Asynchronous FIFO Design – Project Report

Meher Adbol

August 17, 2025

Executive Summary

This report presents the design, implementation, and verification of a 64-depth Asynchronous FIFO using Verilog HDL. The design successfully addresses the critical challenge of data transfer between two clock domains (80 MHz write and 25 MHz read) while preventing metastability issues through the use of gray code counters and dual-flop synchronizers.

Project Specifications:

- FIFO Depth: 64 elements
- Data Width: 8 bits
- Write Clock: 80 MHz
- Read Clock: 25 MHz
- Burst Length: 80 elements
- Technology: Verilog HDL

1 Introduction

1.1 Problem Statement

In modern digital systems, data transfer between different clock domains is a common requirement. However, this introduces the risk of metastability, where signals can become unstable when sampled at clock domain boundaries. Asynchronous FIFOs provide a robust solution for this problem by implementing proper synchronization techniques.

1.2 Objectives

1. Design a 64-depth asynchronous FIFO with 8-bit data width
2. Implement gray code counters to prevent metastability

3. Use dual-flop synchronizers for reliable cross-clock domain communication
4. Support 80 MHz write and 25 MHz read operations
5. Provide comprehensive status flags (full, empty, almost full, almost empty)
6. Verify the design with extensive testing including burst operations

1.3 Design Challenges

- **Metastability Prevention**: Ensuring reliable signal transfer between clock domains
- **Pointer Synchronization**: Maintaining accurate read/write pointer synchronization
- **Performance Optimization**: Achieving maximum throughput with minimal latency
- **Resource Efficiency**: Minimizing area and power consumption

2 Design Methodology

2.1 Architecture Overview

The asynchronous FIFO design follows a modular approach with five main components:

1. Top-level FIFO Module (`async_fifo.v`)
2. Gray Code Counter (`gray_counter.v`)
3. Dual-Flop Synchronizer (`dual_flop_sync.v`)
4. Gray to Binary Converter (`gray_to_binary.v`)
5. FIFO Control Logic (`fifo_control.v`)

2.2 Key Design Principles

2.2.1 Gray Code Counters

Gray code counters are essential for preventing metastability in asynchronous FIFOs.
Advantages:

- Single Bit Transition
- Reduced Metastability Risk
- Efficient XOR-based implementation

Gray Code Generation:

```
gray_count[WIDTH-1] = bin_count[WIDTH-1];
for (i = 0; i < WIDTH-1; i = i + 1) begin
    gray_count[i] = bin_count[i] ^ bin_count[i+1];
end
```

2.2.2 Dual-Flop Synchronizers

Provide reliable cross-clock domain signal transfer:

- Two-Stage Synchronization
- Configurable Width
- Reset Capability

```
always @(posedge clk_dest or negedge rst_n_dest) begin
    if (!rst_n_dest) begin
        sync_ff1 <= {WIDTH{1'b0}};
        data_out <= {WIDTH{1'b0}};
    end else begin
        sync_ff1 <= data_in;
        data_out <= sync_ff1;
    end
end
```

2.3 Design Flow

1. Requirements Analysis
2. Architecture Design
3. RTL Implementation
4. Integration
5. Verification
6. Validation

3 Implementation Details

3.1 Module Descriptions

3.1.1 Top-Level FIFO Module (`async_fifo.v`)

Parameterized top module handling memory, pointers, and interfaces.

```
module async_fifo #(
    parameter DATA_WIDTH = 8,
    parameter ADDR_WIDTH = 6,
    parameter FIFO_DEPTH = 64
)(
    input wire wr_clk, wr_rst_n, wr_en,
    input wire [DATA_WIDTH-1:0] wr_data,
    output wire full,
    input wire rd_clk, rd_rst_n, rd_en,
    output wire [DATA_WIDTH-1:0] rd_data,
    output wire empty,
    output wire almost_full, almost_empty
);
```

3.1.2 Other Modules

- `gray_counter.v`: binary + gray code generation
- `dual_flop_sync.v`: two-stage synchronization
- `gray_to_binary.v`: XOR-based conversion
- `fifo_control.v`: status flag logic

3.2 Memory Organization

```
reg [DATA_WIDTH-1:0] fifo_mem [0:FIFO_DEPTH-1];
```

4 Verification and Testing

4.1 Testbench Architecture

`async_fifo_tb.v` verifies operation with:

- Clock generation (80 MHz write, 25 MHz read)
- Data generation and burst tests
- Full/empty and almost flag scenarios
- Reset and concurrent operation tests

4.2 Simulation Results

All 7 test scenarios passed with 0 errors. Data integrity maintained.

5 Analysis and Discussion

- Modularity and Scalability
- Metastability Prevention
- Performance and Latency Analysis
- Resource Utilization: ∼200 LUTs, 50 FFs, 512-bit memory

6 Conclusions and Future Work

6.1 Achievements

All design specifications and verification goals were met.

6.2 Future Enhancements

- Configurable depth/width
- Advanced flow control
- Formal verification

6.3 Lessons Learned

Importance of Gray codes, dual-flop synchronization, comprehensive test coverage, and documentation.