Birzeit University

Department of Electrical & Computer Engineering

Summer - 2024/2025

ENCS3310 Advanced Digital Systems Design

Dr. Ayman Hroub

Course Project

Design and Verification of a Simplified Single-Channel DMA Controller for UART-to- Memory Data Transfer

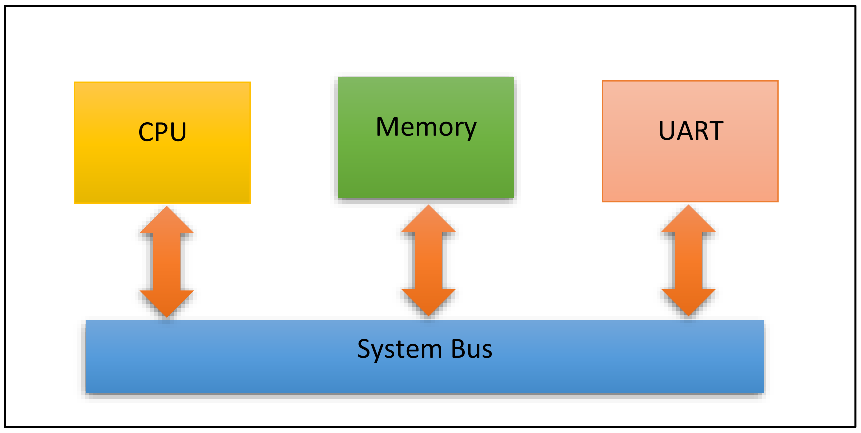
Deadline August 9, 2025

1. Project Objectives

1. Design a simplified DMA (Direct Memory Access) controller in Verilog that facilitates data transfer from a simplified UART (Universal Asynchronous Receiver/Transmitter) peripheral to memory
2. Develop a comprehensive TestBench to verify the functional correctness of this DMA controller

2. The DMA Controller Specifications

The DMA controller is a hardware unit that allows data transfer between Input/output devices and memory without continuous involvement from the CPU. In this project, you are required to design a simplified single-channel DMA controller that allows data transfer from a UART peripheral to a memory. The figure below shows the block diagram of such system.



Simplified UART Model

Develop a simplified UART model that abstracts away serial line behavior and provides buffered byte access to simplify integration with the DMA controller. This model has a small internal buffer preloaded with some fixed data, where the DMA controller reads data directly from this local buffer and transfers it to the memory. The table below lists the input/output signals of this simple UART

Signal	Direction	Description
clk	input	Clock
rstn	input	Active low asynchronous reset signal that clears the UART internal buffer
re	input	Read enable from the DMA to read the next byte
Data_out	output	8-bit data output to the DMA controller, which will be ready one cycle after **re** signal is asserted

Memory Module

You are required to design a very simple memory model to store the data the DMA controller reads from the UART. The table lists the input/output signals of this memory model.

Signal	Direction	Description
clk	input	Clock
rstn	input	Active low asynchronous reset signal
write_address	input	Address to write in memory
data_in	input	8-bit data to write into memory
we	input	Write enable
read_address	input	Address to read from memory
data_out	output	8-bit data out from memory
re	input	read enable

DMA Controller

In order to start the DMA transfer, the DMA controller receives a request from the CPU. In this project, you are not required to design any CPU model, and you can provide these signals in the TestBench. The CPU request includes the following signals:

1. **Start** signal asking for DMA transfer
2. **start_address** which is the destination start address in the memory, where the DMA controller writes the data it receives from the UART into the memory
3. **transfer_size** which refers to the data size in the number of bytes that the DMA controller must transfer from the UART to the memory. The DMA controller must implement and maintain any required internal registers/counters to perform such transfer properly.

After the transfer completes, the DMA controller notifies the CPU by asserting the **done** signal. The table below shows the input/output signals of the DMA controller.

Signal	Direction	Description
clk	input	Clock
rstn	input	Active low asynchronous reset signal
start	input	Start DMA transfer
uart_data	input	8-bit data comes from the UART
start_address	input	Starting destination address in memory
transfer_size	input	Number of bytes to transfer
uart_read_enable	output	Signal to UART to read next byte
memory_write_address	output	Address to write in memory
memory_write_data	output	Data to write into memory
memory_write_enable	output	Write enable for memory
done	output	Indicates DMA transfer completion

Use FSM (Finite State Machine) to implement the DMA transfer cycle, where the DMA controller can be in one of the following three staes

- **idle** waiting DMA reauests from CPU.
- **read_uart** asserting uart_read_enable and receiving data from UART.
- **write_memory** asserting memory_write_enable and writing the received UART data to memory.

Note:

Your design must be parameterized. For any design parameter not explicitly specified, such as, memory depth, data width, and register widths, you may assume a reasonable default value of your choice.

3. Team Work

- You may work in teams of up to two students.
- Teams of more than two will not be allowed.
- All team members are expected to contribute equally in all project phases: design, implementation, verification, and documentation.

4. Project Deliverables

1. RTL design source code of the complete system including the required DMA controller and a simple memory and UART models
2. A Comprehensive TestBench to simulate the design and test its functional correctness
3. A well-structured report, which includes the following components in a single document:
    - Design and implementation detailed description and block diagrams
    - Simulation snapshots for all tests

5. Submission Guidelines

- Submissions must be made by replying to the assignment on Ritaj. Submissions through any other means will not be accepted.
- Late submissions will not be accepted under any circumstances.
- Only one team member should submit the project, i.e., there must be one submission per team only.
- The full names and university IDs of all team members must be included both in the message body of the submission and on the cover page of the report.
- The submission must include the project report and source code, packaged together in a single compressed folder.

6. Assessment Criteria

Report	10%
The way of discussion	10%
Code organization and documentation	10%
DMA Controller design and implementation	20%
UART model design and implementation	15%
Memory model design and implementation	10%
Verification (TestBench)	25%
Total	**100**