Stores the received data into the FIFO buffer and decrements r_count.
- READ_DONE: If more data needs to be read, increments ARADDR and repeats the process; otherwise, de-asserts RREADY.

2. Write FSM:

- WRITE_NONE: Idle state. When the trigger is high, the write process starts.
- WRITE_ADDR: Sends the write address (AWADDR) and asserts AWVALID. Waits for AWREADY from the AXI-Lite interface.
- WRITE_DATA: Waits for WREADY and writes data from the FIFO buffer (WDATA).
- WRITE_VALID: Asserts BREADY and waits for BVALID.
- WRITE_DONE: If more data needs to be written, increments AWADDR and repeats the process; otherwise, asserts done.

FIFO Usage:

The FIFO buffer is used to temporarily store data between the read and write processes, ensuring continuous data flow without loss. When a read transaction occurs, FIFO_WR is asserted, and the received data (RDATA) is stored in the FIFO. The write process then retrieves data from the FIFO by asserting FIFO_RD. To avoid overflow, the FIFO tracks its depth using a counter, with FIFO_FULL preventing new writes when the buffer is full and FIFO_EMPTY preventing reads when there is no data available. Circular indexing is used to efficiently manage memory without unnecessary shifting.

AXI-Lite Handshaking Methodology:

The AXI-Lite protocol ensures proper synchronization between the master and slave using a structured handshaking process. During a read transaction, the master initiates a request by asserting ARVALID along with the read address (ARADDR). The slave acknowledges this by asserting ARREADY. Once the address phase completes, the slave places the data on the bus, asserting RVALID, and the master acknowledges it by asserting RREADY. For a write transaction, the master first sends AWVALID with AWADDR, waiting for AWREADY from the slave. Then, it places the write data (WDATA) on the bus with WVALID, which the slave acknowledges with WREADY. After the data is successfully written, the slave asserts BVALID to indicate completion, and the master responds with BREADY. This systematic handshaking ensures data integrity and proper coordination between read and write operations.