# Lab 7.2: Asynchronous FIFO (Clock Domain Crossing)

## Objective

The objective of this lab is to design and verify an **asynchronous FIFO (First-In-First-Out)** buffer that allows data transfer between two different clock domains. The design ensures safe clock domain crossing using Gray code pointers and synchronization techniques while maintaining reliable FIFO operation.

## Design Specifications

1. **FIFO Parameters**
   - Data width: Configurable (default 8 bits).
   - Depth: Configurable (default 16 entries).
2. **Clock Domains**
   - Separate write and read clock signals.
   - Write and read operations operate independently.
3. **Key Features**
   - **Gray Code Pointers**: Used for safe pointer comparison across domains.
   - **Multi-Flop Synchronizers**: Two-stage synchronizers to reduce metastability when transferring pointer values.
   - **Full/Empty Detection**:
     - **Full flag**: Indicates no more data can be written.
     - **Empty flag**: Indicates no more data can be read.

## Critical Design Points

1. **Pointer Handling**
   - Binary counters increment for read and write addresses.
   - Converted to Gray code before synchronization.
2. **Metastability Protection**
   - Two-stage flip-flop synchronizers used for transferring Gray-coded pointers between clock domains.
3. **Flag Generation**
   - **Full Condition**: Write pointer in Gray code is compared with synchronized read pointer; full occurs when they match except for the MSB inversion.
   - **Empty Condition**: Read pointer equals synchronized write pointer.
4. **Reset Handling**
   - Separate resets are provided for write and read clock domains.
   - Ensures FIFO starts empty and pointers reset to zero.

## Design Methodology

- **Write Side**: Data written into memory when write enable is high and FIFO is not full. Pointer increments each cycle.
- **Read Side**: Data read from memory when read enable is high and FIFO is not empty. Pointer increments accordingly.
- **Clock Domain Crossing**: Pointers are Gray coded, synchronized, and then compared for flag generation.

## Simulation and Verification

The asynchronous FIFO was verified through a structured testbench with different clock frequencies:

1. **Reset Phase**: Both domains reset independently.
2. **Write Test**: 10 random data values written into the FIFO at the write clock rate.
3. **Read Test**: 10 values read out at the read clock rate, confirming FIFO ordering (First-In-First-Out).
4. **Clock Mismatch Verification**: Write clock (100 MHz) and read clock (~71 MHz) demonstrated correct data transfer despite frequency differences.
5. **Full and Empty Flags**: Correctly asserted and deasserted at the appropriate times.

Simulation waveforms and printed logs confirmed:

- Safe synchronization of pointers.
- No metastability issues.
- Correct handling of full and empty states across clock domains.

## Results

- FIFO successfully transferred data between two different clock domains.
- Gray code pointers and synchronizers ensured reliable operation without glitches.
- Full and empty flags behaved correctly.
- Design remained stable under different write/read speeds.

## Conclusion

The asynchronous FIFO design met all the specifications for reliable data transfer across independent clock domains. This lab reinforced key digital design concepts such as:

- **Clock domain crossing** and metastability protection.
- **Gray code pointer usage** for safe synchronization.
- **FIFO memory design** and flag generation logic.